THE MONADIC SECOND-ORDER LOGIC OF GRAPHS XVI: CANONICAL GRAPH DECOMPOSITIONS

BRUNO COURCELLE

LaBRI, Bordeaux 1 University, 33405 Talence, France
e-mail address: courcell@labri.fr

ABSTRACT. This article establishes that the *split decomposition* of graphs introduced by Cunnigham, is definable in Monadic Second-Order Logic.This result is actually an instance of a more general result covering canonical graph decompositions like the modular decomposition and the Tutte decomposition of 2-connected graphs into 3-connected components. As an application, we prove that the set of graphs having the same cycle matroid as a given 2-connected graph can be defined from this graph by Monadic Second-Order formulas.

1. INTRODUCTION

Hierarchical graph decompositions are useful for the construction of efficient algorithms, and also because they give structural descriptions of the considered graphs. Cunningham and Edmonds have proposed in [18] a general framework for defining decompositions of graphs, hypergraphs and matroids. This framework covers many types of decompositions. Of particular interest is the *split decomposition* of directed and undirected graphs defined by Cunningham in [17].

A hierarchical decomposition of a certain type is *canonical* if, up to technical details like vertex labellings, there is a unique decomposition of a given graph (or hypergraph, or matroid) of this type. To take well-known examples concerning graphs, the *modular decomposition* is canonical, whereas, except in particular cases, there is no useful canonical notion of *tree-decomposition* of minimal tree-width. The general results of [18] define canonical decompositions.

The present article shows that many of these canonical decompositions can be defined by *monadic second-order (MS) formulas* "inside" the considered graphs or hypergraphs (we do not consider decompositions of matroids in this article). More precisely, we prove that under the quite natural and generally satisfied hypothesis that the elementary decomposition steps are definable by an MS formula, the mapping from a graph or a hypergraph to the *tree* representing its canonical decomposition (of the type under consideration) is a *monadic second-order (MS) transduction*, i.e., a transformation of relational structures defined by MS formulas. Furthermore, in many concrete cases, a certain relational structure based

on this tree represents the considered decomposition, in such a way that the decomposed graph can be reconstructed from it. We call it a *graph representation of the decomposition* in the case where it uses relations of arity at most 2. Otherwise, we call it a *hypergraph representation* (because relational structures can be viewed as directed labelled ranked hypergraphs). These representations can be constructed from the graphs (equipped with arbitrary linear orderings of their sets of vertices or edges) by MS transductions. Roughly speaking, we obtain that, from the point of view of MS logic, a graph is equivalent to the graph or hypergraph representation of its canonical decomposition, which means that an MS property of the canonical decomposition of a graph G is (equivalent to) an MS property of G and conversely, that an MS property of G is (equivalent to) an MS property of the (hyper)graph representation of its canonical decomposition.

This article contributes to the understanding of the power of MS logic for representing graph properties and graph theoretical notions like canonical graph decompositions and equivalences on graphs. When a graph property is expressible in MS logic, it can be checked in polynomial time on graphs of bounded tree-width or clique-width. When a graph transformation is expressible in MS logic, it preserves the property that a set has bounded tree-width or clique-width. We refer the reader to [10] and [14] for detailed expositions of these consequences.

Why are canonical decompositions interesting? Canonical decompositions and their (hyper)graph representations are interesting for several reasons.

First they contain useful informations on the structure of the graphs. This structural information has two forms: the tree, and the elementary graphs from which the considered graph is built. In most cases, hierarchical decompositions can be viewed as constructions of graphs or hypergraphs by means of particular composition operations (like *graph substitution* in the case of modular decomposition) in terms of *prime* graphs or hypergraphs, i.e., of those which are undecomposable. We will discuss this "algebraic" aspect whenever relevant.

Second, hierarchical graph decompositions are useful for the construction of polynomial algorithms. For example, the first step of the polynomial algorithm recognizing circle graphs by Bouchet [2] consists in constructing the split decomposition of the given graph. It uses the fact that a graph is a circle graph if and only if each component of its split decomposition is a circle graph. The planarity testing algorithm by Hopcroft and Tarjan [25] begins with the decomposition of a graph into 3-connected components. Hence a good understanding of hierarchical graph structure is useful for constructing low degree polynomial algorithms.

Third the (hyper)graph representation of the canonical decomposition of a graph requires in many cases less space to be stored than the given graph.

Finally, canonical decompositions are also useful for establishing logical properties. For example, it is proved in [12] that Seese's Conjecture holds for *interval graphs*, and that it holds in general if and only if it holds for *comparability graphs*. The proof makes an essential use of modular decompositions. (Seese's Conjecture says that if a set of graphs has a decidable satisfiability problem for MS logic, then it has bounded clique-width. A slight weakening of this conjecture is established in [16]).

The companion article [6] develops an application of split decomposition to *circle graphs* that we review briefly. A circle graph is the *intersection graph* of a *set of chords of a circle*. A graph is a circle graph if and only if all components of its split decomposition are circle graphs. Those components which are prime are *uniquely representable* as intersection graphs of sets of chords. It is proved in [6] that the unique representation of a prime circle graph

can be defined by MS formulas (one uses a description of sets of chords by finite relational structures). From the split decomposition of a circle graph G and the chord representations of its prime components, one can define all chord representations of G. And this can be done by MS formulas, since the split decomposition and the chord representations of the prime components of G can be defined by MS formulas. Hence, from a given circle graph, one can define by MS formulas (using also linear orders of the sets of its vertices), all chord representations of this graph. (To be precise, this construction rests upon a result by Courcelle and Oum [16] which uses MS formula with set predicates of the form $Even(X)$ expressing that a set X has even cardinality.)

In the present article, we prove a result of the same general form: all 2-connected graphs *equivalent* to a given graph G, in the sense that they have the same cycle matroid, can be defined from G and the linear orderings of its vertices by a fixed MS transduction. As for circle graphs, the proof uses a canonical decomposition of the considered graph, constructed by MS formulas, from which can be defined all the equivalent graphs. This construction is based on *Whitney's 2-isomorphism theorem*, which characterizes the graphs equivalent to G as those derived from G by a sequences of transformations called *twistings*.

Main results and overview of the article. First, we give a general set theoretical and logical setting in which decompositions of combinatorial structures can be defined. This framework covers actually two cases. In the first case, studied in Section 2, the decomposition tree is rooted and directed. The fundamental example is the very well-known *modular decomposition*. The decompositions of this type correspond to definitions of graphs by algebraic expressions based on graph operations. In Section 3, we consider the second case, where the decomposition tree is unrooted and undirected. In both cases, and under easily applicable conditions, we prove that the decomposition tree is definable by MS formulas, which generalizes the MS definability result of the modular decomposition of [9]. In Section 4 we develop the application to the *split decomposition* of Cunnigham [17] and we prove its MS definability, which is our second main result. We do not assume the reader familiar with this notion and this section presents it in detail. We prove in Section 2 the "logically effective" version of the above mentioned theorem by Whitney. Appendices 1 and 2 review definitions, basic properties and technical lemmas on MS logic, MS transductions and clique-with. This work has been presented at the International Conference on Graph Theory, Hyères, France, in September 2005.

2. Partitive families of sets

Trees, graphs and relational structures are finite. Two sets *meet* if they have a nonempty intersection. They *overlap* if they meet and are incomparable for inclusion. We write $A \perp B$ if and only if A and B overlap. The terms *minimal, least,* and *maximal* applied to sets refer, unless otherwise specified, to inclusion.

2.1. **Rooted trees and families of sets.** We define the family of sets associated with a partition of a set V, the blocks of which form a rooted tree. This is a generalization of the modular decomposition of a graph where V is its set of vertices.

Definition 2.1. *Set families and trees.* A *rooted tree* T has its edges directed so that every node is accessible from the root by a unique directed path. The *leaves* are the nodes of outdegree 0. The other nodes are the *internal nodes.* The set of nodes is denoted by N_T and the set of internal nodes by N_T^{int}. Although a tree is a graph, we will use the term "nodes" for the vertices of a tree (or a forest). This particular terminology will be useful for clarity in situations where we discuss simultaneously a graph and a tree representing it. A rooted tree is *proper* if no node has *outdegree* 1, hence if every node is either a leaf, or has at least two *sons.* We denote the son relation by son_T.

Let T be a rooted tree and $\mathcal{V} = (V(u))_{u \in N_T}$ be a partition of a nonempty set V such that $V(u)$ is nonempty if u is a leaf (but is possibly empty otherwise). For each node u, we let $T(u)$ be the rooted subtree consisting of u (its root) and the nodes reachable from u by a directed path. We let $\overline{V}(u)$ be the union of the sets $V(z)$ where z is a node of $T(u)$. Hence, $\overline{V}(u) = V$ if u is the root. The family $\mathcal{F} = \mathcal{F}(T, \mathcal{V})$ of sets $\overline{V}(u)$ has the following properties:

  (P0) $V \in \mathcal{F}$, $\varnothing \notin \mathcal{F}$,
  (P1) no two elements of $\mathcal{F}$ overlap.

An element of a family $\mathcal{F}$ of subsets of a set V will be called an $\mathcal{F}$-*module.* Every family $\mathcal{F}$ satisfying properties P0 and P1 is associated as above with a rooted tree $T_\mathcal{F}$ that is defined as follows. We take $\mathcal{F}$ as set of nodes, V as root, the inverse of inclusion as ancestor relation. The leaves are the minimal $\mathcal{F}$-modules. For a node N, we let $V(N) = N - \bigcup \{ M \in \mathcal{F} \mid M \subset N \}$ and we denote this family of sets by $\mathcal{V}_\mathcal{F}$. We have $\overline{V_\mathcal{F}}(N) = N$. Hence $\mathcal{F}(T_\mathcal{F}, \mathcal{V}_\mathcal{F}) = \mathcal{F}$. We have $V_\mathcal{F}(N) \neq \varnothing$ for every node N of outdegree 1. Every pair $(T, \mathcal{V})$ such that $\mathcal{F}(T, \mathcal{V}) = \mathcal{F}$ and $V_\mathcal{F}(N)$ is nonempty for every node N of outdegree 1 is isomorphic to $(T_\mathcal{F}, \mathcal{V}_\mathcal{F})$ (this means that there exists an isomorphism h of T onto $T_\mathcal{F}$ such that $V_\mathcal{F}(h(u)) = V(u)$ for every node u of T).

If $M, P \in \mathcal{F}$ their least common ancestor in $T_\mathcal{F}$ is the least $\mathcal{F}$-module N containing $M \cup P$. We define a binary relation $box_{T_\mathcal{F}}(x, N)$ holding if and only if x belongs to $V_\mathcal{F}(N)$. We also define a binary relation $mod_{T_\mathcal{F}}(x, N)$ holding if and only if x belongs to $\overline{V_\mathcal{F}}(N)$. This relation is membership since the nodes of $T_\mathcal{F}$ are the $\mathcal{F}$-modules. However, it will be useful later when we construct $T_\mathcal{F}$ as an abstract tree, and not as a set of sets ordered by inclusion. The relations $box_{T_\mathcal{F}}$ and $mod_{T_\mathcal{F}}$ are definable from each other with the help of the son relation of the tree $T_\mathcal{F}$.

If the family $\mathcal{F}$ satisfies the stronger property:

  (P'0) $V \in \mathcal{F}$, $\varnothing \notin \mathcal{F}$, $\{v\} \in \mathcal{F}$ for each $v \in V$,

then the leaves of $T_\mathcal{F}$ are the singletons $\{v\}$, $T_\mathcal{F}$ is a proper tree and $V_\mathcal{F}(u)$ is empty if u is an internal node. If a family $\mathcal{F}$ satisfies only P0 and P1, then the family $\mathcal{F}^+ = \mathcal{F} \cup \{\{v\} \mid v \in V\}$ satisfies P'0 and P1. The corresponding tree $T_{\mathcal{F}^+}$ is obtained from $T_\mathcal{F}$ as follows: for each v such that $\{v\} \notin \mathcal{F}$, we add $\{v\}$ as a new leaf with father the least $\mathcal{F}$-module containing v.

Let $\mathcal{C}$ be a class of relational structures (see Appendix 1 for definitions). For each $S \in \mathcal{C}$, we let $\mathcal{F}(S)$ be a family of subsets of its domain D_S. We say that $\mathcal{F}$ is *MS-definable* if there exists an MS formula $\varphi(X)$ such that for every S in $\mathcal{C}$, $\mathcal{F}(S) = \{A \mid A \subseteq D_S, S \models \varphi(A)\}$. With these definitions:

Proposition 2.2. *Let $\mathcal{C}$ be a set of $\mathcal{R}$-structures and $\mathcal{F}(S)$ be an MS-definable family of subsets of D_S which satisfies P0 and P1 for every $S \in \mathcal{C}$. There exists a domain extending*

MS-transduction that associates with $(S, \preccurlyeq)$, where $S = \langle D_S, (R_S)_{R \in \mathcal{R}} \rangle \in \mathcal{C}$ and D_S is linearly ordered by $\preccurlyeq$, the structure:

$$Dec(S) = \langle D_S \cup N_T, (R_S)_{R \in \mathcal{R}}, son_T, box_T \rangle$$

where $T = T_{\mathcal{F}(S)}$ and $box_T = box_{T_{\mathcal{F}(S)}}$.

We will give a proof of this proposition adapted from that of [9], Section 5. In the structure $Dec(S)$ the domain consists of elements of different natures. If we are given a structure $\langle D_U, (R_U)_{R \in \mathcal{R}}, son_U, box_U \rangle$ intended to be isomorphic to $\langle D_S \cup N_T, (R_S)_{R \in \mathcal{R}}, son_T, box_T \rangle$ we can identify the nodes of T as the elements x of D_U such that $son_U(x, y)$ or $son_U(y, x)$ or $box_T(y, x)$ holds for some y. (We assume D_S nonempty; T may have a single node).

Monadic Second-order logic (MS logic in short) and Monadic Second-order transductions (MS transductions) are defined in Appendix 1. We only recall here that an MS transduction (also called sometimes an MS interpretation) is a transformation of relational structures that is specified by MS formulas forming its definition scheme. It transforms a structure S into a structure T (possibly over a different set of relations) such that the domain D_T of T is a subset of $D_S \times \{1, \ldots, k\}$. The numbers $1, \ldots, k$ are just a convenience for the formal definition; we are actually interested by relational structures up to isomorphism. In many cases, this transformation involves a bijection of D_S onto a subset of D_T, and the definition scheme can be constructed in such a way that this bijection is the mapping: $x \mapsto (x, 1)$. Hence, in this case D_T contains $D_S \times \{1\}$, an isomorphic copy of D_S and we will say that the MS transduction is domain extending, because it defines the domain of T as an extension of that of S. This does not imply that the relations of T extend those of S. An FO transduction is a transduction defined by a first-order definition scheme.

Definition 2.3. The leaves of a tree. Let T be a proper rooted tree. We write $x \leq y$ if x is below y and we denote by $y \vee z$ the least upper bound of two elements y and z. The root is thus the unique maximal element of T for this order. We let $\lambda(T) = \langle Leaves(T), R_T \rangle$ where $Leaves(T)$ denotes the set of leaves of T and $R_T(x, y, z)$ holds if and only if $x \leq y \vee z$. The next lemma shows that if $Leaves(T)$ is linearly ordered by some auxiliary order $\preccurlyeq$, then T is definable from $(\lambda(T), \preccurlyeq)$ by a domain extending MS-transduction. The resulting tree T does not depend on the linear order $\preccurlyeq$.

Lemma 2.4. There exists a domain extending FO transduction that maps $(\lambda(T), \preccurlyeq)$ to T, whenever T is a proper rooted tree and $\preccurlyeq$ is a linear order on $Leaves(T)$.

Proof. Let T be a proper rooted tree and $\preccurlyeq$ be a linear order on its leaves. For every internal node x of T we let:

- $fl(x)$ be the $\preccurlyeq$-smallest leaf below x, called the *first leaf below* x, and we let $fs(x)$ be the unique son y of x such that $fl(x) \leq y$;
- $rep(x)$ be the $\preccurlyeq$-smallest leaf below x and not below $fs(x)$ (this is well-defined because in a proper tree, every internal node has at least two sons).

We call $rep(x)$ the *leaf representing* x. We have $fl(x) < x$, $rep(x) < x$, and $fl(x) \prec rep(x)$.

Claim 1: Let x, y be two internal nodes. If $rep(x) = rep(y)$ then $x = y$.

Proof of the claim. By contradiction. Let x, y be distinct internal nodes such that $u = rep(x) = rep(y)$. Since u is below x and y, x and y are comparable. We can assume that

$x < y$. By the definitions, u is not below $fs(x)$. Hence $fl(x) \prec u$ and $fl(x) \le fs(x) < x$. Since $x < y$, u and $fl(x)$ are below the same son of y, call it z; we may have $x = z$. We have $fl(y) \le fs(y) < y$, where $fs(y) \ne z$ since $u = rep(y)$. Hence, since $u = rep(y)$, u is the $\preccurlyeq$-smallest leaf among the set of leaves below y and not below $fs(y)$, and this set contains $fl(x)$. Hence $u \preccurlyeq fl(x)$, contradicting the above observation that $fl(x) \prec u$. □

One can define a bijection of the nodes of T onto a subset of $\lambda(T) \times \{1, 2\}$. Each leaf u is mapped to $(u, 1)$, hence the transduction we are constructing will be domain extending. Each internal node u is mapped to $(rep(u), 2)$.

Claim 2: One can write a first-order formula $\alpha(x, y, z)$ such that:

$$(\lambda(T), \preccurlyeq) \models \alpha(x, y, z) \qquad \text{if and only if} \qquad x \ne y \quad \text{and} \quad z = rep(x \vee y) \,.$$

Proof of the claim. We recall that $R(u, v, w)$ means: $u \le v \vee w$ for leaves u, v, w. The relation $\le$ denotes the ancestor relation in T and should not be confused with the linear order $\preccurlyeq$ on the set of leaves of T. Using R, one can construct an FO formula $\beta(x, y, z)$ expressing that $x \ne y$ and $z = fl(x \vee y)$. An FO formula $\gamma(x, y, u, v)$ can be constructed to express that:

$$x \ne y, \ u \ne v, \ u \le x \vee y, \ v \le x \vee y, \ x \le u \vee v, \ \text{and } y \le u \vee v \,,$$

which means that for leaves $u, v \ne u, x, y \ne x$, u and v are below $x \vee y$ but are not below the same son of this node. We can construct $\alpha(x, y, z)$ so as to express the following:

$$\exists u [\beta(x, y, u) \wedge "z \text{ is the } \preccurlyeq\text{-smallest element such that } \gamma(x, y, u, z) \text{ holds"}] \,.$$

This completes the proof Claim 2. □

We let $N = (\lambda(T) \times \{1\}) \cup (REP_T \times \{2\})$, where REP_T is the set of leaves of the form $rep(x \vee y)$ for some leaves $x, y \ne x$. We order N by letting:

$(x, 1) \le (y, 1)$ if and only if $x = y$,

$(x, 2) \le (y, 1)$ never holds,

$(x, 1) \le (y, 2)$ if and only if there exist leaves u, v such that

$\qquad y = rep(u \vee v)$ and $R_T(x, u, v)$ holds,

$(x, 2) \le (y, 2)$ if and only if there exist leaves u, v, w, z such that

$\qquad x = rep(u \vee v)$, $y = rep(w \vee z)$, $R_T(u, w, z)$ and $R_T(v, w, z)$ hold.

Claim 3 : The tree $(T, \le)$ is isomorphic to $(N, \le)$ under the bijection which maps a leaf u of T to $(u, 1)$ and an internal node u to $(rep(u), 2)$.

Proof of the claim. The four clauses above correspond to the facts that two different leaves are incomparable, that an internal node cannot be below a leaf, that a leaf x is below an internal node $u \vee v$ if and only if $R_T(x, u, v)$ holds, and that an internal node $u \vee v$ is below $w \vee z$ if and only if u and v are both below $w \vee z$. □

These claims give the desired result because the set REP_T is FO definable in the structure $(\lambda(T), \preccurlyeq)$ by Claim 2. The ancestor relation defined by the formulas before Claim 3 is also FO definable. From it, one can obtain an FO definition of the *son* relation. Hence, we have an FO transduction as claimed. □

Remark. *On the role of $\preccurlyeq$.* The role of the linear order $\preccurlyeq$ is to make possible the construction of a set REP_T so that FO formulas can specify in a unique way the element of REP_T intended to represent an internal node, and to express in terms of this encoding, the *son* relation of T. The tree T is uniquely defined for every structure $\lambda(T)$. *Uniquely* means here that if T and T' are such that $\lambda(T) = \lambda(T')$, there is a unique isomorphism of T onto T' that is the identity on leaves.

Proof of Proposition 2.2. We will use $\mathcal{F}(S)^+$ instead of $\mathcal{F}(S)$. (We have $\mathcal{F}(S)^+ = \mathcal{F}(S)$ if $\mathcal{F}(S)$ satisfies P'0). It is clear that $\mathcal{F}(S)^+$ is MS definable. We construct a structure with domain a subset of $D_S \times \{1, 2, 3\}$. Its domain is the union of three sets:

- the set $D_S \times \{1\}$, a copy of D_S,
- the set $D_S \times \{2\}$ which is the set of leaves of $T = T_{\mathcal{F}(\mathcal{S})^+}$ (the pair $(v, 2)$ represents the leaf $\{v\}$ for each v in D_S),
- and of a subset of $D_S \times \{3\}$, namely $REP_T \times \{3\}$, (cf. the proof of Lemma 2.4) in bijection with the set of internal nodes of T.

The relation $R_T(z, x, y)$ is "$z \in N$ where N is the least set in $\mathcal{F}(S)$ that contains x and y". This is expressible by an MS formula. Hence the structure $\lambda(T)$ is definable from S by an MS-transduction. Since the set of leaves of T is linearly ordered by $\preccurlyeq$ (because D_S is, and is in bijection with D_S) we can obtain T from $(\lambda(T), \preccurlyeq)$ by a domain extending MS-transduction.

Then we reduce $T = T_{\mathcal{F}(\mathcal{S})^+}$ into $T_{\mathcal{F}(\mathcal{S})}$ by eliminating the leaves $(v, 2)$ such that $\{v\} \notin \mathcal{F}(S)$. The relation $box_{T_{\mathcal{F}(S)}}$ is also MS definable since $mod_{T_{\mathcal{F}(S)}}(x, u)$ holds (cf. Definition 2.1) if and only if in the tree $T_{\mathcal{F}(\mathcal{S})^+}$, the singleton $\{x\}$ is a leaf below u. We obtain thus an MS transduction. A definition scheme for it can be written from the above description. □

Remark. One could alternatively define the domain of the constructed structure as a subset of $D_S \times \{1, 2\}$ by letting $(v, 1)$ represent simultaneously the element v of D_S and the leaf $\{v\}$ of $T_{\mathcal{F}(\mathcal{S})}$, in the case where $\{v\} \in \mathcal{F}(S)$. In this case, the internal nodes of $T_{\mathcal{F}(\mathcal{S})}$ are pairs $(v, 2)$. However, in most cases, we will keep separated the domain D_S of the structure and the set of nodes of its decomposition tree, even if there is a "natural" bijection between a subset of D_S and a set of nodes of the tree.

An MS property is *order-invariant* if it is expressible by an MS formula using an auxiliary linear order $\preccurlyeq$ of the domain of the considered structure, that can be chosen arbitrarily. See Appendix 1 for a more precise definition.

Corollary 2.5. *Under the hypotheses of Proposition 2.2, every MS property of the structures $Dec(S)$ for $S \in \mathcal{C}$ is equivalent to an order-invariant MS property of the structures S.*

Proof. Let $\mathcal{P}$ be an MS property of the structures $Dec(S)$. By Proposition A.1.2 (in Appendix 1), applied to the transduction of Proposition 2.2, $\mathcal{P}(Dec(S))$ is equivalent to an MS property $\mathcal{Q}$ of $(S, \preccurlyeq)$, where $\preccurlyeq$ is *any* linear order of D_S. For any two linear orders $\preccurlyeq$ and $\preccurlyeq'$ on D_S, one obtains isomorphic structures $Dec(S)$ by the remark before the proof of Proposition 2.2. Hence $\mathcal{Q}$ is an order-invariant MS property. □

Definition 2.6. *Partitive families of sets* Let V be a nonempty set. A family $\mathcal{F}$ of subsets of V is *weakly partitive* if it satisfies the following properties:

(P0) $V \in \mathcal{F}$, $\varnothing \notin \mathcal{F}$.

(P2) For every two overlapping $\mathcal{F}$-modules A and B we have $A \cup B$, $A \cap B$, $A - B \in \mathcal{F}$.

It is *partitive* if, in addition, it satisfies the following property:

(P3) For every two overlapping $\mathcal{F}$-modules A and B we have $A \Delta B \in \mathcal{F}$, (where $A \Delta B = (A - B) \cup (B - A)$).

The *strong* $\mathcal{F}$-modules are the $\mathcal{F}$-modules which do not overlap any other $\mathcal{F}$-module. The family $Strong(\mathcal{F})$ of strong $\mathcal{F}$-modules satisfies P0 and P1. The corresponding rooted tree $T_{Strong(\mathcal{F})}$ is called the *decomposition tree* of $\mathcal{F}$ or of the structure S, if $\mathcal{F} = \mathcal{F}(S)$ is known from the context. Its leaves are the minimal $\mathcal{F}$-modules (they are strong). They are the singletons $\{v\}$ for all elements v of V if $\mathcal{F}$ satisfies P'0. Since a singleton does not overlap any set, if $\mathcal{F}$ is weakly partitive or partitive, then $\mathcal{F}^+$ is weakly partitive or partitive respectively.

The conditions of partitivity and weak partitivity on a family $\mathcal{F}$ imply some particular structure associated with the nodes of $T_{Strong(\mathcal{F})}$. They are formulated in an easier way in terms of the tree $T_{Strong(\mathcal{F}^+)}$ rather than in terms of $T_{Strong(\mathcal{F})}$. We recall that the nodes of $T_{Strong(\mathcal{F}^+)}$ are subsets of V.

Theorem 2.7. *Let $\mathcal{F}$ be a partitive family.*

(1) Every internal node N of the tree $T_{Strong(\mathcal{F}^+)}$ satisfies one of the following two properties:

> T1: *N has k sons, $N_1, \ldots, N_k, k \geq 2$, and for every nonempty subset I of $\{1, \ldots, k\}$, the set $\bigcup\{N_i \mid i \in I\}$ belongs to $\mathcal{F}$.*
>
> T2: *N has k sons, $N_1, \ldots, N_k, k \geq 2$, and for every subset I of $\{1, \ldots, k\}$, the set $\bigcup\{N_i \mid i \in I\}$ belongs to $\mathcal{F}$ if and only if I is $\{1, \ldots, k\}$ or singleton.*

(2) If an $\mathcal{F}$-module is not strong, it is of the form $\bigcup\{N_i \mid i \in I\}$ for some node N satisfying T1 and a non singleton set $I \subset \{1, \ldots, k\}$.

> *Let $\mathcal{F}$ be weakly partitive.*

(3) Every internal node N of the tree $T_{Strong(\mathcal{F}^+)}$ satisfies one of properties T1, T2 or

> T3: *The sons of N can be numbered $N_1, \ldots, N_k$, $k \geq 2$, in such a way that for every subset I of $\{1, \ldots, k\}$, the set $\bigcup\{N_i \mid i \in I\}$ belongs to $\mathcal{F}$ if and only if I is an interval $[m, n]$ for some m, n with $1 \leq m \leq n \leq k$.*

(4) If an $\mathcal{F}$-module is not strong, it is of the form $\bigcup\{N_i \mid i \in I\}$ for some node N satisfying T1 and a non singleton set $I \subset \{1, \ldots, k\}$, or of the form $\bigcup\{N_i \mid i \in [m, n]\}$ for a node N satisfying T3 and $m < n$. $\square$

See [4, 18, 22, 19, 26] for the proof. The nodes of types T1, T2, T3 are called respectively the *complete nodes*, the *prime nodes* and the *linear nodes*. In this theorem, one could require $k \geq 3$ in conditions T1 and T3 because the nodes with two sons satisfy T2, and then properties T1, T2 and T3 would be mutually exclusive. However, in the application of this theorem to the modular decomposition, properties T1, T2, T3 and the notions of complete, prime or linear nodes correspond to three different graph operations, and those corresponding to T1 and T3 may have two arguments only.

This theorem will be used for classes $\mathcal{C}$ of relational structures, where for each S in $\mathcal{C}$:

(i) we have a partitive or weakly partitive MS definable family $\mathcal{F}(S)$ of subsets of its domain D_S,

(ii) for each node N of the decomposition tree of S, one can express $S[N]$, the substructure of S induced by N, as a composition of the substructures $S[N_1], S[N_2], \ldots, S[N_k]$ by an operation f (such operations can be seen as generalized concatenations) where $N_1, \ldots, N_k$ are the sons of N,

(iii) the nature of this operation f can be determined by an MS formula with free variables which take $N_1, N_2, \ldots, N_k$ as values.

In this case, Proposition 2.2 can be improved, and one can define an MS transduction that takes as input $S = \langle D_S, (R_S)_{R \in \mathcal{R}} \rangle$, together with an arbitrary linear order $\preccurlyeq$ of D_S and produces a structure $Rep(S)$ consisting of the decomposition tree of S augmented with some relations which encode the operations f, and from which S can be reconstructed by an MS transduction. Such a structure contains information on the hierarchical construction of S, and it is, in some cases, a space efficient representation of S. (See the book by Spinrad [29] on efficient graph representations in a very general sense).

Hence our method consists in doing the following steps:

(i) first, we construct from $(S, \preccurlyeq)$ a structure:

$$Dec(S) = \langle D_S \cup N_T, (R_S)_{R \in \mathcal{R}}, son_T, box_T \rangle$$

which includes S *and* the decomposition tree T together with the relation box_T which links both; this structure is independent of $\preccurlyeq$ up to isomorphism;

(ii) second, we construct, *if possible*, a structure $Rep(S)$ with domain $D_S \cup N_T$ and relations son_T, box_T together with some relations encoding the operations f. The objective is here to have a space efficient representation of S, from which S can be reconstructed by an MS transduction.

In some cases, the structure $Rep(S)$ encodes a term over a signature of operations on graphs or, more generally, on relational structures, the value of which is S. If these constructions can be done with MS transductions then Corollary 2.5 applies to $Rep(S)$ in place of $Dec(S)$. At this point it is not appropriate to formalize more this notion in the general setting. We rather show its application in two important examples, the *modular decomposition* based on the family of *nonempty modules of a graph*, and the decomposition in *blocks* of certain directed acyclic graphs called *inheritance graphs*. Some new results are also established.

2.2. **The modular decomposition.** Graphs are simple, directed, loop-free. *Simple* means that there is at most one edge from a vertex x to a vertex y. Graphs are finite, as already indicated. We denote by $x \longrightarrow y$ the existence of an edge from x to y. The undirected graphs are those where each edge $x \longrightarrow y$ has an *opposite edge* $y \longrightarrow x$. We write $x - y$ if $x \longrightarrow y$ and $y \longrightarrow x$. We denote by V_G the set of vertices of a graph G. If X is a set of vertices of G, we denote by $G[X]$ its *induced subgraph* consisting of X and all the edges, the two ends of which are in X. If E is a set of edges, we denote by $G[E]$ its subgraph consisting the edges of E and their end vertices.

Definition 2.8. *Modules and graph substitution.* A *module* of a graph G is a subset M of V_G such that for every vertices x, y in M and every vertex z not in M: $x \longrightarrow z$ implies $y \longrightarrow z$ and $z \longrightarrow x$ implies $z \longrightarrow y$. In words this means that every vertex not in M "sees" all vertices of M in the same way. This frequently rediscovered notion is surveyed in [26]

(see also [20] for numerous references using various names for the same notion). The book by Spinrad [29] contains also many definitions, results, algorithms and references.

We denote by $\mathcal{M}(G)$ the family of nonempty modules of a graph G. It satisfies Property P'0 (each singleton is a module) and is weakly partitive. It is partitive if G is undirected. We denote by $\mathcal{S}(G)$ the corresponding family of *strong modules*: they are the nonempty modules that do not overlap any module. The tree of strong modules is called the *modular decomposition*, and its leaves are the vertices of the considered graph ("are" means that we identify v and $\{v\}$). The relevant operations that combine substructures are *vertex-substitutions*, that we now review.

If G and H are graphs with disjoint sets of vertices, and u is a vertex of G, we denote by $G[H/u]$ the graph such that:
  (a) its set of vertices is $V_G \cup V_H - \{u\}$,
  (b) its edges are those of H, those of G that are not incident with u, the edges $x \longrightarrow y$ whenever $x \in V_G - \{u\}$, $x \longrightarrow u$ in G, $y \in V_H$, and the edges $y \longrightarrow x$ whenever $x \in V_G - \{u\}$, $u \longrightarrow x$ in G, $y \in V_H$.

If G and H are not disjoint, we replace H by an isomorphic copy disjoint with G. When we write: let K be a graph of the form $G[H/u]$ we assume, unless otherwise specified, that G and H are disjoint. This graph is called the result of the *substitution of H for u in G*. It is undirected if G and H are.

If $u_1, \ldots, u_n$ are vertices of G and $H_1, \ldots, H_n$ are graphs, we define $G[H_1/u_1, \ldots, H_n/u_n]$ as $G[H_1/u_1] \ldots [H_n/u_n]$. The order in which substitutions are done is irrelevant, hence we can consider they are done simultaneously.

A graph is *prime* if it has at least 3 vertices and is not of the form $G[H/u]$, except in a trivial way with G or H reduced to a single vertex. The paths $a \longrightarrow b \longrightarrow c$ and $a - b - c - d$ are examples of small prime graphs.

We will also use the graph operations $\oplus, \otimes$ and $\overrightarrow{\otimes}$: $G \oplus H$ is the disjoint union of G and H, $G \overrightarrow{\otimes} H$ is $G \oplus H$ augmented with edges from each vertex of G to each vertex of H, and $G \otimes H$ is $G \overrightarrow{\otimes} H$ augmented with edges from each vertex of H to each vertex of G. In all cases, we replace if necessary H by an isomorphic copy disjoint with G. These operations can be defined by $K[G/u, H/v]$ for graphs K with two vertices u and v, and, respectively, no edge, an edge from u to v, edges between u and v in both directions. They are associative. We will consider them as operations of variable arity in the usual way. The operations $\oplus$ and $\otimes$ are also commutative. They transform undirected graphs into undirected graphs. More generally, every graph G can be turned as follows into a graph operation. We enumerate its vertices as $v_1, \ldots, v_n$, and we define an n-ary graph operation σ_G (where σ stands for *substitution*) by $\sigma_G(H_1, \ldots, H_n) = G[H_1/v_1, \ldots, H_n/v_n]$.

Let us go back to modular decomposition. The complete nodes are of two possible types, $\oplus$ or $\otimes$, because if N is a "complete" strong module with n sons $N_1, \ldots, N_n$, then either $G[N] = G[N_1] \oplus \cdots \oplus G[N_n]$ or $G[N] = G[N_1] \otimes \cdots \otimes G[N_n]$.

If N is "linear" with n sons $N_1, \ldots, N_n$ ordered in this way (cf. T3 in Theorem 2.7), then either $G[N] = G[N_1] \overrightarrow{\otimes} \ldots \overrightarrow{\otimes} G[N_n]$ or $G[N] = G[N_n] \overrightarrow{\otimes} \ldots \overrightarrow{\otimes} G[N_1]$.

If N is "prime", with n sons $N_1, \ldots, N_n$ then $G[N] = \sigma_K(G[N_1], \ldots, G[N_n])$ for some prime graph K. We have $n \geq 3$; the operations corresponding to the graphs K with 2 vertices are $\oplus, \otimes$ and $\overrightarrow{\otimes}$.

The terms *complete, linear* and *prime* are defined after Theorem 2.7. If the given graph is a *dag*, i.e., a directed graph without circuits, then no node is of type $\otimes$. If it is undirected, no node is of type $\overrightarrow{\otimes}$.

An MS formula $\varphi_\oplus(X,Y)$ can express that X is a complete strong module of type $\oplus$ and Y is one of its sons. An MS formula $\varphi_\otimes(X,Y)$ can do the same for $\otimes$. An MS formula $\varphi_{\overrightarrow{\otimes}}(X,Y,Z)$ can express that X is a linear strong module, Y and Z are two sons such that $G[X] = \ldots \overrightarrow{\otimes} G[Y] \overrightarrow{\otimes} G[Z] \overrightarrow{\otimes} \ldots$. An MS formula $\varphi_{\mathrm{Pr}}(X,Y,Z)$ can express that X is a prime strong module, Y and Z are two sons such that $G[X] = K[\ldots, G[Y]/u_i, \ldots, G[Z]/u_j, \ldots]$ where $u_i \longrightarrow u_j$ in the prime graph K.

By using these formulas, one can build the *graph representation of the modular decomposition* of a graph G, denoted by $Gdec(G)$. This is a binary relational structure consisting of the rooted tree $T_{\mathcal{S}(G)}$ enriched with the following informations:

(a) its nodes of types $\oplus, \otimes$ and $\overrightarrow{\otimes}$ are labelled by their respective types,
(b) if N is a "linear" node and $G[N] = G[N_1] \overrightarrow{\otimes} \ldots \overrightarrow{\otimes} G[N_n]$, we set an edge $N_i \longrightarrow N_{i+1}$ for each $i = 1, \ldots, n-1$.
(c) If N is a "prime" strong module, N_i and N_j are two sons and

$$G[N] = K[\ldots, G[N_i]/u_i, \ldots, G[N_j]/u_j, \ldots] \ ,$$

we set an edge $N_i \longrightarrow N_j$ whenever $u_i \longrightarrow u_j$ in K.

This representation is in certain cases space efficient. Consider the graph G of a strict linear order on n elements (in other words, a *transitive tournament*). This graph has n vertices and $n(n-1)/2$ edges. The graph $Gdec(G)$ has $n+1$ vertices and $2n-1$ edges.

The tree $T_{\mathcal{S}(G)}$ and the structure $Gdec(G)$ can be constructed by MS transductions using an arbitrary linear order of the vertices of the given graph as auxiliary information. From the relational structure $Gdec(G)$, that is actually a vertex- and edge-labelled directed graph, one can reconstruct G by an MS transduction. We refer the reader to [9] for illustrated examples and further developments, and to [13] for the extension of these constructions to countable graphs.

There are two distinct extensions of modular decomposition to hypergraphs: the decomposition into *committees* of undirected unranked hypergraphs (see [4, 18]) and the modular decomposition of *k-structures* which are *k*-ary relational structures, hence are labelled directed hypergraphs of rank k (see [20]). In both cases the families of sets are MS definable and Proposition 2.2 is applicable.

2.3. **Factors in directed acyclic graphs.** We review some results of Courcelle [11], Capelle [3], Habib et al. [23] concerning directed acyclic graphs. We show that they can be reformulated in the framework of this section and slightly improved.

Definition 2.9. *2-graphs and 2-dags* In this subsection, we consider directed graphs, possibly with *multiple edges* (hence, not necessarily simple, as in the previous section). Those without circuits (and loops) are called *dags* (for *directed acyclic graphs*). We denote by E_G the set of edges of a graph G. A *2-graph* is a graph with two distinct distinguished vertices denoted by $s_1(G)$ and $s_2(G)$ called its *sources*. We denote by G^0 the underlying graph, i.e., the same graph without distinguished vertices (the sources are turned into "ordinary" vertices). We use "2-graph" as an abreviation of "graph with 2 sources" also called sometimes "2-terminal graph"; 2-graphs *are not* particular C-graphs in the sense of the definition of clique-width, recalled in Appendix 2.

A *2-dag* is a 2-graph without circuits such that $s_1(G)$ is the unique vertex of indegree 0, $s_2(G)$ is the unique vertex of outdegree 0 and every vertex is on a directed path from $s_1(G)$ to $s_2(G)$. We denote by V_G^0 the set $V_G - \{s_1(G), s_2(G)\}$, called the set of *internal vertices* of G. An orientation of a graph making it into a 2-dag is also called a *bipolar orientation*.

For example, the graph of the "Wheatstone bridge" consisting of the directed path $a \longrightarrow b \longrightarrow c \longrightarrow d$ with additional edges $a \longrightarrow c$ and $b \longrightarrow d$ is a 2-dag if its two sources are a and d (in this order) and is a 2-graph and a dag but is not a 2-dag if its two sources are a and b.

A *factor* of a 2-dag G is a 2-dag H such that H^0 is a subgraph of G^0 and if an edge of G has one end in V_H^0, then it is in H. An edge is a factor (its two ends being the sources) and a 2-dag is one of its own factors. We let $\mathcal{FE}(G)$ denote the set of edge sets of the factors of G.

The following proposition is proved in [11], Lemma 3.5 and Corollary 3.6. Its first assertion is also proved in [3] and [23].

Proposition 2.10. *For every 2-dag G, the family $\mathcal{FE}(G)$ is weakly partitive and MS definable.* ∎

Note that $\mathcal{FE}(G)$ satisfies Property P'0. The family of strong $\mathcal{FE}(G)$-modules is denoted by $\mathcal{SFE}(G)$. In order to define the tree $T_{\mathcal{SFE}(G)}$ of a 2-dag G by an MS transduction we need to quantify over edge sets. Hence, we represent graphs by their *incidence structures*. For a graph G, we let $Inc(G) = \langle V_G \cup E_G, inc_G \rangle$ where V_G is the set of vertices, E_G is the set of edges and inc_G is the ternary relation such that $inc_G(e, x, y)$ holds if and only if $e : x \longrightarrow y$ in G. By the general definitions, the binary relation $mod_{\mathcal{SFE}(G)}$ defines for every node x of $T_{\mathcal{SFE}(G)}$ the set of edges of the corresponding factor, that we will denote by $G(x)$; hence $G(x) = G[N]$ if N is the $\mathcal{FE}(G)$-module represented by the node x. The following proposition is Theorem 3.12 of [11].

Proposition 2.11. *There exists an MS transduction that transforms $Inc(G)$ into $Dec(G) = \langle V_G \cup E_G \cup N_{T_{\mathcal{SFE}(G)}}, inc_G, son_{T_{\mathcal{SFE}(G)}}, mod_{\mathcal{SFE}(G)} \rangle$ for every 2-dag G.* ∎

Although the leaves of $T_{\mathcal{SFE}(G)}$ are (or correspond to) the edges of G, we keep $N_{T_{\mathcal{SFE}(G)}}$ and E_G disjoint in the structure $Dec(G)$. Proposition 2.2 could be used here because the family $\mathcal{SFE}(G)$ is MS definable (since graphs are represented by their incidence structures), but it would give a weaker result than Proposition 2.11, because we would need an auxiliary linear ordering of the edges of G as input of the transduction, which is not the case in Proposition 2.11.

The MS transduction of Proposition 2.11 uses edge set quantifications. In the case of simple graphs, one can do the same *without edge set quantifications*.

Corollary 2.12. *There exists a monadic second-order transduction that transforms the structure $\langle V_G, edg_G \rangle$ into:*

$$\langle V_G \cup N_{T_{\mathcal{SFE}(G)}}, edg_G, son_{T_{\mathcal{SFE}(G)}}, fact_{\mathcal{SFE}(G)} \rangle$$

for every simple 2-dag G, where $fact_{\mathcal{SFE}(G)}(v, x)$ is defined to hold if and only if v is a vertex of $G(x)$.

Proof. The proof of Theorem 3.12 in [11] defines $\langle V_G \cup E_G \cup N_T, inc_G, son_T, mod_T \rangle$ from $\langle V_G \cup E_G, inc_G \rangle$ by an MS transduction that specifies the set of nodes of $T = T_{\mathcal{SFE}(G)}$ as follows: Its internal nodes are pairs (v, i) where $i = 2$ or 3 and v is a vertex, or pairs $(e, 3)$

where e is an edge. The latter case corresponds to factors which are sets of at least two parallel edges. In the case of simple graphs, there are no such factors, hence these pairs are not needed. A leaf of T corresponds to a factor of G reduced to a single edge e and is defined as the pair $(e, 4)$. However, from the above remark, its father is an internal node defined as a pair (v, i) where $i = 2$ or 3 for a vertex v. Hence, this leaf can be defined as the pair $(v, 4)$ in the former case and $(v, 5)$ in the latter. It follows that T can be specified with a set of nodes defined as a subset of $V_G \times \{2, 3, 4, 5\}$. □

We now review the graph operations associated with the various types of nodes of $T_{\mathcal{SFE}(G)}$ where G is a 2-dag. We define them actually for 2-graphs.

Definition 2.13. *Operations on 2-graphs* The main operation is *edge-substitution*. Two other operations will be defined as particular instances of it. For a 2-graph K with directed edges $e_1, \ldots, e_k$, we denote by $K[G_1/e_1, \ldots, G_k/e_k]$ the result H of the substitution of the 2-graphs $G_1, \ldots, G_k$ for the edges $e_1, \ldots, e_k$ respectively. For defining H, we assume what follows:

(i) $K, G_1, \ldots, G_k$ have pairwise disjoint sets of edges,
(ii) $e_i : s_1(G_i) \longrightarrow s_2(G_i)$ for each i,
(iii) $K, G_1, \ldots, G_k$ have no vertices in common other than the ends of the edges e_i and the sources of G_i, as required by (ii).

We let $V_H = V_K \cup V_{G_1} \cup \cdots \cup V_{G_k}$, $E_H = E_K \cup E_{G_1} \cup \cdots \cup E_{G_k} - \{e_1, \ldots, e_k\}$ and $s_i(H) = s_i(K)$ for $i = 1, 2$. If the graphs are not disjoint as required, one takes disjoint copies and one fuses the sources of the graphs G_i with the end vertices of the edges e_i of K. In this case, the result of the substitution is well-defined up to isomorphism.

The *parallel composition* of two 2-graphs G and H is the graph $G//H$ defined as $K[G/e, H/f]$ where K consists of two parallel edges, $e, f \colon s_1(K) \longrightarrow s_2(K)$. This operation is associative and commutative so that the expression $G_1//\ldots//G_k$ is well-defined, and the ordering of the arguments is irrelevant. We define similarly the *series composition*: $G \bullet H = K[G/e, H/f]$ where K consists of two edges $e : s_1(K) \longrightarrow u$ and $f : u \longrightarrow s_2(K)$ for some (arbitrary) u. This operation is associative, so that the expression $G_1 \bullet \cdots \bullet G_k$ is well-defined, but the order of arguments matters. In order to have a shorter notation we will use $\theta_K(G_1, \ldots, G_k)$ for $K[G_1/e_1, \ldots, G_k/e_k]$ where $e_1, \ldots, e_k$ is an enumeration of the set of edges of K (it is not made explicit in the notation θ_K).

We will also use the constant **e** denoting the 2-dag consisting of the single edge $e : x \longrightarrow y$, with $s_1(\mathbf{e}) = x, s_2(\mathbf{e}) = y$, and the constant $\overline{\mathbf{e}}$ defined similarly, with $s_1(\overline{\mathbf{e}}) = y, s_2(\overline{\mathbf{e}}) = x$.

Terms built with these operations and constants denote 2-graphs. In some proofs, we will require that the arguments of each operation in a term are graphs with disjoint sets of edges (hence not graphs up to isomorphism). In this case, if t is a term denoting a graph with edges $e_1, \ldots, e_k$, it has k occurrences of constants, which are $\mathbf{e}_i$ or $\overline{\mathbf{e}_i}$ for $i = 1, \ldots, k$.

We now consider the case of 2-dags.

Proposition 2.14. *Let G be a 2-dag. An internal node N of its decomposition tree $T_{\mathcal{SFE}(G)}$ is of one of the following mutually exclusive types:*

1) *N is a complete node with sons $N_1, \ldots, N_k$, $N = N_1 \cup \cdots \cup N_k$, we have $G[N] = G[N_1]//\ldots//G[N_k]$ and none of $N_1, \ldots, N_k$ is a complete node.*
2) *N is a prime node, $G[N] = \theta_K(G[N_1], \ldots, G[N_k])$ where K is a 2-dag that cannot be written $L \bullet M$ or $L//M$ or $K'[M_1/f_1, \ldots, M_p/f_p]$ except in a trivial way, with either K' or all $M_1, \ldots, M_p$ reduced to single edges.*

3) N *is a linear node with sons* $N_1, \ldots, N_k$, *we have* $N = N_1 \cup \cdots \cup N_k$, *none of* $N_1, \ldots, N_k$ *is linear and, either* $G[N] = G[N_1] \bullet \cdots \bullet G[N_k]$ *or* $G[N] = G[N_k] \bullet \cdots \bullet G[N_1]$.

A leaf of this tree is an edge $e : s_1(\mathbf{e}) \longrightarrow s_2(\mathbf{e})$.

Proof. This follows from Proposition 2.10 and Theorem 2.7. The three types of nodes correspond respectively to properties T1, T2 and T3 of Theorem 2.7. In all three cases, the graphs $G[N_1], \ldots, G[N_k]$ have disjoint sets of edges and we need not make isomorphic copies. In the second case, the vertices of K are vertices of G. Its edges are not edges of G: they mark positions where the subgraphs $G[N_1], \ldots, G[N_k]$ must be substituted. The graph K is simple because otherwise it can expressed as $\mathbf{e}//M$ or as $K'[M_1/f_1, \ldots, M_p/f_p]$ in a nontrivial way with some M_i consisting of two parallel edges. □

In Case 2, K has $k \geq 3$ edges because otherwise, it is of the form $\mathbf{e} \bullet \mathbf{f}$ or $\mathbf{e}//\mathbf{f}$, and θ_K is $\bullet$ or $//$. A 2-dag K satisfying the conditions of Case 2 will be called *prime*. In Case 3, the sons of a node will be numbered so that $G[N] = G[N_1] \bullet \cdots \bullet G[N_k]$. For building a term t denoting a 2-dag, we need only the operations $\bullet$, $//$ and θ_K where K is a prime 2-dag, and the constants $\mathbf{e}$.

We now examine how such a term can be constructed in MS logic. MS formulas analogous to the formulas $\varphi_\oplus$, $\varphi_\otimes$, $\varphi_{\overrightarrow{\otimes}}$, φ_{Pr} used in Subsection 2.2 for the modular decomposition, can recognize which case applies to a given module N, and can specify its sons. According to the general method sketched at the end of Subsection 2.1, we transform the structure constructed by the transduction of Proposition 2.11 into a *graph representation* of the canonical decomposition of the considered 2-dag, intended to be as space-efficient as possible. We let $Rep(G)$ be the structure:

$$\langle V_G \cup E_G \cup N_{T_{\mathcal{SFE}(G)}}, inc_G, son_{T_{\mathcal{SFE}(G)}}, src_{1\mathcal{SFE}(G)}, src_{2\mathcal{SFE}(G)}, leaf_{T_{\mathcal{SFE}(G)}} \rangle,$$

where:

(1) $src_{i\mathcal{SFE}(G)} = \{(x, s_i(G(x))) \mid x \in N_{T_{\mathcal{SFE}(G)}}\}$ and

(2) $leaf_{T_{\mathcal{SFE}(G)}} = \{(x, e) \mid x \text{ is a leaf of } T_{\mathcal{SFE}(G)} \text{ and } e \text{ is the unique edge of } G(x)\}$.

We replace thus the relation $mod_{\mathcal{SFE}(G)}$ of $Dec(G)$ by three functional relations, and we avoid a certain amount of redundancy. The relation $mod_{\mathcal{SFE}(G)}$ can be defined by an MS formula in the structure $Rep(G)$. The relation $leaf_{T_{\mathcal{SFE}(G)}}$ is useful to establish the bijection between the leaves of the tree and the edges of the considered graph. For simple graphs, we can use the simpler structure:

$$Rep'(G) = \langle V_G \cup N_{T_{\mathcal{SFE}(G)}}, edg_G, son_{T_{\mathcal{SFE}(G)}}, src_{1\mathcal{SFE}(G)}, src_{2\mathcal{SFE}(G)} \rangle$$

because there is no need to relate a leaf of the tree to the corresponding edge which no longer exists as an element of the domain.

Definition 2.15. *Separated representations.* As explained above, to every internal node of the tree $T_{\mathcal{SFE}(G)}$ corresponds an edge substitution operation θ_K, and this tree can be considered as the syntax tree of a term t that denotes the 2-dag G and is written with operations θ_K and constants $\mathbf{e}$ denoting the different edges. In a proof in the next subsection, we will transform such a term t denoting a 2-dag into another one t', intended to denote a 2-graph G', by replacing at certain occurrences in t, some operations θ_K by operations $\theta_{K'}$ of same arity. Then the evaluation of t' giving G' will be done by an MS transduction. For this

purpose, we introduce a variant of the structure $Rep(G)$, called a *separated representation*, where G is a 2-dag.

We let

$Rep^{sep}(G) =$

$\langle V_H \cup E_G \cup N_{T_{S\mathcal{F}\mathcal{E}(G)}}, inc_H, \varepsilon - edg_H, son_{T_{S\mathcal{F}\mathcal{E}(G)}}, ssrc_{1S\mathcal{F}\mathcal{E}(G)}, ssrc_{2S\mathcal{F}\mathcal{E}(G)}, leaf_{T_{S\mathcal{F}\mathcal{E}(G)}}\rangle,$

where:

(1) V_H is the set of pairs (x, i) for $x \in N_{T_{S\mathcal{F}\mathcal{E}(G)}}$ and $i = 1, 2$,

(2) inc_H is the set of triples $(e, (x, 1), (x, 2))$ such that $e \in E_G$, x is the corresponding leaf of $T_{S\mathcal{F}\mathcal{E}(G)}$,

(3) $\varepsilon - edg_H((x, i), (y, j))$ is defined as holding if and only if $s_i(G(x)) = s_j(G(y))$ and, either x and y are adjacent (one is the father of the other) or x and y are sons of some node z, and $s_i(G(x)) \neq s_k(G(z))$ for $k = 1, 2$.

(4) $ssrc_{iS\mathcal{F}\mathcal{E}(G)} = \{(x, (x, i)) \mid x \in N_{T_{S\mathcal{F}\mathcal{E}(G)}}\}$,

(5) the other sets and relations are as in $Rep(G)$.

These sets and relations define a graph H. Its vertices are pairs $(x, 1)$ and $(x, 2)$ denoting the two sources of the factors associated with the nodes x of the tree. Each pair represents a vertex of G. Since a vertex of G belongs to several factors, it has several representations by vertices of H. For an example, if z is a node with sons x and y such that $G(z) = G(x) \bullet G(y)$, then $s_1(G(z)) = s_1(G(x))$, $s_2(G(z)) = s_2(G(y))$, $s_2(G(x)) = s_1(G(y))$. The undirected ε-*edges*, defined by the symmetric relation $\varepsilon - edg_H$ materialize such equalities. In this case, we have the following ε-edges: $(z, 1) - (x, 1)$, $(z, 2) - (y, 2)$ and $(x, 2) - (y, 1)$. In the case where $G(z) = G(x)//G(y)$, we have the ε-edges: $(z, i) - (x, i)$, $(z, i) - (y, i)$, for $i = 1, 2$, which represent the equalities $s_i(G(z)) = s_i(G(x))$, $s_i(G(z)) = s_i(G(y))$, and the equalities $s_i(G(x)) = s_i(G(y))$ follow by transitivity. The graph H has also edges which correspond to those of G, however, they are not adjacent in H. They are "separated" by ε-edges. Since t is a term denoting a 2-dag, its leaves correspond to factors with an edge directed from the first source to the second one. This justifies condition (2).

This graph H, denoted by $Sep(G)$, can be "extracted" from $Rep^{sep}(G)$ which contains also $T_{S\mathcal{F}\mathcal{E}(G)}$ as it is defined from $\langle V_H \cup E_G, inc_H, \varepsilon - edg_H\rangle$. Figure 1 below shows a graph G, and Figure 2 shows the corresponding graph $Sep(G)$. Edge directions are omitted for the purpose of readability. Dotted lines represent the pairs in ε-edg_H. The following fact, which shows how one can reconstruct G from $Sep(G)$, is clear from the definition.

Lemma 2.16. *The graph G is obtained from $Sep(G)$ by the contraction of all ε-edges. There exist MS transductions transforming $Rep(G)$ and $Rep^{sep}(G)$ into each other, and $Sep(G)$ into $Inc(G)$.* □

Remark. The structures $Rep(G)$ and $Rep^{sep}(G)$ use fixed finite signatures. They encode terms written with the operations $//$ and $\bullet$ of variable arity, and the infinitely many operations associated with the prime graphs K. They are interesting from the point of view of the study of graph structure, but they are not space efficient as can be the graph representations of modular decompositions.

We now give an application related to matroids.

2.4. **Whitney's 2-isomorphism theorem.** We consider directed graphs without loops or isolated vertices and possibly with multiple edges. The *cycle matroid* of a graph G is the pair $M(G) = \langle E_G, indep_G \rangle$ where, for $X \subseteq E_G$, $indep_G(X)$ holds if and only if $G[X]$ has no undirected cycle. The matroid $M(G)$ does not depend on the directions of edges, however, the definitions are given for directed graphs because edge directions will be useful for some constructions. For matroids in general we refer the reader to the books by White [32] and Oxley [28]. Actually, we will need no more than this definition.

We say that two graphs G and H are *equivalent* if $E_G = E_H$ and $M(G) = M(H)$. We require the equality of the sets of edges but nothing on the sets of vertices. The vertices and the incidencies may be different in the two graphs. In particular, any two forests with the same sets of edges have the same (trivial) cycle matroids, independently of how their edges are incident with vertices. A theorem by Whitney characterizes the equivalence of 2-connected graphs.

Definition 2.17. *Twisting* . For a 2-graph G, we let $\widetilde{G}$ be the 2-graph with same underlying graph as G except that its sources are swapped: $s_1(\widetilde{G}) = s_2(G)$ and $s_2(\widetilde{G}) = s_1(G)$. We recall that G^0 is G with its sources made into ordinary vertices. For disjoint 2-graphs G and H, we let $G//H$ be their parallel composition, obtained from the union of G and H by the fusion of $s_1(G)$ and $s_1(H)$, and of $s_2(G)$ and $s_2(H)$. It is important to note here that $E_{G//H} = E_G \cup E_H$.

A graph is *2-connected* if it is connected and the deletion of any vertex yields a connected graph. A graph with just one edge or several parallel edges is 2-connected, and we consider graphs without loops. A 2-graph G is *2-connected* if $(\mathbf{e}//G)^0$ is 2-connected. Edge directions do not matter in these definitions. A 2-dag is a 2-connected 2-graph.

If $G = (L//M)^0$ and $H = (L//\widetilde{M})^0$ where L and M are connected 2-graphs, we say that H is obtained from G by a *twisting*. Note that $E_H = E_G$. Reversing an edge direction is a twisting.

The equivalence of two graphs G and H without isolated vertices is characterized by Whitney's *2-isomorphism Theorem* as the existence of a transformation of G into H by a finite sequence of twistings and of transformations of two other types called *vertex splitting* and *vertex identification*. See the chapter by J. Oxley in the book edited by N. White [32], or [30]. However, these latter transformations do not apply to 2-connected graphs. Hence, this theorem yields the following:

Proposition 2.18. *Two 2-connected graphs are equivalent if and only if one can be transformed into the other by a finite sequence of twistings.* $\qquad\square$

Figure 1 shows two graphs which are 2-isomorphic. Our aim is to prove the following theorem:

Theorem 2.19. *There exists an MS transduction that associates with $\langle V_G \cup E_G, inc_G, \preccurlyeq \rangle$ where G is a 2-connected graph and $\preccurlyeq$ ranges over all linear orders on V_G, the set of graphs having the same cycle matroid as G.*

Lemma 2.20. *A 2-connected 2-graph G is either:*

   (o) $\mathbf{e}$ *or* $\overline{\mathbf{e}}$ *or*
   (i) $G_1//\ldots//G_k$ *for $k \geq 2$, and some 2-connected 2-graphs $G_1, \ldots, G_k$ not of this form, or*
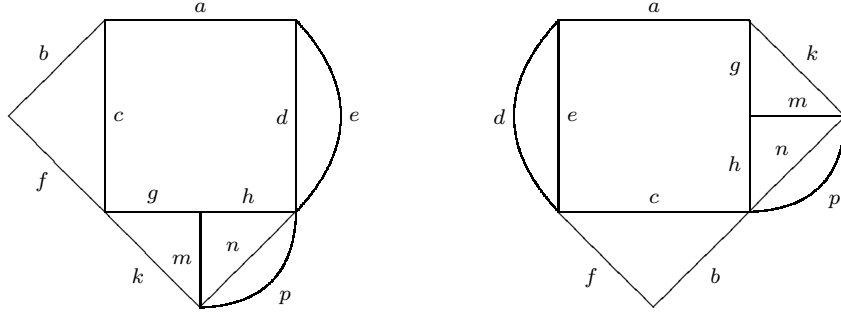
Figure 1: Two 2-isomorphic graphs G (to the left) and H.

(ii) $G_1 \bullet \cdots \bullet G_k$ for $k \geq 2$ and some 2-connected 2-graphs $G_1, \ldots, G_k$ not of this form, or

(iii) $\theta_K(G_1, \ldots, G_k)$ where K is a prime 2-dag and $G_1, \ldots, G_k$ are 2-connected 2-graphs.

Prime 2-dags are defined after Proposition 2.14. These expressions are unique except for the directions of the edges of K and the ordering of $G_1, \ldots, G_k$ in (i), but we will not need this fact.

Proof. For a 2-dag G, this result is Proposition 2.14. Otherwise G can be made into a 2-dag H by reversing some edge directions. The result holds for H, whence also for G by reversing again the same edges. This corresponds to changing certain constants $\mathbf{e}$ into $\overline{\mathbf{e}}$. ◻

It follows that every 2-connected 2-graph G can be expressed as the value of a term t belonging to the set $\mathcal{T}$ of finite terms defined recursively as follows:

either $t = \mathbf{e}$,
  or $t = \overline{\mathbf{e}}$,
  or $t = //(t_1, \ldots, t_k)$,
  or $t = \bullet(t_1, \ldots, t_k)$,
  or $t = \theta_K(t_1, \ldots, t_k)$,

where $t_1, \ldots, t_k$ are in $\mathcal{T}$.

We recall that we denote by θ_K the substitution operation associated with K and the list $e_1, \ldots, e_k$ of its edges: $\theta_K(G_1, \ldots, G_k) = K[G_1/e_1, \ldots, G_k/e_k]$. (The list $e_1, \ldots, e_k$ is implicit in the notation θ_K). The operations $//$ and $\bullet$ have a variable arity. A term in $\mathcal{T}$ obtained by using recursively Lemma 2.20 will be called a *canonical term for* G. It is unique up to the ordering of the arguments of the operations $//$ and up to the directions of edges in K. If the set of edges of G is $\{f_1, \ldots, f_n\}$, then the constants occurring in a canonical term for G are $\mathbf{f}_1$ or $\overline{\mathbf{f}_1}, \ldots, \mathbf{f}_n$ or $\overline{\mathbf{f}_n}$.

Definition 2.21. *Twistings of 2-graphs.* Twisting for graphs is defined above. In order to characterize the twistings of 2-connected graphs in terms of their decompositions in 2-graphs, we extend the notion of twisting to 2-graphs. A *twisting of a 2-graph* G is a 2-graph H such that either $H = \widetilde{G}$, or H^0 is a twisting of G^0, $s_i(H) = s_i(G)$ for $i = 1, 2$, and $G^0 = (L//M)^0$, $H^0 = (L//\widetilde{M})^0$ in such a way that the two sources of G are two vertices of L, of M, or of both (we may have $G = L//M$). These conditions imply that for every 2-graph K, $K//H$ is a twisting of $K//G$. They also imply that if G is a 2-connected 2-graph, then so are H and, L and M when the second case of the definition is used.

For every 2-graph G, we denote by $\triangledown(G)$ the least set of 2-graphs containing G and closed under twisting, hence of 2-graphs obtained from G by a finite sequence of twistings. For every graph H in $\triangledown(G)$, $E_H = E_G$, and either $s_i(H) = s_i(G)$ for $i = 1, 2$, or $s_i(H) = s_{3-i}(G)$ for $i = 1, 2$.

Lemma 2.22. *Let G be a 2-connected 2-graph and H be a twisting of G.*

   (i) *If $G = G_1 // \ldots // G_k$ for $k \geq 2$, and some 2-connected 2-graphs $G_1, \ldots, G_k$ not of this form, then $H = G_1 // \ldots // G_{i-1} // H_i // G_{i+1} \ldots // G_k$ where H_i is a twisting of G_i, or $H = L_1 // \ldots // L_k$ where each L_i is either G_i or $\widetilde{G_i}$,*

   (ii) *if $G = G_1 \bullet \cdots \bullet G_k$ for $k \geq 2$ and some 2-connected 2-graphs $G_1, \ldots, G_k$ not of this form, then $H = G_1 \bullet \cdots \bullet G_{i-1} \bullet H_i \bullet G_{i+1} \bullet \cdots \bullet G_k$ where H_i is a twisting of G_i, or $H = G_1 \bullet \cdots \bullet G_{i-1} \bullet \widetilde{G_j} \bullet \widetilde{G_{j-1}} \bullet \cdots \bullet \widetilde{G_{i+1}} \bullet \widetilde{G_i} \bullet G_{j+1} \bullet \cdots \bullet G_k$ for $1 \leq i < j \leq k$,*

   (iii) *$G = K[G_1/e_1, \ldots, G_k/e_k]$ where K is a prime 2-dag and $G_1, \ldots, G_k$ are 2-connected 2-graphs, then $H = K[G_1/e_1, \ldots, H_i/e_i, \ldots, G_k/e_k]$ where H_i is a twisting of G_i, or $H = \widetilde{K}[G_1/e_1, \ldots, G_k/e_k] = \widetilde{G}$.*

Conversely, in all cases, every graph H of the above forms is either G or a twisting of G.

Note the special cases of

   (i) $H = \widetilde{G} = \widetilde{G_1} // \ldots // \widetilde{G_k}$, and
   (ii) $H = \widetilde{G} = \widetilde{G_k} \bullet \widetilde{G_{k-1}} \bullet \cdots \bullet \widetilde{G_2} \bullet \widetilde{G_1}$.

Proof. Let G be defined from $G_1, \ldots, G_k$ by one of cases (i)-(iii) and H be a twisting of G. If $H = \widetilde{G}$, then the conclusions hold in all three cases. Let us now assume that $G^0 = (L // M)^0$ and $H^0 = (L // \widetilde{M})^0$ and, without loss of generality, that the two sources of G are vertices of L.

We will prove that we have one of the following three cases:

   (a) M^0 is a subgraph of some G_i in any of cases (i)-(iii), then the replacement of M by $\widetilde{M}$ yields a 2-graph H_i, and by replacing G_i by H_i, we obtain H from G, as required.

   (b) G satisfies case (i) and the two sources of M are those of G: then M is the parallel composition of some of the G_i's, and we obtain H from G by replacing each of these G_i's by $\widetilde{G_i}$, this is the second possibility of case (i).

   (c) G satisfies case (ii) and one source of M is a source of some factor G_i. Then the other one is also a source of some factor G_j (with $i \neq j$, otherwise case (a) applies). Then $M = G_{i'} \bullet \cdots \bullet G_{j'}$ for some $1 \leq i' < j' \leq k$, and H is defined by the second possibility of case (iii).

To complete the proof, we need only verify that there are no other cases.

As in the proof of Lemma 2.20, we make G into a 2-dag G' by reversing if necessary some edge directions. We denote by G'_i, L' and M' the 2-graphs obtained from G_i, L and M by these reversals. An internal vertex of M' is on a directed path from $s_1(G')$ to $s_2(G')$. This path goes through the two sources of M'. By changing if necessary the source numbers of M' we may assume that this path traverses M' from $s_1(M')$ to $s_2(M')$. All paths associated in this way with the internal vertices of M' do the same. They must traverse M' from $s_1(M')$ to $s_2(M')$ otherwise M' whence G' has a circuit. Hence M' is a 2-dag and a factor of G'. Clearly, the G_i's are also factors of G'. Consider its decomposition tree $T_{\mathcal{SFE}}(G')$: the G_i's are the sons of its root. We apply Theorem 2.7(4) to M': if it is a strong module (with respect to $\mathcal{SFE}(G')$), it corresponds to a node of this tree, and thus is a factor of (possibly equal to) some G_i. If it is not strong it is a union of sons of a

strong module N, satisfying T1 or T3. If N is the root, we are in the above cases (b) or (c) for $M', G'_1, \ldots, G'_k$ instead of $M, G_1, \ldots, G_k$. Otherwise, M' is a factor of some G'_i. By resestablishing the original edge directions, we see that M satisfies one of (a), (b), (c) as required.

This completes the proof of the "only if" direction. The other one is easy to verify. $\square$

A k-*permutation* is a permutation of $\{1, \ldots, k\}$. For every k-permutation π, we denote by $\bullet_\pi$ the operation of arity k such that:

$$\bullet_\pi(G_1, \ldots, G_k) = \bullet(G_{\pi(1)}, \ldots, G_{\pi(k)}).$$

For every canonical term t, we denote by $\triangledown(t)$ the set of terms defined inductively as follows:

$$\triangledown(\mathbf{e}) = \triangledown(\overline{\mathbf{e}}) = \{\mathbf{e}, \overline{\mathbf{e}}\}$$

$$\triangledown(//(t_1, \ldots, t_k)) = \{//(s_1, \ldots, s_k) \mid s_i \in \triangledown(t_i)\}$$

$$\triangledown(\bullet(t_1, \ldots, t_k)) = \{\bullet_\pi(s_1, \ldots, s_k) \mid s_i \in \triangledown(t_i), \pi \text{ is a } k\text{-permutation}\,\}$$

$$\triangledown(\theta_K(t_1, \ldots, t_k)) = \{\theta_K(s_1, \ldots, s_k) \mid s_i \in \triangledown(t_i)\} \cup \{\theta_{\widetilde{K}}(s_1, \ldots, s_k) \mid s_i \in \triangledown(t_i)\}.$$

Lemma 2.23. *For every 2-connected 2-graph G with canonical term t, the set of 2-graphs $\triangledown(G)$ is the set of values of the terms in $\triangledown(t)$.*

Proof. For every 2-connected 2-graph G we have in the four cases of Lemma 2.20

(o) $\triangledown(\mathbf{e}) = \triangledown(\overline{\mathbf{e}}) = \{\mathbf{e}, \overline{\mathbf{e}}\}$,

(i) $\triangledown(G_1// \ldots //G_k) = \triangledown(G_1)// \ldots //\triangledown(G_k)$,

(ii) $\triangledown(G_1 \bullet \cdots \bullet G_k) = \bigcup\{\triangledown(G_{\pi(1)}) \bullet \cdots \bullet \triangledown(G_{\pi(k)}) \mid \pi \text{ is a } k\text{-permutation}\,\}$,

(iii) $\triangledown(K[G_1/e_1, ..., G_k/e_k]) = K[\triangledown(G_1)/e_1, ..., \triangledown(G_k)/e_k] \cup \widetilde{K}[\triangledown(G_1)/e_1, ..., \triangledown(G_k)/e_k]$,

where in all cases the operations on 2-graphs extend to sets of 2-graphs in the natural way.

The result for cases (o),(i),(iii) follows from Lemmas 2.20 and 2.22. For case (ii), the inclusion $\subseteq$ follows from Lemma 2.22 (ii), and the inclusion $\supseteq$ follows also from the facts that every permutation is a composition of transpositions and that

$$G_1 \bullet \cdots \bullet G_{i-1} \bullet G_{i+1} \bullet G_i \bullet \cdots \bullet G_k$$
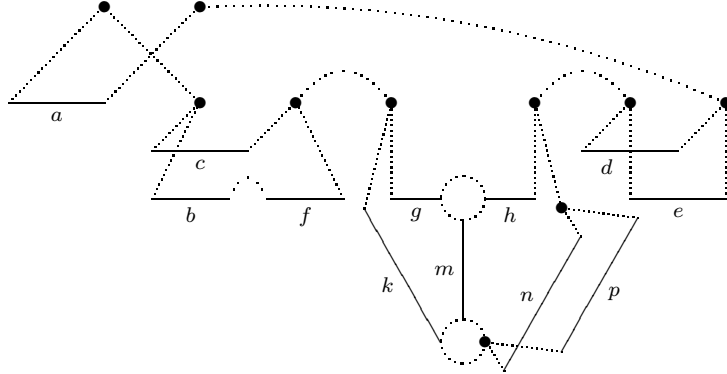
is a twisting of

$$G_1 \bullet \cdots \bullet G_{i-1} \bullet \widetilde{G_i} \bullet \widetilde{G_{i+1}} \bullet G_{i+1} \cdots \bullet G_k \,. \qquad \square$$

The idea of the proof of Theorem 2.19 is illustrated by Figures 1 to 3. By reversing some edge directions if necessary, we make the given graph into a 2-dag G with its two sources the ends of some edge. From the decomposition tree $T_{\mathcal{SFE}(G)}$ we construct the canonical term t of G and the graph $Sep(G)$ from which G is obtained by contraction of the ε-edges. The ε-edges of $Sep(G)$ represent the graph operations with which t is built. In order to produce a term t' in $\triangledown(t)$ yielding G' equivalent to G, it suffices to modify some operations in t according to Lemma 2.23. These modifications are reflected by modifications of the ε-edges of $Sep(G)$ giving a graph $M(Sep(G))$ (where M means "modification") from which G' is obtained by contracting the ε-edges. All these manipulations can be done by MS transductions.

Example 2.24. These definitions are illustrated in Figures 1–3. Figure 1 shows a graph G and a graph H that is 2-isomorphic to G. We make G into a 2-dag, the two sources of which are the ends of edge a. The corresponding canonical term is:

$$t = //(\mathbf{a}, \bullet[//(\mathbf{c}, \bullet(\mathbf{b}, \mathbf{f})), \theta_K(\mathbf{g}, \mathbf{k}, \mathbf{m}, \mathbf{h}, //(\mathbf{n}, \mathbf{p})), //(\mathbf{d}, \mathbf{e})])$$

Figure 2: The separated graph $Sep(G)$.

where K is the graph K_4^- (defined as K_4 minus one edge). Figure 2 shows the corresponding graph $Sep(G)$. The ε-edges are represented by dotted lines. The graph $M(Sep(G))$ on Figure 3 is obtained by modifying certain ε-edges, and the modified edges are represented by broken lines. These modifications correspond to replacing in t the subterm $\bullet(\mathbf{b}, \mathbf{f})$ by $\bullet(\mathbf{f}, \mathbf{b})$, θ_K by $\theta_{\widetilde{K}}$ and the operation $\bullet$ occurring first in the subterm $\bullet[t_1, t_2, t_3]$ by $\bullet_\pi$ where $\pi(1) = 3$, $\pi(2) = 1$, $\pi(3) = 2$. For the purpose of readability, the edges of G are undirected.

We now detail the proof more formally.

Proof of Theorem 2.19. We choose in the given graph G two adjacent vertices, we make them into sources $s_1(G)$ and $s_2(G)$, and we change some edge directions to make G into a 2-dag. This is possible by Lemma 3.1 in [11] since G is 2-connected. Furthermore this can be done by an MS transduction taking $Inc(G)$ as input (by the reorientation technique of [8]). Hence, we obtain a 2-dag from G by a finite sequence of twistings if G is not a 2-dag, because reversing an edge is a twisting. Without loss of generality we now consider that the given graph G is a 2-dag. Using Lemma 2.16, we can construct the structure $Rep^{sep}(G)$, and from it the structure $Sep(G)$.
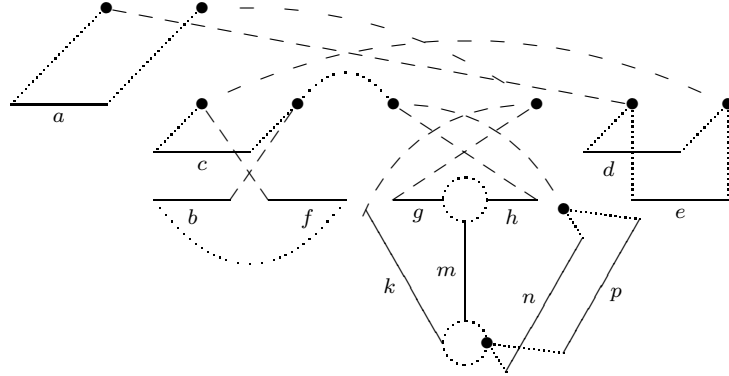
The tree $T = T_{\mathcal{SFE}(G)}$ in the structure $Rep^{sep}(G)$ is the syntactic tree of the canonical term t for G.

The terms t' in $\triangledown(t)$ are obtained by selecting:
  - a set X of prime nodes of T corresponding to an operation θ_K to be replaced by $\theta_{\widetilde{K}}$,
  - for each linear node x of arity k a k-permutation π such that the operation $\bullet$ at x is to be replaced by $\bullet_\pi$,
  - a set Y of leaves corresponding to reversals of edge directions.

The sets X, Y are straightforward to specify as parameters $X, Y \subseteq N_T$ of the MS transduction we are constructing.

The permutations associated with the linear nodes are obtained from a linear order $\preccurlyeq$ on V_G as follows. We call $s_2(G(x))$ the *leading vertex of the factor* $G(x)$ of G, for x in N_T. (See after Proposition 2.10 for the notation $G(x)$). A vertex may be leading for several factors. Let x be a linear node with sequence of sons $y_1, \ldots, y_k$. The linear order $\preccurlyeq$ on V_G will be used here to permute this sequence. The leading vertices of $G(y_1), \ldots, G(y_k)$

Figure 3: The graph $M(Sep(G))$.

are pairwise distinct. There exists a unique permutation π such that $s_2(G(y_{\pi(1)})) \prec \ldots$ $\prec s_2(G(y_{\pi(k)}))$. We obtain in this way a (possibly identity) permutation of the list of sons of x. It is clear that the new ordering of the sons of x is MS definable from $\preccurlyeq$ and the other relations of the structure $Rep^{sep}(G)$. The variable Π will denote families of permutations of appropriate types associated with linear nodes (a k-permutation for a node with k sons) and $\Pi(\preccurlyeq)$ will denote the one induced as defined above by a linear order $\preccurlyeq$ on V_G. We denote by $t_{X,Y,\Pi}$ the term obtained from t by the modifications described above, based on X, Y and Π. The set $\nabla(t)$ is thus the set of all terms $t_{X,Y,\Pi}$.

Claim 1: There exists an MS transduction that associates the graph defined by the term $t_{X,Y,\Pi(\preccurlyeq)}$ with the structure $(Rep^{sep}(G), X, Y, \preccurlyeq)$, where X is a set of prime nodes of the decomposition tree T, Y a set of leaves and $\preccurlyeq$ is a linear order on V_G.

Proof of the claim. By using X, Y and $\preccurlyeq$, we transform $Rep^{sep}(G)$ into a graph H' from which the graph G' defined by the term $t_{X,Y,\Pi(\preccurlyeq)}$ can be obtained by the MS transduction that contracts the ε-edges. The construction consists in modifying the ε-edges in $Sep(G)$, so as to represent the replacements of θ_K by $\theta_{\widetilde{K}}$ at the nodes in X, those of the operation $\bullet$ by $\bullet_\pi$ at every linear node where the corresponding permutation π is specified by $\Pi(\preccurlyeq)$, and the reversal of edges at the leaves of Y.

For this purpose we modify in $Rep^{sep}(G)$ the relation $\varepsilon\text{-}edg_H$ into $\varepsilon\text{-}edg_{H'}$ as follows:

1) For every x in X and every son y of x, we replace the pairs $((x, i), (y, j))$ and $((y, j), (x, i))$ in $\varepsilon\text{-}edg_H$ by $((x, 3 - i), (y, j))$ and $((y, j), (x, 3 - i))$.

2) For every linear node with ordered list of sons $y_1, y_2, \ldots, y_k$ we have in $\varepsilon\text{-}edg_H$ the following pairs, together with their inverses:

$$((x, 1), (y_1, 1)), ((y_1, 2), (y_2, 1)), \ldots, ((y_{k-1}, 2), (y_k, 1)), ((y_k, 2), (x, 2)) .$$

We replace them by the following ones, together with their inverses:

$$((x, 1), (y_{\pi(1)}, 1)), ((y_{\pi(1)}, 2), (y_{\pi(2)}, 1)), \ldots ((y_{\pi(k-1)}, 2), (y_{\pi(k)}, 1)), ((y_{\pi(k)}, 2), (x, 2))$$

where π is the k-permutation of the family $\Pi(\preccurlyeq)$ associated with x,

3) For every y in Y, we replace a triple $(e, (y, 1), (y, 2))$ in inc_H by $(e, (y, 2), (y, 1))$.

This modification of $Rep^{sep}(G)$ can be done by an MS transduction using X, Y and $\preccurlyeq$. We obtain in this way the graph $M(Sep(G))$ from which can be defined by edge contractions the value G' of the term $t_{X,\Pi(\preccurlyeq)}$. It follows from Lemma 2.16 that G' can be obtained from

$M(Sep(G))$, whence also from $(Rep^{sep}(G), X, Y, \preccurlyeq)$ by an MS transduction (we use here Proposition A.1.2). This proves Claim 1.   □

Since $Rep^{sep}(G)$ can be constructed from $Inc(G)$ by an MS transduction, and by composing this transduction with the one just constructed, we get an MS transduction τ that defines *some* graph 2-isomorphic to G from $Inc(G), X, Y$ and any linear order $\preccurlyeq$ on V_G. To complete the proof, it remains to establish that *every* family Π of permutations (of appropriate types) is $\Pi(\preccurlyeq)$ for some linear order $\preccurlyeq$ on V_G. This will give us that every graph 2-isomorphic to G is obtained by τ from $Inc(G)$ for some sets X and Y and *some* linear order $\preccurlyeq$ on V_G.

Claim 2: Every family Π of permutations associated with the linear nodes is $\Pi(\preccurlyeq)$ for some linear order $\preccurlyeq$ on V_G.

Proof of the claim. Let a family Π be given. By using bottom up induction on T, we construct, for every node x of T, an appropriate linear order on the vertices of $V_{G(x)} - \{s_1(G(x))\}$ handled as the increasing sequence $Seq(x)$ of its elements.

Let us note that if x and y are incomparable nodes in the tree T (*incomparable* means that no one is an ancestor of the other) then the only vertices that can be common to $G(x)$ and $G(y)$ are among the source vertices.

We can construct the sequences $Seq(x)$ by bottom up induction as follows:

If x is a leaf (it corresponds to an edge of G) then $Seq(x)$ is the sequence with single element $s_2(G(x))$.

If $x = //(x_1, \ldots, x_k)$ we have sequences $Seq(x_i)$ by induction. Since two distinct factors $G(x_i)$ and $G(x_j)$ have only in common the sources $s_a(G(x)) = s_a(G(x_i)) = s_a(G(x_j))$ for $a = 1, 2$, the only vertex common to two sequences $Seq(x_i)$ and $Seq(x_j)$ is $s_2(G(x_i)) = s_2(G(x_j)) = s_2(G(x))$, and all sequences $Seq(x_i)$ can be merged into a single one, that we can take as $Seq(x)$.

If $x = \bullet(x_1, \ldots, x_k)$ and π is the permutation of $\{1, \ldots, k\}$ associated with x (by the family Π), then, since the sequences $Seq(x_i)$ obtained by induction are pairwise disjoint, we can take $Seq(x) = Seq(x_{\pi(1)}) \ldots Seq(x_{\pi(k)})$. By this construction, the permutation associated with x by any linear order for which $Seq(x)$ is increasing is actually π.

If $x = \theta_K(x_1, \ldots, x_k)$ we have sequences $Seq(x_i)$ by induction. Two sequences $Seq(x_i)$ and $Seq(x_j)$ either are disjoint or share the only vertex $s_2(G(x_i)) = s_2(G(x_j)) = v$, in the case where the two edges x_i, x_j of K have the same target v. Hence we can merge all sequences $Seq(x_i)$ such that x_i has target v into a sequence $L(v)$. Then we concatenate the sequences $L(v)$ for all vertices of v of K (except for $v = s_1(K)$), which gives $Seq(x)$.

Every sequence $Seq(x)$ is a subsequence of $Seq(y)$ if y is an ancestor of x by this construction. From the choice made for linear nodes x it follows that the sequence $Seq(root_T)$ of the root of T yields the appropriate permutation of Π at each x. To have a linear order on V_G, we add to $Seq(root_T)$ the vertex $s_1(G)$ as very first element, proving Claim 2.   □

This concludes the proof of the theorem.   □

By a simple counting argument, one can see that it is impossible to specify all permutations of arbitrarily large sets X with k-tuples of subsets of X for fixed k. For this reason, we specify the permutations associated with the linear nodes by linear orders on the vertices.

3. Partitive families of bipartitions

The general framework for split decomposition is defined by Cunnigham and Edmonds in [18]. It applies to other cases, in particular to hypergraphs and matroids. Our presentation owes a lot to the dissertation of Montgolfier [19].

3.1. Definitions and general properties.

We define families of bipartitions of a set V associated with a partition of this set, the blocks of which are organized into an unrooted tree. These definitions generalize two important examples: the decomposition defined by Tutte of a 2-connected graph in 3-connected components and the *split decomposition* defined by Cunnigham [17].

Definition 3.1. *Bipartitions, overlapping bipartitions.* A *bipartition* of a nonempty set V is an unordered pair of subsets, $P = \{A, B\}$ such that $V = A \cup B$, $A \cap B = \varnothing$, $\{A, B\} \neq \{\varnothing, V\}$. The sets A, B will be called the *blocks* of P. We denote by $\mathcal{B}(V)$ the set of bipartitions of V. Two bipartitions P and Q *overlap* if $A \perp B$ for all $A \in P$ and $B \in Q$. Hence P and Q do not overlap if and only if $A \cap B = \varnothing$ for some $A \in P$ and $B \in Q$. A bipartition of the form $\{\{v\}, V - \{v\}\}$ does not overlap any bipartition.

By an *unrooted tree* we mean a simple undirected connected graph without cycles (and without loops). It has no root, and the *leaves* are the nodes of degree 1. Its other nodes are the *internal* nodes. The sets of nodes and of internal nodes of a tree T are denoted by N_T and N_T^{int} respectively. For each edge $e : x - y$ of T, we denote by $T(x, y)$ the set of nodes, including x, that are reachable from x by a path that *does not* use edge e.

Let T be an unrooted tree with at least two nodes and $\mathcal{V} = (V(x))_{x \in N_T}$ be a partition of a set V such that $V(x)$ is not empty if x is a node of degree 1 or 2. We call $(T, \mathcal{V})$ a *tree-partition* of V. For each edge $e : x - y$ of T, we let $P_e = \{P_x, P_y\}$ where P_x is the union of the sets $V(z)$ for $z \in T(x, y)$, and similarly for P_y with $T(y, x)$. The family $\mathcal{B} = \mathcal{B}(T, \mathcal{V})$ of bipartitions P_e is not empty and satisfies the following property:

B1: no two bipartitions of $\mathcal{B}$ overlap.

If $V(x)$ is empty for every internal node x and is singleton for each leaf x, then $\mathcal{B} = \mathcal{B}(T, \mathcal{V})$ satisfies in addition the property:

B0: $\{\{v\}, V - \{v\}\} \in \mathcal{B}$ for every $v \in V$.

For a tree-partition $(T, \mathcal{V})$ we define $box_T(v, x)$ to hold if and only if $x \in N_T$ and $v \in V(x)$. For every nonempty family $\mathcal{B}$ of bipartitions, we let $\mathcal{B}^+ = \mathcal{B} \cup \{\{\{v\}, V - \{v\}\} \mid v \in V\}$. Since $\{\{v\}, V - \{v\}\}$ does not overlap any bipartition, $\mathcal{B}^+$ satisfies B1 if and only if $\mathcal{B}$ satisfies B1. A block of a bipartition of $\mathcal{B}$ is called a $\mathcal{B}$-*block*.

Lemma 3.2. *For every nonempty family $\mathcal{B} \subseteq \mathcal{B}(V)$ satisfying B1, there exists a tree-partition $(T, \mathcal{V})$ such that $\mathcal{B}(T, \mathcal{V}) = \mathcal{B}$. It is unique up to isomorphism.*

Proof. We first make some observations concerning $\mathcal{B}(T, \mathcal{V})$ where $(T, \mathcal{V})$ is a tree-partition, by using the notation of the definition.

Claim: A $\mathcal{B}$-block is minimal if and only if it is $V(x)$ for some leaf x of T.

Proof of the claim. It is clear that $V(x)$ is a minimal $\mathcal{B}$-block if x is a leaf. For the other direction, assume that A is a minimal $\mathcal{B}$-block and $A = P_x$ where $P_e = \{P_x, P_y\}, e : x - y$. If x is not a leaf there is an edge $x - z$, $z \neq y$, and P_z is a proper subset of P_x by the condition that $V(u)$ is not empty if u is a node of degree 1 or 2. Hence, this cannot happen, x is a leaf and $A = P_x = V(x)$. $\qquad\square$

We first prove the unicity property. Assume $\mathcal{B}(T,\mathcal{V}) = \mathcal{B}(T',\mathcal{V}')$ for two tree-partitions.

Let R be a minimal block. We have $R = V(r)$ for some leaf r of T by the claim. We make T into a rooted (directed) tree with root r and we let s be adjacent to r in T. Let $V_1(u) = V(u)$ for every node $u \neq r$. By Definition 2.1, $T(s)$ is the subtree of T with root s; its nodes are those of T except r. Clearly, $\mathcal{F}(T(s),\mathcal{V}_1)$ is the set of $\mathcal{B}$-blocks that do not include R.

Similarly for $\mathcal{B}(T',\mathcal{V}')$ we have $R = V'(r')$ for some leaf r' of T', we let s' be adjacent to r' in T', and we have $\mathcal{F}(T'(s'),\mathcal{V}'_1) = \mathcal{F}(T(s),\mathcal{V}_1)$. For a node u of $T(s)$ of outdegree 1, hence of degree 2 in T, the set $V_1(u)$ is not empty, and the same holds for $(T'(s'),\mathcal{V}'_1)$. Hence, by an observation made in Definition 2.1, $(T(s),\mathcal{V}_1)$ and $(T'(s'),\mathcal{V}'_1)$ are isomorphic. So are $(T,\mathcal{V})$ and $(T',\mathcal{V}')$, as was to be proved.

We now prove the existence $(T,\mathcal{V})$ such that $\mathcal{B}(T,\mathcal{V}) = \mathcal{B}$ where $\mathcal{B} \subseteq \mathcal{B}(V)$ satisfies B1. We let R be a minimal block and $\mathcal{F}$ be the set of $\mathcal{B}$-blocks that do not include R. Hence $\mathcal{F}$ is a family of subsets of $V - R$ that satisfies P0. It satisfies Condition P1 because if A and A' in $\mathcal{F}$ overlap, then $\{A,B\}$ and $\{A',B'\}$ overlap since $R \subseteq B \cap B'$ and this cannot happen since $\mathcal{B}$ satisfies B1.

Let $(T_\mathcal{F},V_\mathcal{F})$ be as in Definition 2.1. Hence $\mathcal{F}(T_\mathcal{F},V_\mathcal{F}) = \mathcal{F}$. We recall that $V_\mathcal{F}(N) = N - \bigcup\{M \mid M$ is a son of N in $T_\mathcal{F}\}$. We add a new node r linked to the root s of $T_\mathcal{F}$, we denote by T the undirected tree obtained in this way, and we let $V(r) = R$, $V(N) = V_\mathcal{F}(N)$ if N is a node of $T_\mathcal{F}$. Then (T,V) is a tree-partition of V. In particular if N has degree 2, then $V(N)$ is not empty because N has outdegree 1 in $T_\mathcal{F}$ and $V_\mathcal{F}(N)$ is not empty. We claim that $\mathcal{B}(T,\mathcal{V}) = \mathcal{B}$.

Let P_e be the bipartition of V associated with an edge e of T. If $e : r - s$, then $P_e = \{R,V - R\}$ which belongs to $\mathcal{B}$. Otherwise let e be directed $x \longrightarrow y$ in $T_\mathcal{F}$. Then $P_e = \{P_x,P_y\}$. The set P_y is the $\mathcal{F}$-module associated with the node y of $T_\mathcal{F}$. Hence $\{P_y,V - P_y\} \in \mathcal{B}$. But $V - P_y = P_x$. Hence $P_e \in \mathcal{B}$.

Conversely, let $P = \{A,B\} \in \mathcal{B}$. If $P = \{R,V - R\}$ it is in $\mathcal{B}(T,\mathcal{V})$, corresponding to the edge $r - s$. Otherwise it does not overlap $\{R,V - R\}$ and since R is minimal, we have $R \subset A$ or $R \subset B$. Assume the first. Then $B \in \mathcal{F}$, hence is a node y of $T_\mathcal{F}$, its father is some x and we have $\{B,V - B\} \in \mathcal{B}(T,\mathcal{V})$. But $A = V - B$, hence $P \in \mathcal{B}(T,\mathcal{V})$. This completes the proof. $\square$

We denote by $(T_\mathcal{B},\mathcal{V}_\mathcal{B})$ the tree-partition associated with $\mathcal{B}$ by Lemma 3.2. It does not depend on the choice of r by the unicity property. It will be useful to extend this definition to the case where $\mathcal{B}$ is empty: we let then $T_\mathcal{B}$ consist of a single node r and $V_\mathcal{B}(r) = V$.

Lemma 3.3. *Let $\mathcal{B} \subseteq \mathcal{B}(V)$ satisfy B1. For a node $x \in N_{T_\mathcal{B}}$ of degree k with incident edges $e_1 : x - y_1, \ldots, e_k : x - y_k$, the sets $P_{y_1}, \ldots, P_{y_k}$ (where $P_{e_i} = \{P_x^i, P_{y_i}\}$) are pairwise disjoint and we have:*

$$V_\mathcal{B}(x) = V - (P_{y_1} \cup \cdots \cup P_{y_k}) = \bigcap\{P_x^i \mid i = 1,\ldots,k\} .$$

If x is a leaf, then $k = 1$ and $V_\mathcal{B}(x) = P_x^1$.

Proof. This is clear from the construction of Lemma 3.2, and the definition of $\overline{V_\mathcal{F}}$ in Definition 2.1. $\square$

Let $\mathcal{C}$ be a class of relational structures as in Section 2. For each S in $\mathcal{C}$, we let $\mathcal{B}(S)$ be a family of bipartitions of the domain D_S of S satisfying condition B1. We say that $\mathcal{B}$ is *MS-definable* if there exists an MS formula $\varphi(X)$ such that for every S in $\mathcal{C}$, $\{A \mid \{A,B\} \in \mathcal{B}(S)$ for some $B\} = \{A \subseteq D_S \mid S \models \varphi(A)\}$. With these definitions:

Proposition 3.4. *Let $\mathcal{C}$ be a class of $\mathcal{R}$-structures and $\mathcal{B}$ be an MS definable family of bipartitions of the domains of the structures in $\mathcal{C}$ which satisfies conditions B1. There exists a domain extending MS-transduction that associates with $(S, \preccurlyeq)$ where $S = \langle D_S, (R_S)_{R \in \mathcal{R}} \rangle \in \mathcal{C}$ and D_S is linearly ordered by $\preccurlyeq$, the structure*

$$Dec(S) = \langle D_S \cup N_{T_{\mathcal{B}(S)}}, (R_S)_{R \in \mathcal{R}}, edg_{T_{\mathcal{B}(S)}}, box_{T_{\mathcal{B}(S)}} \rangle$$

such that $\langle N_{T_{\mathcal{B}(S)}}, edg_{T_{\mathcal{B}(S)}} \rangle = T_{\mathcal{B}(S)}$.

Proof. Lemma 3.2 reduces the construction of the tree $T_{\mathcal{B}(S)}$ to that of a tree associated with a family $\mathcal{F}$ of subsets of D_S. Since the structure S is linearly ordered, one can take for R the unique minimal $\mathcal{B}(S)$-block that contains the $\preccurlyeq$-smallest element of D_S. The corresponding family $\mathcal{F}$ is thus MS definable. Using Proposition 2.2, an MS transduction can construct the corresponding rooted tree $T_{\mathcal{F}}$, modified so as to yield the tree T (cf. the proof of Lemma 3.2). One gets the desired unrooted tree $T_{\mathcal{B}(S)} = \langle N_{T_{\mathcal{B}(S)}}, edg_{T_{\mathcal{B}(S)}} \rangle$. The definition of the relation $box_{T_{\mathcal{B}(S)}}$ is easy to write in MS logic. $\square$

The constructed structure is, up to isomorphism, independent on $\preccurlyeq$, by the unicity result of Lemma 3.2. This proposition has a corollary fully similar to Corollary 2.5 of Proposition 2.2.

Definition 3.5. *Partitive families of bipartitions.* Let V be a nonempty set. A family $\mathcal{B}$ of bipartitions of V is *weakly partitive* if it satisfies the following property:

B2: For every two overlapping elements P and Q of $\mathcal{B}$, we have $\{A \cap B, A' \cup B'\} \in \mathcal{B}$, whenever $\{A, A'\} = P$ and $\{B, B'\} = Q$.

It is *partitive* if, in addition, it satisfies the following property:

B3: For every two overlapping elements P and Q of $\mathcal{B}$, we have $\{A \Delta B, A \Delta B'\} \in \mathcal{B}$, whenever $\{A, A'\} = P$ and $\{B, B'\} = Q$.

Note that in B3, we have $A \Delta B' = A' \Delta B$.

The bipartitions of $\mathcal{B}$ will be called the $\mathcal{B}$-*splits* of V (or of the structure S, if $\mathcal{B} = \mathcal{B}(S)$). Those which do not overlap any other bipartition of $\mathcal{B}$ are called the *good* $\mathcal{B}$-splits. (We keep our terminology close to that of [18] and [17] which are the fundamental articles for these notions). If $\mathcal{B}$ is weakly partitive, the family $Good(\mathcal{B})$ of good $\mathcal{B}$-splits is nonempty: let A be a minimal $\mathcal{B}$-block among those containing an element v; if $\{A, V - A\}$ overlaps $\{B, C\}$ where B contains v, then, by B2, $\{A \cap B, (V - A) \cup C\} \in \mathcal{B}$, and A is not a minimal block containing v; hence $\{A, V - A\}$ is a good $\mathcal{B}-$split. Clearly, $Good(\mathcal{B})$ satisfies B1. The corresponding unrooted tree is $T_{Good(\mathcal{B})}$. If we transform a family $\mathcal{B}$ into $\mathcal{B}^+$ so as to insure Property B0, then $\mathcal{B}^+$ is weakly partitive or partitive if $\mathcal{B}$ is weakly partitive or partitive respectively.

If $\{A, B\}$ is a split of a structure S, we consider S as a composition of the smaller induced substructures $S[A]$ and $S[B]$. By iterating the splitting, one reaches a decomposition of S into unsplittable pieces. The objective is to obtain in this way a canonical decomposition.

As in Theorem 2.7, the conditions of partitivity and weak partitivity on a family $\mathcal{B}$ imply some particular structure associated with the nodes of $T_{Good(\mathcal{B})}$, and we also express this structural property for the tree $T_{Good(\mathcal{B}^+)}$. We recall that the leaves of this tree are the singletons $\{v\}$ for v in V. If N and M are adjacent nodes of $T_{Good(\mathcal{B}^+)}$, we also recall that $T_{Good(\mathcal{B}^+)}(N, M)$ denote the set of nodes of $T_{Good(\mathcal{B}^+)}$ (including N) that are reachable

from N by a path not containing M. For $\mathcal{B}$ defined by the context, we denote by $V(N, M)$ the set of elements of V at the leaves belonging to $T_{Good(\mathcal{B}+)}(N, M)$.

Theorem 3.6. ([18], [19]) *Let $\mathcal{B} \subseteq \mathcal{B}(V)$ be partitive.*
(1) *Every internal node N of the tree $T_{\mathcal{B}+}$ satisfies one and only one of the following two properties:*

S1: *N has k neighbours, $N_1, \ldots, N_k, k \geq 3$, and for every nonempty proper subset I of $\{1, \ldots, k\}$, the pair*
$$\mathcal{B}(N, I) := \left\{ \bigcup \{V(N_i, N) \mid i \in I\}, \bigcup \{V(N_i, N) \mid i \in \{1, \ldots, k\} - I\} \right\}$$
belongs to $\mathcal{B}$.

S2: (1) *N has k neighbours, $N_1, \ldots, N_k, k \geq 3$, and for every subset I of $\{1, \ldots, k\}$, the pair $\mathcal{B}(N, I)$ (as defined above) belongs to $\mathcal{B}$ if and only if I or $\{1, \ldots, k\} - I$ is singleton.*

(2) *If a $\mathcal{B}$-split is not good, it is of the form $\mathcal{B}(N, I)$ for some node N satisfying T1 and a non singleton set $I \subset \{1, \ldots, k\}$.*

   Let $\mathcal{B} \subseteq \mathcal{B}(V)$ be weakly partitive.
(3) *Every internal node N of the tree $T_{Good(\mathcal{B}+)}$ satisfies one and only one of properties S1, S2 or the following*

S3: *N has at least 3 neighbours that can be numbered as $N_1, \ldots, N_k$ in such a way that for every subset I of $\{1, \ldots, k\}$, the pair $\mathcal{B}(N, I)$ belongs to $\mathcal{B}$ if and only if I is an interval $[m, n]$ or its complement for some m, n with $1 \leq m \leq n \leq k$ (and $\{1, \ldots, k\} \neq [m, n]$).*

(4) *If a $\mathcal{B}$-split is not good, it is of the form $\mathcal{B}(N, I)$ for some node N satisfying S1 or S3 (with $m < n$).* □

   The nodes satisfying S1, S2, S3 are said to be, respectively, *complete*, *prime*, and *circular*. Montgolfier's dissertation [19] reviews several applications from [18], together with other ones that we do not discuss here.

3.2. **The Tutte decomposition of 2-connected graphs.** We review briefly the Tutte decomposition of 2-connected graphs used in [18] to introduce the theory of graph decomposition. This notion does not depend on edge directions, hence graphs will be *undirected* in this section. They are loop-free, without isolated vertices, they may have multiple edges. The notation and definitions of Subsection 2.3 for 2-graphs are used here with obvious adaptations to undirected graphs.

Definition 3.7. *2-separations.* A *2-separation* of a graph G is a bipartition $\{A, B\}$ of its set of edges E_G such that A and B have at least two elements and there are exactly two vertices, u and v, which are incident with edges from both blocks of the bipartition. (For example $\{\{a, b, c, f\}, \{d, e, g, h, k, m, n, p\}\}$ is a 2-separation of the graph G of Figure 1.) Hence, $G = (G_{uv}[A]//G_{uv}[B])^0$, where $s_1(G_{uv}[A]) = s_1(G_{uv}[B]) = u$, $s_2(G_{uv}[A]) = s_2(G_{uv}[B]) = v$, $G_{uv}[A]^0 = G[A]$ and similarly for B. For the purpose of iterating the decomposition process, it is convenient to consider that the two graphs resulting from this decomposition step are $G^+[A]$ and $G^+[B]$, obtained from $G[A]$ and $G[B]$ by the addition of a new undirected edge $u - v$, labelled in a particular way, and called a *marker*. The graphs $G^+[A]$ and $G^+[B]$ have in common the marker edge, its two ends and nothing else. They have no distinguished vertices. The decomposition process is applied to them recursively.

If G is 2-connected, then $G[A]$ and $G[B]$ are connected, and furthermore $G^+[A]$ and $G^+[B]$ are 2-connected ([18], Lemma 1). A graph without any 2-separation is 3-connected. We denote by $2\mathcal{S}(G)$ the set of 2-separations of a graph G.

A *decomposition of a graph* H is a set of graphs (called the *components* of the decomposition) which is either $\{H\}$ or the set obtained from a decomposition by replacing one of its components, say G, by $G^+[A]$ and $G^+[B]$ defined from a 2-separation $\{A, B\}$ of G. This process is applied recursively to a 2-connected graph H and each component of a decomposition is 2-connected. The graphs in a decomposition are not disjoint, they form a single connected graph. If we delete from this graph the marker edges, we obtain H. To every decomposition corresponds a tree, the nodes of which are the components of the decomposition. Two nodes are adjacent if they share a marker edge.

It is proved in [18] that the family $2\mathcal{S}(G)$ for a 2-connected graph G is weakly partitive. If we decompose a 2-connected graph by using only good bipartitions at each step, we obtain at the end a canonical (unique up to isomorphism) decomposition (Theorem 1 of [18]). This canonical decomposition is the one defined by Tutte and proved unique in [31], chapter 11: every 2-connected graph has a unique decomposition in terms of *bonds* (i.e., graphs consisting of several parallel edges between two vertices), cycles and 3-connected graphs such that no two bonds and no two cycles share an edge.

Proposition 3.4 is applicable and shows that the tree of the Tutte decomposition can be constructed by an MS transduction using an auxiliary order on the set of edges. However, another construction, not using any linear order is given in [11], Theorem 4.7. It uses the detour through 2-dags, as in the proof of Lemma 2.20. We do not discuss any longer this construction.

Question: In the case of a simple graph G, one might hope, by using Corollary 2.12 to be able to construct the structure $\langle V_G \cup N_T, edg_G, edg_T, box_T \rangle$ from $\langle V_G, edg_G \rangle$ by an MS transduction. However this is not immediate from the above results because the proof of this corollary is valid for directed graphs, and in order to orient the edges of a graph, edge set quantifications are necessary (see [8]). However, an alternative construction might be possible, giving a statement analogous to Corollary 2.12. We leave this as an open question.

4. The split decomposition

In this section we apply the results of Section 3 to the *split decomposition* of graphs defined by Cunnigham in [17] and used as a preliminary step in several algorithms: for the polynomial time recognition of circle graphs in [2], for the recognition of *parity graphs* in [5] and for the construction of *distance labellings* in [21].

4.1. **Splitting a graph.** As in Subsection 2.2, graphs are simple, directed and loop-free. The split decomposition will be applied to connected graphs. Hence, most definitions are restricted to connected graphs, which permits to avoid some technical difficulties. A directed graph is *strongly connected* if for any two vertices u, v, there are directed paths from u to v and v to u. An undirected graph is connected if and only if it is strongly connected.

Definition 4.1. *Splitting a graph.* A *split* of a connected graph G is a bipartition $\{A, B\}$ of V_G such that $E_G = E_{G[A]} \cup E_{G[B]} \cup (A_1 \times B_1) \cup (B_2 \times A_2)$ for some $A_i \subseteq A$, $B_i \subseteq B$, and each of A and B has at least 2 elements. If $\{A, B\}$ is a split, then G can be expressed

as the union of $G[A]$ and $G[B]$ linked by one or two directed, complete bipartite graphs. (Since G is connected the set $(A_1 \times B_1) \cup (B_2 \times A_2)$ is not empty).

The inverse of splitting is the *join operation*, defined as follows. Let H and K be two disjoint graphs with distinguished vertices h in H and k in K. We define $H \boxtimes_{(h,k)} K$ as the graph with set of vertices $V_H \cup V_K - \{h, k\}$ and edges $x \longrightarrow y$ such that, either $x \longrightarrow y$ is an edge of H, or an edge of K, or we have $x \longrightarrow h$ in H and $k \longrightarrow y$ in K, or we have $h \longrightarrow y$ in H and $x \longrightarrow k$ in K. The subscript (h, k) in $\boxtimes_{(h,k)}$ will be omitted whenever possible.

If $\{A, B\}$ is a split, then $G = H \boxtimes_{(h,k)} K$ where H is $G[A]$ augmented with a new vertex h and edges $x \longrightarrow h$ whenever there are in G edges from x to some u in B, and edges $h \longrightarrow x$ whenever there are edges from some u in B to x. The graph K is defined similarly from $G[B]$, with a new vertex k. These new vertices are called *markers* in [17]. We say that h and k are *neighbours* if they are created from a same split. Note that the graphs H and K have at least 3 vertices and strictly less vertices than G.

A technical variant (used in [17]) consists in letting $h = k$. In this case the graphs H and K have in common the marker vertex h and nothing else. We write in this case $G = H \boxtimes_{(h,h)} K$. The advantage is that $H \cup K$ is a single connected graph. However, the marker must be identified in some way. But when one iterates the decomposition process, it is easier to handle of the components of the decomposition as disjoint graphs.

Definition 4.2. *Decompositions.* A *decomposition* of a connected graph G is defined inductively as follows: $\{G\}$ is the only decomposition of size 1; if $\{G_1, \ldots, G_n\}$ is a decomposition of size n and $G_i = H \boxtimes_{(h,k)} K$, then $\{G_1, \ldots, G_{i-1}, H, K, G_{i+1}, \ldots, G_n\}$ is a decomposition of G of size $n + 1$. The graphs G_i are called the *components* of the decomposition. The graph G can be reconstructed without ambiguity provided the marker vertices and their matchings are specified. We say that two components are *neighbours* if they have neighbour marker vertices. From the inductive definition of decompositions, it is clear that the components of a decomposition form an unrooted tree for the neighbourhood relation.

It will be convenient to handle a decomposition $\mathcal{D} = \{G_1, \ldots, G_n\}$ of a graph G as a single graph $Sdg(\mathcal{D})$ called a *split decomposition graph*, an *SD graph* in short. The components of $\mathcal{D}$ being pairwise disjoint, we let $Sdg(\mathcal{D})$ be their union together with particular edges, called ε-edges between any two neighbour marker vertices. Every vertex of G is a vertex of $Sdg(\mathcal{D})$. The graph G can be reconstructed in a unique way from $Sdg(\mathcal{D})$. Two decompositions $\mathcal{D}$ and $\mathcal{D}'$ of a graph G are *isomorphic* if there exists an isomorphism of $Sdg(\mathcal{D})$ onto $Sdg(\mathcal{D}')$ which is the identity on V_G.

The objective is to construct for every connected graph a *canonical decomposition* by iterated good splittings. Figure 4 shows a graph G and Figure 5 shows the graph representing its canonical split decomposition. The dotted lines are the ε-edges.

In the perspective of obtaining a canonical decomposition, we first observe that the graphs H and K associated with a split $\{A, B\}$ of a graph G such that $G = H \boxtimes K$, H contains A and K contains B are not always uniquely defined. Consider G: $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4$, $A = \{1, 2\}$, $B = \{3, 4\}$. One can take $H = 1 \longrightarrow 2 \longrightarrow h$, $K = k \longrightarrow 3 \longrightarrow 4$, but one can also add an edge: $h \longrightarrow 2$ to H, or an edge from $3 \longrightarrow k$ to K (but not both simultaneously), and we still have $G = H \boxtimes_{(h,k)} K$. However, H and K are uniquely defined in certain situations, as shows the following lemma.
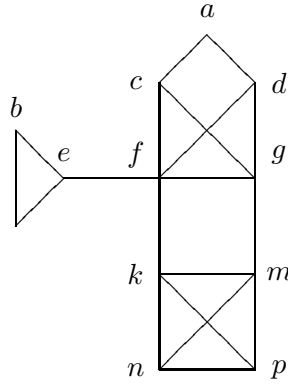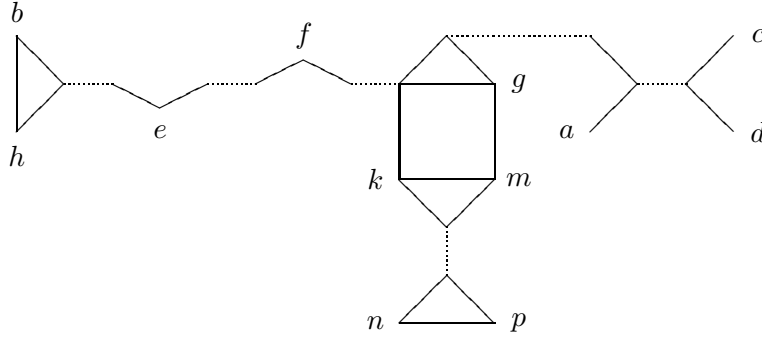
Figure 4: A graph G.

Figure 5: The split decomposition graph $Sdg(Split(G))$.

Lemma 4.3.

1) *Let G be a strongly connected (resp. undirected and connected) graph and $\{A, B\}$ be a split. There is a unique pair of graphs (H, K) such that H contains A, K contains B and $G = H \boxtimes K$, where unique is meant up to isomorphism.*

2) *Furthermore, the graphs H and K are strongly connected (resp. undirected and connected). They are isomorphic to induced subgraphs of G or to graphs obtained from induced subgraphs of G by fusing one vertex of indegree 0 and one vertex of outdegree 0. If G is undirected, only the former case occurs.*

Proof. The verifications are easy. For assertion 2), if G is undirected then H is isomorphic to $G[A \cup \{v\}]$ where v is any vertex of B adjacent to some vertex of A. Otherwise, H is isomorphic to the graph obtained from $G[A \cup \{u, v\}]$ by the fusion of u and v where u is any vertex of B such that $x \longrightarrow u$ for some vertex x of A, and v is any vertex of B such that $v \longrightarrow x$ for some vertex x of A. It may happen that $u = v$. $\qquad\qquad\square$

Remark. These assertions are not true for nonconnected graphs: the undirected graph I_4 (where I_n has n vertices and no edge) is equal to $I_3 \boxtimes H$, where H consist of one isolated vertex and an edge, one end of which is the marker vertex h. If G is strongly connected,

the graphs H and K need not be induced subgraphs: consider $\overrightarrow{C}_4 = \overrightarrow{C}_3 \boxtimes \overrightarrow{C}_3$ where $\overrightarrow{C}_n$ denotes the directed circuit with n vertices.

The decomposition process must terminate because the components are getting smaller and smaller, and are thus at the end "unsplittable". A graph is *prime* if it has at least 4 vertices and no split. The graphs with at most 3 vertices have no split for the trivial reason that they have not enough vertices. They are not called prime. We give easy verifiable examples of prime and splittable graphs. For further reference, we put them into a lemma.

Lemma 4.4.
   1) *A prime graph is 2-connected.*
   2) *There is no prime undirected graph with 4 vertices.*
   3) *For each $n \geq 5$, the graph C_n is prime, and the graphs P_n, K_n, S_{n-1}, L_n, all with n vertices, are not.* □

As usual, we denote by K_n the *n-clique,* i.e., the complete undirected graph with n vertices, by S_n the *n-star* consisting of one vertex, the *center*, adjacent to n vertices by undirected edges, by P_n the undirected path with $n - 1$ edges and n vertices, by L_n the transitive (acyclic) tournament on n vertices (the directed graph of a strict linear order), by C_n the undirected cycle with n vertices. The graphs K_n, S_{n-1} for $n \geq 4$ are "highly decomposable", or *brittle* in the terminology of [18, 17]: every bipartition, each block of which has at least 2 elements is a split. They are the only undirected graphs with this property. The highly decomposable directed graphs have a more complex structure that we will review later.

The 2-connected undirected graphs having 4 vertices are K_4, C_4, and K_4^- (i.e., K_4 minus one edge). None of them is prime. The directed, 2-connected graph with 4 vertices defined as the union of the paths $a \longrightarrow b \longrightarrow c \longrightarrow d$ and $a \longrightarrow d \longrightarrow c$ is prime, as one checks by trying the three possibilities to split it.

Definition 4.5. *Canonical decompositions.* A decomposition of a connected undirected graph G is *canonical* if and only if:
   (1) each component is either prime or is isomorphic to K_n or to S_{n-1} for n at least 3,
   (2) no two clique components are neighbour,
   (3) two neighbour vertices in star components are both centers or both not centers.
If G has one or two vertices, we define $\{G\}$ as its canonical decomposition.

Restrictions (2) and (3) can be justified as follows: if two clique components, isomorphic to K_n and K_m are neighbour they can be merged into a single one isomorphic to K_{n+m-2}, by using the elimination of ε-edges described below and K_{n+m-2} has several overlapping splits ($n + m - 2 \geq 4$). Similarly, if two star components, isomorphic to S_n and S_m are neighbours, and the center of one is linked by an ε-edge to a non-center vertex of the other, they can be merged into a single star isomorphic to S_{n+m-1}, $n + m - 1 \geq 3$, and S_{n+m-1} has several overlapping splits. It is thus necessary to assume (2) and (3) in order to obtain a *unique decomposition theorem* because stars and cliques have several overlapping, hence "incompatible" splits. Note that the connected undirected graphs with 3 vertices are K_3 and S_2, hence are among the possible types of nonprime components.

As in Section 3, a split is *good* if it does not overlap other splits. Starting from a graph G and the decomposition $\{G\}$, one can refine it by iteratively splitting its components *with respect to good splits only.* Since a graph breaks into two strictly smaller graphs, one reaches

a decomposition that cannot be refined by any split. Since one only applies good splits, one cannot generate neighbour components that are cliques, or that are stars with a center neighbour to a non-center vertex. It is thus canonical.

The following theorem concerns *connected graphs.* By using the obvious decomposition of a graph into connected components, we get thus a canonical decomposition for every undirected graph. The isomorphism of decompositions is defined in Definition 4.2.

Theorem 4.6. [17, Theorem 3] *A connected undirected graph has a canonical decomposition. It is unique up to isomorphism. It can be obtained by iterated splitting relative to good splits.* $\square$

For directed graphs, there exists a similar notion of canonical decomposition, for which one needs another notion of "highly decomposable" graph, called a *circle of transitive tournaments,* a CTT in short. A *CTT* is a graph with $n \geq 3$ vertices $v_0, \ldots, v_{n-1}$, such that its edges are described in terms of a sequence of integers $0 = p_1 < p_2 < \cdots < p_k < p_{k+1} = n$ as the pairs $v_i \longrightarrow v_j$ such that $p_m \leq i < j \leq p_{m+1}$ for some $m, 1 \leq m \leq k$. (We let $v_n = v_0$.) In the special case $k = 1$, the loop $v_0 \longrightarrow v_0$ is excluded. The vertices $v_{p_1}, \ldots, v_{p_k}$ are called the *hinges.* We write that this graph is a k-CTT to specify the number k of hinges.

A CTT is strongly connected and is not undirected. Each of its splits has a block of the form $\{v_i, \ldots, v_j\}$ for some i, j with $0 \leq i < j \leq n - 1$.

Here are some examples: For $k = n$, one gets a directed circuit. For $k = 2, n = 4$, $p_1 = 0, p_2 = 2, p_3 = 4$ one gets the graph $0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 0$ with additional edges $0 \longrightarrow 2$ and $2 \longrightarrow 0$. For $k = 1, n = 3$, one gets the graph $0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 0$ with additional edges $1 \longrightarrow 0$ and $0 \longrightarrow 2$. A 1-CTT with n vertices has all its vertices of degree n, except the hinge which has degree $2n - 2$. (Since graphs are defined as directed, a vertex in a loop-free directed graph with n vertices has maximum degree $2n - 2$).

A decomposition of a strongly connected graph G is *canonical* if and only if:

(1) each component is either prime, or is isomorphic to K_n or to S_{n-1} for n at least 3, or is a CTT,

(2) and (3) hold as for undirected graphs,

(4) if two neighbour components are CTTs and each of them has at least two hinges, then the neighbour vertices are not two hinges.

If G has one or two vertices, we define $\{G\}$ as its canonical decomposition.

To justify the roles of cliques and stars, we recall that an undirected edge is defined as a pair of opposite directed edges. If two neighbour components in a decomposition are CTTs with respectively k and m vertices and $k' + 1$ and $m' + 1$ hinges, and two hinges are linked by an ε-edge e, then they can be merged (by what we will call in the next subsection the *elimination* of e) into a single $(k' + m')$-CTT with $k + m - 2$ vertices. This is shown on Figures 6 and 7 : two 3-CTTs are merged into a single 4-CTT. In all other cases where two CTTs are neighbour, the elimination of the ε-edge linking them does not yield a CTT, a star or a clique.

Theorem 4.7. [17, Theorem 2] *A strongly connected graph has a canonical decomposition. It is unique up to isomorphism. It can be obtained by iterated splitting relative to good splits.* $\square$
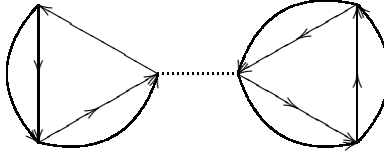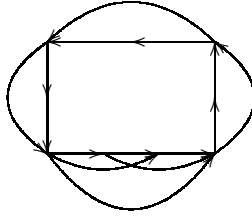
Figure 6: Two 3-CTT's linked by an ε-edge.

Figure 7: Two CTT's merged into a 4-CTT.

The *split decomposition* of a strongly connected graph G (which includes the case of a connected undirected graph) denoted by $Split(G)$ is the canonical decomposition of Theorems 4.6 and 4.7. In the next subsections we define its representation by a graph, and its construction by an MS transduction.

4.2. **Graph representations of decompositions.** We have defined a single graph $Sdg(\mathcal{D})$ linking all components of a decomposition $\mathcal{D}$. We obtain in this way a binary relational structure on a fixed finite signature, actually an edge-labelled graph, from which the decomposed graph can be reconstructed by an MS transduction, as we will see. We get something similar to the *graph representation of modular decompositions* defined in Section 2.

Definition 4.8. *Split decomposition graphs.* A *split decomposition graph*, (an *SD graph* in short) is a graph H with two types of edges, defined as a triple $\langle V_H, edg_H, \varepsilon\text{-}edg_H \rangle$ satisfying the following conditions

  (i) the *solid edges* are represented by a binary relation edg_H, and the (undirected) ε-*edges* are represented by a symmetric binary relation $\varepsilon\text{-}edg_H$;
 (ii) every vertex is incident to a solid edge;
(iii) no two ε-edges are adjacent;
 (iv) the graph obtained from H by contracting the solid edges is an undirected tree.

Condition (iv) implies that a cycle in H can only consist of solid edges, and that H is connected. The connected components of the subgraph $H[E_H^{sol}]$ where E_H^{sol} is the set of solid edges, are called the *components* of H. They are not isolated vertices and they are linked to one another by ε-edges, in the global shape of a tree.

Lemma 4.9. *The graph $Sdg(\mathcal{D})$ associated with a decomposition $\mathcal{D}$ of a connected graph G is an SD graph. Its vertices incident to no ε-edge are the vertices of G.* $\square$

We now explain how a graph can be reconstructed from the SD graph $Sdg(\mathcal{D})$ representing one of its decompositions.

Definition 4.10. *Evaluating SD graphs* If e is an ε-edge $u-v$ of an SD graph H, we define an SD graph $H' = Elim_e(H)$ as follows:

(a) $V_{H'} = V_H - \{u, v\}$,

(b) the edges of H' are those of H not incident to u or v, and the edges $x \longrightarrow y$ if $x \longrightarrow u - v \longrightarrow y$ or $x \longrightarrow v - u \longrightarrow y$ in H. (The edges $x \longrightarrow u$, $v \longrightarrow y$, $x \longrightarrow v$ and $u \longrightarrow y$ are necessarily solid edges).

We will say that this operation *eliminates* the edge e.

Lemma 4.11. *If H and K are two disjoint graphs and G is their union with an ε-edge linking h in H and k in K, then $Elim_e(G) = H \boxtimes_{(h,k)} K$. If e and f are two ε-edges of an SD graph H, we have $Elim_f(Elim_e(H)) = Elim_e(Elim_f(H))$.*

Hence one can eliminate simultaneously (or in any order) the ε-edges of a given set. We let $Eval(H)$ be obtained by eliminating all ε-edges of an SD-graph H. We use the notation $Eval$ because we consider this mapping as the evaluation of a kind of algebraic expression, the operations of which are defined by the components of H.

Lemma 4.12. *For an SD graph H, the graph $G = Eval(H)$ can be defined as follows*

(a') *V_G is the set of vertices of H incident to no ε-edge,*

(b') *the edges of G are the solid edges of H not adjacent to any ε-edge and the edges $x \longrightarrow y$ such that there is in H a path*

$$x \longrightarrow u_1 - v_1 \longrightarrow u_2 - v_2 \longrightarrow \ldots \longrightarrow u_k - v_k \longrightarrow y$$

where the edges $u_i - v_i$ are ε-edges and alternate with solid edges. $\square$

Example 4.13. The following graph H is an SD graph:

$$a \longrightarrow b \longrightarrow u - v \longrightarrow c \longleftarrow u' - v' \longleftarrow d \longleftarrow u" - v" \longrightarrow e$$

and $Eval(H)$ is the non connected graph: $a \longrightarrow b \longrightarrow c \longleftarrow d\ e$. This example shows that not every SD graph is associated with a decomposition of a connected graph.

Proposition 4.14. *If $\mathcal{D}$ is a decomposition of a connected graph G, then $Eval(Sdg(\mathcal{D})) = G$.*

Proof. By induction on the size k of $\mathcal{D}$. If $k = 1$, then we have $Sdg(\mathcal{D}) = G$. For the induction step, we let $\mathcal{D}' = \{G_1, \ldots, G_k\}$ be a decomposition with corresponding graph $Sdg(\mathcal{D}')$ such that $Eval(Sdg(\mathcal{D}')) = G$. We prove the assertion for $\mathcal{D} = \{G_1, \ldots, G_{k-1}, M, M'\}$, obtained by splitting one component, say G_k without loss of generality. The graph $Sdg(\mathcal{D})$ is obtained from $Sdg(\mathcal{D}')$ by the replacement of the subgraph G_k by the union of M and M' linked by an ε-edge, say e, and $Elim_e(Sdg(\mathcal{D})) = Sdg(\mathcal{D}')$. We have $Eval(Sdg(\mathcal{D})) = Eval(Elim_e(Sdg(\mathcal{D}))) = Eval(Sdg(\mathcal{D}'))$. Since $Eval(Sdg(\mathcal{D}')) = G$ by the induction hypothesis, we obtain $Eval(Sdg(\mathcal{D})) = G$. $\square$

The notion of *clique-width* of a directed or undirected graph G, denoted by $cwd(G)$, and a few results about it, are recalled in Appendix 2. It is defined for graphs with labelled edges, hence is applicable to SD graphs.

Proposition 4.15. *The mapping $Eval$ from SD graphs to graphs is an MS transduction. There exists a function f such that $Eval(H)$ has clique-width $\leq f(k)$ if each component of an SD graph H has clique-width $\leq k$.*

Proof. That the mapping $Eval$ is an MS transduction is clear from its definition and the fact that the transitive closure of an MS definable binary relation is MS definable.

For the second assertion, we use the fact for every MS transduction τ, there exists a function f such that $cwd(G') \leq f(cwd(G))$ whenever G' is obtained from G by τ. (See Lemma A.2.2). Hence assuming that each connected component of $H[E_H^{sol}]$ has clique-width $\leq k$, it is enough to prove that $cwd(H) \leq k + 2$.

We need a few technical facts about the algebraic expressions defining clique-width. We recall here that if C is a set of k labels, a C-expression defining a graph G witnesses that G has clique-width at most k (full definitions in Appendix 2). Let C and D be disjoint sets of labels. Let M be a graph with pairwise distinct vertices $v_1, \ldots, v_m$. Let $N_1, \ldots, N_m$ be pairwise disjoint graphs such that N_i has in common with M the single vertex v_i. We assume that M is defined by a C-expression, and that each N_i is defined by a $(C \cup D)$-expression, its vertices are labelled in D, and v_i has a label r_i that is different from those of the other vertices of N_i.

Claim 1: The graph $L = M \cup N_1 \cup \cdots \cup N_m$ can be defined by a $(C \cup D)$-expression.

Proof of the claim. Let E be an expression defining M. It has occurrences of constants $\mathbf{p}_1, \ldots, \mathbf{p}_m$ which define respectively the vertices $v_1, \ldots, v_m$. Let $F_1, \ldots, F_m$ be $(C \cup D)$-expressions defining respectively $N_1, \ldots, N_m$. The expressions $F_i' = ren_{r_i \to p_i}(F_i)$ define the graphs N_i with v_i now labelled by p_i. The desired $(C \cup D)$-expression for L is obtained by substituting in E the expressions $F_1', .., F_m'$ for the occurrences of $\mathbf{p}_1, \ldots, \mathbf{p}_m$ defining $v_1, \ldots, v_m$, giving an expression E'. Since E does not contain operations involving labels in D, the substitution of the expressions $F_1', .., F_m'$ in E does not result in edge creations between the vertices of the graphs N_i other than v_i and the vertices not in N_i. Hence, E' is a $(C \cup D)$-expression defining L. $\square$

We now continue the proof of the proposition. Let H be an SD graph with components of clique-width $\leq k$. We wish to prove that $cwd(H) \leq k + 2$. The case where H has a single component is obvious.

We let $D = \{\top, \bot\}$, and C be a set of k labels. For every ε-edge $e : v - u$, we let $H_{v,e}$ be the subgraph of H consisting of v, e and the connected component of u in the graph H minus the edge e. (We recall that the ε-edges link the components of H in the global shape of a tree). We label v by $\top$ and all other vertices of $H_{v,e}$ by $\bot$.

Claim 2: Each graph $H_{v,e}$ labelled in this way is definable by a $(C \cup D)$-expression.

Proof of the claim. The proof is by induction on the number of ε-edges of $H_{v,e}$. We let M be the component of H containing u, the other end of e. It is a subgraph of $H_{v,e}$. By the hypothesis, M is defined by a C-expression. We let $v_1, \ldots, v_n$ be the other vertices of M incident with ε-edges, respectively $e_1, \ldots, e_n$, which are the ε-edges linking M at vertices $v_1, \ldots, v_n$ to other components of H. Using induction, we obtain that each graph H_{v_i,e_i} is definable by a $(C \cup D)$-expression. We let N be the edge e, with v labelled by $\top$ and u labelled by $\bot$. Claim 1 is applicable to the graph $L = M \cup H_{v_1,e_1} \cup \cdots \cup H_{v_n,e_n} \cup N$, which is equal to $H_{v,e}$. Hence $H_{v,e}$ is definable by a $(C \cup D)$-expression.

This argument applies for $n = 0$ which is the basis of the induction. $\square$

The graph H is itself is expressible as $M \cup H_{v_1,e_1} \cup \cdots \cup H_{v_n,e_n}$ where M is any component, using the notation of Claim 2. Its proof yields the desired result since the cardinality of $C \cup D$ is $k + 2$. This completes the proof of the proposition. $\square$

We leave as an open question to determine a good bounding function f.

Proposition 4.16. *A set of strongly connected graphs has bounded clique-width if and only if the prime components of their split decompositions have bounded clique-width.*

Proof. We first consider undirected graphs (for them strong connectedness is just connectedness). The "only if" direction is clear because the prime components of the split decomposition of an undirected graph are isomorphic to induced subgraphs of this graph, and clique-width is monotone with respect to induced subgraph inclusion. (See Lemma A.2.1).

For the other direction, it suffices to apply Theorem 4.6 and Proposition 4.15 knowing that the cliques K_n and the stars S_n have clique-width 2.

We now consider directed graphs. We will use Theorem 4.7. For the "only if" direction, we note that, by Lemma 4.3, a prime component M of the split decomposition of a strongly connected graph G is either an induced subgraph of G or is obtained from an induced subgraph N by the fusion of a vertex of indegree 0 and a vertex of indegree 1. In this case, $cwd(N) \leq k$ implies $cwd(M) \leq 4k$ by Lemma A.2.3.

For the "if" direction, we argue as above, and it remains to prove that CTTs have bounded clique-width. Actually they have clique-width at most 4. Let G be a k-CTT with vertices $v_0, \ldots, v_{n-1}$, $n \geq 3$, and edges $v_i \longrightarrow v_j$ such that $p_m \leq i < j \leq p_{m+1}$ for some $m, 1 \leq m \leq k$, where $0 = p_1 < p_2 < \cdots < p_k < p_{k+1} = n$ and $v_n = v_0$. For every $i = 0, \ldots, n-1$, we let G_i be the subgraph of G defined as follows

(a) its vertices are $v_0, \ldots, v_i$,
(b) its edges are those of G of the form $v_j \longrightarrow v_k$ for $0 \leq j < k \leq i$ (hence G_{n-1} is G minus the edges towards v_0);
(c) its vertices are labelled as follows: we label v_0 by 1; letting m be such that $p_m \leq i < p_{m+1}$, we label v_j by $\perp$ if $0 < j < p_m$ and we label v_j by 2 if $p_m \leq j \leq i$.

The graphs G_i are defined by the following expressions:

 - $G_1 = add_{1,2}(\mathbf{1} \oplus \mathbf{2})$;
 - if $2 \leq i < p_2$, then $G_i = ren_{3 \to 2}(add_{2,3}(add_{1,3}(G_{i-1} \oplus \mathbf{3})))$;
 - if $p_m < i < p_{m+1}$ and $m \geq 2$, then $G_i = ren_{3 \to 2}(add_{2,3}(G_{i-1} \oplus \mathbf{3}))$;
 - if $i = p_2 > 1$, then $G_i = ren_{3 \to 2}(ren_{2 \to \perp}(add_{2,3}(add_{1,3}(G_{i-1} \oplus \mathbf{3}))))$,

and finally

 - if $i = p_m$ and $m > 2$, then $G_i = ren_{3 \to 2}(ren_{2 \to \perp}(add_{2,3}(G_{i-1} \oplus \mathbf{3})))$.

Then $G = add_{2,1}(G_{n-1})$. This shows that G can be constructed with the 4 labels $1, 2, 3, \perp$ hence has clique-width at most 4. If G has a single hinge, then $n - 1 < p_2 = n$ and labels $1,2,3$ suffice. Thus 1-CTTs have clique-width at most 3. $\square$

Remark. The clique-width of a graph may be strictly larger than the maximum clique-width of the components of its split decomposition. For an example the clique-width of P_4 is 3, $P_4 = P_3 \boxtimes P_3$ and the clique-width of P_3 is 2. By contrast, the clique-width of a graph is the maximum clique-width of its prime components for the modular decomposition (by Lemma A.2.1).

Another complexity measure for undirected graphs called *rank-width* is defined by Oum and Seymour (see [16, 27]). It is equivalent to clique-width in the sense that the same sets of undirected graphs have bounded clique-width and bounded rank-width (because $rwd(G) \leq cwd(G) \leq 2^{rwd(G)+1} - 1$ where $rwd(G)$ denotes the rank-width of G). The

rank-width of a graph is the maximal rank-width of its prime components for the split decomposition.

4.3. Monadic Second-Order definition of the split decomposition. The following theorem is actually the basis for Theorems 4.6 and 4.7.

Theorem 4.17. [17, Theorem 9]*: The family of splits of a strongly connected graph is weakly partitive. The family of splits of a connected undirected graph is partitive.*   □

Remark. This result may not hold for a graph that is not strongly connected. Take for example: $1 \longleftarrow 2 \longrightarrow 3 \longrightarrow 4 \longleftarrow 5 \longrightarrow 6$ with additional edge $6 \longrightarrow 1$. The two splits $\{\{1,2,3\},\{4,5,6\}\}$ and $\{\{2,3,4\},\{5,6,1\}\}$ overlap but $\{\{2,3\},\{4,5,6,1\}\}$ is not a split. Hence, the family of splits of this graph is not weakly partitive.

We denote by $\mathcal{BS}(G)$ the family of splits of a graph G, and by $\mathcal{BS}_g(G)$ the family of good ones. The tree $T_{\mathcal{BS}_g(G)}$ (defined in Section 3) is the tree of the split decomposition $Split(G)$. To simplify the notation, we will denote it by $T_{\mathcal{BS}(G)}$, remembering that it is based on good splits. Proposition 3.4 yields the following:

Proposition 4.18. *There exists an MS transduction that associates with a strongly connected graph G and a linear ordering $\preccurlyeq$ of its set of vertices the structure*

$$\langle V_G \cup N_{T_{\mathcal{BS}(G)}}, edg_G, edg_{T_{\mathcal{BS}(G)}}, box_{T_{\mathcal{BS}(G)}} \rangle$$

such that $T_{\mathcal{BS}(G)} = \langle N_{T_{\mathcal{BS}(G)}}, edg_{T_{\mathcal{BS}(G)}} \rangle$.   □

From the tree $T_{\mathcal{BS}(G)}$, we build an SD graph $H_{\mathcal{BS}(G)}$ and we will prove that it represents $Split(G)$, i.e. that $H_{\mathcal{BS}(G)} = Sdg(Split(G))$.

Definition 4.19. *The SD graph $H_{\mathcal{BS}(G)}$ constructed from $T_{\mathcal{BS}(G)}$.* To avoid special cases, we assume that G has at least 3 vertices. The tree-partition $(T_{\mathcal{BS}(G)}, \mathcal{V}_{\mathcal{BS}(G)})$ is defined by Lemma 3.2 from the family $\mathcal{BS}_g(G)$ (the set of good splits, which do not overlap any other). We let N be the set of nodes of the unrooted tree $T_{\mathcal{BS}(G)}$. An edge $e : x - y$ of $T_{\mathcal{BS}(G)}$ corresponds to a bipartition $\{P_x, P_y\} \in \mathcal{BS}_g(G)$.

For each such edge, we create two new vertices (e, x) and (e, y): they will be the marker vertices of Definition 4.1. More precisely, the nodes of $T_{\mathcal{BS}(G)}$ correspond to the components of the split decomposition, and the markers of the component at a node x will be the vertices (e, x) for all edges e of T incident with x.

For a node $x \in N$ with neighbours $y_1, \ldots, y_k$ we let $P_{y_1}, \ldots, P_{y_k}$ be the sets associated with the edges $e_1 : x - y_1, \ldots, e_k : x - y_k$ (we use the notation of Lemma 3.3). They are pairwise disjoint. By this lemma, $V_{\mathcal{BS}(G)}(x) = V_G - (P_{y_1} \cup \cdots \cup P_{y_k})$ (this set may be empty). We define a graph $H(x)$ as follows:

  (i) $V_{H(x)} = V_{\mathcal{BS}(G)}(x) \cup \{(e_i, x) \mid i = 1, \ldots, k\}$,
  (ii) its edges are of several types:
   - the edges $u \longrightarrow v$ in G, for $u, v \in V_{\mathcal{BS}(G)}(x)$,
   - the edges $u \longrightarrow (e_i, x)$ if $u \in V_{\mathcal{BS}(G)}(x)$ and there is in G an edge $u \longrightarrow v$ for some v in P_{y_i},
   - the edges $u \longleftarrow (e_i, x)$ if $u \in V_{\mathcal{BS}(G)}(x)$ and there is in G an edge $u \longleftarrow v$ for some v in P_{y_i},

- the edges $(e_i, x) \longrightarrow (e_j, x), i \neq j$ if there is in G an edge $u \longrightarrow v$ for some u in P_{y_i} and some v in P_{y_j}.

As we will prove, these graphs are the components of the split decomposition. In order to obtain an SD graph $H_{\mathcal{BS}(G)}$, we take their union and we link them by undirected ε-edges between (e, x) and (e, y) for every edge $e : x - y$ of $T_{\mathcal{BS}(G)}$. This completes the definition of $H_{\mathcal{BS}(G)}$. If G has no good split, then $\mathcal{BS}_g(G)$ is empty, the tree $T_{\mathcal{BS}(G)}$ has one node and no edge, and $H_{\mathcal{BS}(G)} = G$.

Proposition 4.20. *If a graph G is strongly connected with at least 3 vertices, we have* $H_{\mathcal{BS}(G)} = Sdg(Split(G))$ *and* $Eval(H_{\mathcal{BS}(G)}) = G$.

Proof. The proof is by induction on the number of vertices of G.

1) The case of graphs with 3 vertices is checked directly: each graph is a clique, a star or a CTT, hence is necessarily a component, $\mathcal{BS}(G)$ is empty, and $H_{\mathcal{BS}(G)} = G$.
2) If G has no good split, then it follows from [17], Theorems 10 and 11, that G is either S_n, or K_n, or a CTT, or is prime. In all cases we have $H_{\mathcal{BS}(G)} = G$.
3) If none of these cases hold, then G has a good split $\{A, B\}$ and G can be written as $H \boxtimes K$ in a unique way (Lemma 4.3) with $V_H \supseteq A$, $V_K \supseteq B$. We let h and k be their marker vertices (cf. Definition 4.1).

Claim 1: The tree $T_{\mathcal{BS}(G)}$ is the union of the trees $T_{\mathcal{BS}(H)}$ and $T_{\mathcal{BS}(K)}$ linked by an edge between x and y, where x is the node of $T_{\mathcal{BS}(H)}$ such that $h \in V_{\mathcal{BS}(H)}(x)$ and y is the node of $T_{\mathcal{BS}(K)}$ such that $k \in V_{\mathcal{BS}(K)}(y)$.

Proof of Claim 1. Property F3 of Theorem 8 of [17], states that for a split $\{A, B\}$, if $A' \subset A$, then $\{A', B \cup A - A'\}$ is a split of G if and only if $\{A', \{h\} \cup A - A'\}$ is a split of H. It follows that if $\{A, B\}$ is a good split, then, with H and K associated with it as above:

$$\mathcal{BS}_g(G) = \{\{A, B\}\} \cup \{\{A', C \cup B\} \mid \{A', C \cup \{h\}\} \in \mathcal{BS}_g(H)\}$$
$$\cup \{\{B', C \cup A\} \mid \{B', C \cup \{k\}\} \in \mathcal{BS}_g(K)\} .$$

This fact gives the bijection between $T_{\mathcal{BS}(G)}$ and the union of the trees $T_{\mathcal{BS}(H)}$ and $T_{\mathcal{BS}(K)}$ linked by an edge as in the statement. The edge $x - y$ corresponds to $\{A, B\}$. □

Claim 2: The graph $H_{\mathcal{BS}(G)}$ is isomorphic to the union of the graphs $H_{\mathcal{BS}(H)}$ and $H_{\mathcal{BS}(K)}$ linked by an ε-edge between h and k.

Proof of Claim 2. Let e be the ε-edge linking h and k. Let x_h and x_k be the nodes of $T_{\mathcal{BS}(H)}$ and $T_{\mathcal{BS}(K)}$ such that $h \in V_{\mathcal{BS}(H)}(x_h), k \in V_{\mathcal{BS}(K)}(x_k)$.

We denote by $H_{\mathcal{BS}(H)} + H_{\mathcal{BS}(K)}$ the union of the graphs $H_{\mathcal{BS}(H)}$ and $H_{\mathcal{BS}(K)}$ together with e where h is replaced by (e, x_h) and k by (e, x_k).

Our goal is to prove that $H_{\mathcal{BS}(G)} = H_{\mathcal{BS}(H)} + H_{\mathcal{BS}(K)}$. By Claim 1 and the definitions, the vertices of the graph $H_{\mathcal{BS}(G)}$ are those of $H_{\mathcal{BS}(H)} + H_{\mathcal{BS}(K)}$. It remains to prove that the edges are the same in both.

This is clear for the ε-edges as an immediate consequence of Claim 1. We now consider the various types of solid edges.

a) A solid edge of the form $u \longrightarrow v$, $u, v \in V_{\mathcal{BS}(G)}(x)$, where none of u and v is a vertex (f, y), is in $H_{\mathcal{BS}(G)}$ if and only if it is in $H_{\mathcal{BS}(H)} + H_{\mathcal{BS}(K)}$ because $V_{\mathcal{BS}(G)}(x) = V_{\mathcal{BS}(H)}(x) \cap V_G$ for x a node of $T_{\mathcal{BS}(H)}$ and similarly for K.

b) Consider a solid edge $(e_i, x) \longrightarrow (e_j, x)$. Without loss of generality, we assume that x is a node of $T_{\mathcal{BS}(H)}$.

Consider such an edge in $H_{\mathcal{BS}(G)}$: there is in G an edge $u \longrightarrow v$ for some u in P_{y_i} and some v in P_{y_j}, where $y_1, \ldots, y_n$ are the neighbours of x in $T_{\mathcal{BS}(G)}$ as in Definition 4.19.

Subcase 1: One of (e_i, x) or (e_j, x), say (e_j, x), is (e, x_h).

Then we have $u \longrightarrow v$ in G with $v \in P_{y_j} = B$. Hence, we have an edge $(e_i, x) \longrightarrow h$ in H, hence the edge $(e_i, x) \longrightarrow (e_j, x)$ in $H_{\mathcal{BS}(H)} + H_{\mathcal{BS}(K)}$ since $(e, x_h) = (e_j, x)$ replaces h.

Subcase 2: None of $(e_i, x), (e_j, x)$ is (e, x_h) or (e, x_k), u and v are both in H, and they are not h (because $u \longrightarrow v$ is an edge of G).

Then the edge $(e_i, x) \longrightarrow (e_j, x)$ is also in $H_{\mathcal{BS}(H)}$, because if we denote by P'_{y_i} the blocks like P_{y_i} relative to H, then we have either $P'_{y_i} = P_{y_i}$ or $P'_{y_i} = P_{y_i} - V_K \cup \{h\}$, by the result recalled in the proof of Claim 1.

Subcase 3: As in the previous subcase except that one of u, v, say u is in H, and the other is in K.

Then the edge $u \longrightarrow h$ is in H, and we also have the edge $(e_i, x) \longrightarrow (e_j, x)$ in $H_{\mathcal{BS}(H)}$ because $h \in P'_{y_j}$, since $P'_{y_j} = P_{y_j} - V_K \cup \{h\}$, with the notation of the previous subcase.

Conversely, let us assume that $(e_i, x) \longrightarrow (e_j, x)$ in $H_{\mathcal{BS}(H)}$. We have in H an edge $u \longrightarrow v$ for some u in P'_{y_i} and some v in P'_{y_j}.

Subcase 1: None of u, v is h, then we have also $(e_i, x) \longrightarrow (e_j, x)$ in $H_{\mathcal{BS}(G)}$, using the observation on the blocks P_{y_i}, P'_{y_i} made above in Subcase 2.

Subcase 2: If $u = h$, then we have $w \longrightarrow v$ in G for some w in K. Hence $(e_i, x) \longrightarrow (e_j, x)$ is in $H_{\mathcal{BS}(G)}$.

The arguments are of course the same with K in place of H.

c) Consider a solid edge $u \longrightarrow (e_i, x)$ in $H_{\mathcal{BS}(G)}$, $u \in V_{\mathcal{BS}(G)}(x)$. There is in G an edge $u \longrightarrow v$ for some v in P_{y_i}, where $e_1 : x - y_1, \ldots, e_n : x - y_n$ are the edges of $T_{\mathcal{BS}(G)}$ incident to x, as in Definition 4.19. There are several subcases:

Subcase 1: $u \in V_H$, $(e_i, x) = (e, x_h)$.

Then $v \in V_K$, but we have $u \longrightarrow h$ in $H_{\mathcal{BS}(H)}$. Hence the edge $u \longrightarrow (e, x_h)$ is in $H_{\mathcal{BS}(H)} + H_{\mathcal{BS}(K)}$.

Subcase 2: $u \in V_H$, $(e_i, x) \neq (e, x_h)$.

Then (e_i, x) is in $H_{\mathcal{BS}(H)}$. Either $v \in V_H$, and then the edge $u \longrightarrow (e_i, x)$ is also in $H_{\mathcal{BS}(H)}$ or $v \in V_K$, so the edge $u \longrightarrow h$ is in H and the edge $u \longrightarrow (e_i, x)$ is also in $H_{\mathcal{BS}(H)}$ because $h \in P'_{y_i}$.

The argument is similar if $u \in V_K$ and for the edges $u \longleftarrow (e_i, x)$.

Consider conversely a solid edge $u \longrightarrow (e_i, x)$ in $H_{\mathcal{BS}(H)}$, $u \in V(x)$, $u \neq h$. There is in H an edge $u \longrightarrow v$, where v in P'_{y_i} (a block relative to H, same notation as in case b).

If $v = h$, we have $u \longrightarrow w$ for some $w \in V_K$, hence $u \longrightarrow (e_i, x)$ in $H_{\mathcal{BS}(G)}$. If $v \neq h$, we have $u \longrightarrow v$ in G, hence also $u \longrightarrow (e_i, x)$ in $H_{\mathcal{BS}(G)}$.

Again the argument is similar for a solid edge $u \longrightarrow (e_i, x)$ in $H_{\mathcal{BS}(K)}$, and for the edges $u \longleftarrow (e_i, x)$. $\square$

We can now complete the Proof. We have $G = H \boxtimes K$. By induction, we can assume that $H_{\mathcal{BS}(H)} = Sdg(Split(H))$ and $H_{\mathcal{BS}(K)} = Sdg(Split(K))$. Using the notation of Claim 2, the SD graph $Sdg(Split(G))$ is, by its definition, equal to $Sdg(Split(H)) + Sdg(Split(K))$. Hence, by Claim 2 and these equalities following from induction, it is isomorphic to $H_{\mathcal{BS}(G)}$. This completes the proof. $\square$

Theorem 4.21. *There exists an MS transduction that associates with a linearly ordered strongly connected graph the SD graph representing its canonical split decomposition.*

Proof. By Proposition 4.18, we have an MS transduction associating with $(G, \preccurlyeq)$ the structure $\langle V_G \cup N_T, edg_G, edg_T, box_T \rangle$ where T is the tree of the canonical decomposition, i.e, $T = T_{\mathcal{BS}(G)}$.

The next task is to specify the pairs (e, x) for the edges e of T and their nodes x as pairs (u, i) for u in $V_G \cup N_T$ and integers i in a fixed finite set. By using the ordering of V_G one can select the leaf r of T which contains a smallest vertex of G. We make T into a directed tree with root r. This orientation is MS definable. For an edge e of T, directed, say : $x \longrightarrow y$, we can define (e, x) as the pair $(y, 1)$ and (e, y) as the pair $(y, 2)$. Since T is a directed tree, each edge is specified in a unique way by its target. Hence, the vertex y refers to a single edge e.

Hence the set of vertices of $H_{\mathcal{BS}(G)}$ is defined as $V_G \times \{1\} \cup (N_T - \{r\}) \times \{1, 2\}$. The conditions defining the edges of the graph $H_{\mathcal{BS}(G)}$ are straightforward to express in MS logic, provided for each edge of T one can determine the corresponding good split. This is possible using the relation box_T. $\square$

Hence we have proved that the split decomposition of a strongly connected graph is definable by an MS transduction from the graph and a linear order of its vertices. It follows from Proposition A.1.1 (in Appendix 1) that a property of graphs expressed as an MS property of their prime components and/or of the underlying trees of their split decompositions is an order-invariant MS property.

5. Conclusion

In this article, we have applied Monadic Second-Order logic to the graph decompositions which follow the pattern of modular decomposition and to those defined in the framework of Cunnigham and Edmonds [18]. We have established general definability results in Monadic Second-Order logic, and we have applied them to the canonical decompositions of 2-connected graphs. We have obtained as new results a logical expression of Whitney's 2-isomorphism Theorem and the definability in Monadic Second-Order logic of the split decomposition of Cunnigham [17]. The article [6] applies this result to *circle graphs* studied in the framework of Monadic Second-order logic. This application is presented in the Introduction.

Here are some open questions (a few others are presented also in the main text).

Question 1: The *split decomposition* works well for undirected graphs and for strongly connected directed graphs, because these graphs have canonical decompositions. What about *connected directed graphs* ? The strongly connected components of a graph G form a *directed acyclic graph D*. Directed acyclic graphs have unique modular decompositions. However, it is not clear how to combine the modular decomposition of D and the split decompositions of the strongly connected components of G in order to obtain a notion of

canonical decomposition subsuming these cases. Although directed graphs have no *canonical* split decomposition, it may be useful to construct non canonical ones for algorithmic purposes or for investigations on the structure of graphs.

Question 2: Our logical formalization of decompositions, based on families of sets and on families of bipartitions can be applied to hypergraphs (along the lines of [4]), to k-structures which are also hypergraphs (see [20]), to matroids (the MS logic of matroids has been studied by Hliněny [24]). These applications should be developped.

Question 3: Another topic for future research is the extension of split decomposition to countable graphs, generalizing what is done in [13] for modular decomposition.

Acknowledgement

References

[1] M. Benedikt and L. Segoufin. Towards a characterization of order-invariant queries over tame structures. In *Computer Science Logic 2005*, volume 3634 of *Oxford, Lec. Notes Comput. Sci.* **3634**, pages 276–291. S-V, 2005.

[2] A. Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorica*, 7:243–254, 1987.

[3] C. Capelle. Block decomposition of inheritance hierarchies. In R. Möhring, editor, *Proceedings of WG'97*, volume 1335 of *LNCS*, pages 118–131. S-V, 1997.

[4] M. Chein, M. Habib, and M. Maurer. Partitive hypergraphs. *Discrete mathematics*, 37:35–50, 1981.

[5] S. Cicerone and G. Stefano. On the extension of bipartite to parity graphs. *Discrete Appl. Math.*, 95:181–195, 1999.

[6] B. Courcelle. submitted.

[7] B. Courcelle. Monadic second-order graph transductions: A survey. *Theoret. Comput. Sci.*, 126:53–75, 1994.

[8] B. Courcelle. The monadic second-order logic of graphs VIII: Orientations. *Ann. Pure Appl. Logic*, 72:103–143, 1995.

[9] B. Courcelle. The monadic second-order logic of graphs X: Linear orderings. *Theoret. Comput. Sci.*, 160:87–143, 1996.

[10] B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of graph grammars and computing by graph transformations*, volume 1: Foundations, pages 313–400. World Scientific, 1997.

[11] B. Courcelle. The monadic second-order logic of graphs XI Hierarchical decompositions of connected graphs. *Theoret. Comput. Sci.*, 224:35–58, 1999.

[12] B. Courcelle. The monadic second-order logic of graphs XV: On a Conjecture by D. Seese. *J. Appl. Logic*, 4:79–114, 2006.

[13] B. Courcelle and C. Delhommé. The modular decomposition of countable graphs: Constructions in Monadic Second-Order Logic. In *Computer Science Logic 2005*, volume 3634 of *Oxford, Lec. Notes Comput. Sci.*, pages 325–338. S-V, 2005.

[14] B. Courcelle, J. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computer Systems*, 33:125–150, 2000.

[15] B. Courcelle and S. Olariu. Upper bounds to the clique-width of graphs. *Discrete Appl. Math.*, 101:77–114, 2000.

[16] B. Courcelle and S. Oum. Vertex-minors, monadic second-order logic and a conjecture by Seese. To appear in J. of Combinatorial Theory B.

[17] W. Cunnigham. Decomposition of directed graphs. *SIAM J. Algebraic Discrete Methods*, 3:214–228, 1982.

[18] W. Cunnigham and J. Edmonds. A combinatorial decomposition theory. *Canad. J. Math*, 32:734–765, 1980.

[19] F. de Montgolfier. *Décomposition modulaire des graphes, Théorie, extensions et algorithmes*. PhD thesis, Montpellier 2 University, 2003.

[20] A. Ehrenfeucht and R. McConnell. A k-structure generalization of the theory of 2-structures. *Theoretical Computer Science*, 132:209–227, 1994.

[21] C. Gavoille and C. Paul. Distance labeling scheme and split decomposition. *Discrete Mathematics*, 273:115–130, 2003.

[22] M. Habib. *Substitution des structures combinatoires, théorie et algorithmes*. PhD thesis, Université Paris-6, 1981.

[23] M. Habib, M. Huchard, and J. Spinrad. A linear algorithm to decompose inheritance graphs into modules. *Algorithmica*, 13:573–591, 1995.

[24] P. Hliněny. On matroid properties definable in the MSO logic. In *Mathematical Foundations of Computer Science 2003*, volume 2747 of *LNCS*, pages 470–479. S-V, 2003.

[25] J. Hopcroft and R. Tarjan. Isomorphism of planar graphs. In *Complexity of computer computations*. Plenum Press, New York, 1972.

[26] R. Möhring and F. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. *Ann. Discrete Math.*, 19:257–356, 1984.

[27] S. Oum and P. Seymour. Appoximating clique-width and branch-width. 2004, To appear in J. of Combinatorial Theory B.

[28] J. Oxley. *Matroid theory*. Oxford University Press, 1992.

[29] J. Spinrad. *Efficient graph representations*, volume 19 of *Fields Institute Monographs*. A.M.S., Providence, 2003.

[30] K. Truemper. On Whitney's 2-isomorphism theorem for graphs. *J. Graph Theory*, 4:43–49, 1980.

[31] W. Tutte. *Connectivity in graphs*. University of Toronto Press, 1966.

[32] N. White. *Theory of matroids*. Cambridge University Press, 1986.

APPENDIX 1: MONADIC SECOND-ORDER LOGIC

We review Monadic Second-Order (MS) logic and transformations of structures expressed in this language, called *MS transductions*. The reader is refered to the book chapter [10], or to the preliminary sections of the articles [7, 9, 12] for more detailed expositions. However all necessary definitions are given in full in the present section.

Relational structures and monadic second-order logic. Let $R = \{A, B, C, \dots\}$ be a finite set of *relation symbols* each of them given with a nonnegative integer $\rho(A)$ called its *arity*. We denote by $\mathcal{STR}(R)$ the set of *finite R-structures* $S = \langle D_S, (A_S)_{A \in R} \rangle$ where $A_S \subseteq D_S^{\rho(A)}$ if $A \in R$ is a relation symbol. If R consist of relation symbols of arity one or two, then we say that the structures in $\mathcal{STR}(R)$ are *binary*.

A simple graph G can be defined as an $\{edg\}$-structure $G = \langle V_G, edg_G \rangle$ where V_G is the set of vertices of G and $edg_G \subseteq V_G \times V_G$ is a binary relation representing the edges. For undirected graphs, the relation edg_G is symmetric. If in addition we need vertex labels, we will represent them by unary relations. Binary structures can be seen as vertex- and edge-labelled graphs. If we have several binary relations say A, B, C, the corresponding graphs have edges of types A, B, C.

We recall that *Monadic Second-order logic* (*MS logic* for short) is the extension of *First-Order logic* (*FO logic* for short) by variables denoting subsets of the domains of the

considered structures, and new atomic formulas of the form $x \in X$ expressing the membership of x in a set X. (Uppercase letters will denote set variables, lowercase letters will denote first-order variables).

We denote by $FO(R, W)$ (resp. by $MS(R, W)$) the set of *First-order* (resp. *Monadic Second-order*) formulas written with the set R of relation symbols and having their free variables in a set W consisting of *first-order as well as of set variables*. Hence, we allow first-order formulas with free set variables and written with atomic formulas of the form $x \in X$. In first-order formulas, only first-order variables can be quantified.

As a typical and useful example of MS formula, we give a formula with free variables x and y expressing that (x, y) belongs to the reflexive and transitive closure of a binary relation A:

$$\forall X(x \in X \wedge \forall u, v[(u \in X \wedge A(u,v)) \implies v \in X] \implies y \in X) \ .$$

If the relation A is not given in the structure but defined by an MS formula, then one replaces $A(u, v)$ by this formula with appropriate substitutions of variables.

A *monadic second-order (MS) property* of the structures S of a class $\mathcal{C} \subseteq \mathcal{STR}(R)$ is a property $\mathcal{P}$ such that for $S \in \mathcal{C}$:

$$\mathcal{P}(S) \quad \text{holds if and only if} \quad S \vDash \varphi \ ,$$

for some fixed formula φ in $MS(R, \varnothing)$. Let $\leq$ be a binary relation symbol not in R. A formula φ in $MS(R \cup \{\leq\}, \varnothing)$ is *order-invariant on a class* $\mathcal{C}$, if for every $S \in \mathcal{C}$, for every two linear orders $\preccurlyeq$ and $\preccurlyeq'$ on the domain D_S

$$(S, \preccurlyeq) \vDash \varphi \quad \text{if and only if} \quad (S, \preccurlyeq') \vDash \varphi \ ,$$

where $\preccurlyeq$ and $\preccurlyeq'$ interpret $\leq$. We say that $\mathcal{P}$ is an *order-invariant* MS property of the structures of a class $\mathcal{C} \subseteq \mathcal{STR}(R)$ if and only if

$$\mathcal{P}(S) \quad \text{holds if and only if} \quad (S, \preccurlyeq) \vDash \varphi \quad \text{for some linear order} \ \preccurlyeq \text{on } D_S \ ,$$

where φ is a fixed order-invariant MS formula. Order-invariant MS properties are investigated in [1, 9]. A difficulty with this definition is that the set of order-invariant MS formulas is undecidable. However, we will use formulas that are order-invariant by construction.

Monadic Second-order transductions. We will also use FO and MS formulas to define certain graph transformations. As in Language Theory, a binary relation $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$ where $\mathcal{A}$ and $\mathcal{B}$ are sets of relational structures will be called a *transduction*: $\mathcal{A} \to \mathcal{B}$.

An *MS transduction* is a transduction specified by MS formulas. It transforms a structure S, given with an n-tuple of subsets of its domain called the *parameters*, into a structure T, the domain of which is a subset of $D_S \times \{1, \ldots, k\}$. Furthermore, each such transduction, has an associated *backwards translation*, a mapping that transforms effectively every MS formula φ relative to T, possibly with free variables, into one, say $\varphi^{\#}$, relative to S having free variables corresponding to those of φ (k times as many actually) together with those denoting the parameters. This new formula expresses in S the property of T defined by φ. We now give some details. More can be found in [7, 10].

We let R and Q be two finite sets of relation symbols. Let W be a finite set of set variables, called *parameters*. A (Q, R)-*definition scheme* is a tuple of formulas of the form

$$\Delta = (\varphi, \psi_1, \cdots, \psi_k, (\theta_w)_{w \in Q*k}) \quad \text{where } k > 0 \text{ and } Q*k := \{(q, \vec{j}) \mid q \in Q, \vec{j} \in [k]^{\rho(q)}\} \ ,$$

$$\varphi \in MS(R, W), \psi_i \in MS(R, W \cup \{x_1\}) \quad \text{for } i = 1, \cdots, k, \text{ and}$$

$$\theta_w \in MS(R, W \cup \{x_1, \cdots, x_{\rho(q)}\}) \quad \text{for } w = (q, \vec{j}) \in Q^* k .$$

These formulas are intended to define a structure T in $\mathcal{STR}(Q)$ from a structure S in $\mathcal{STR}(R)$. Let $S \in \mathcal{STR}(R)$, let γ be a W-assignment in S. A Q-structure T with domain $D_T \subseteq D_S \times [k]$ is *defined in* (S, γ) by Δ if

(i) $(S, \gamma) \models \varphi$,
(ii) $D_T = \{(d, i) \mid d \in D_S, i \in [k], (S, \gamma, d) \models \psi_i\}$,
(iii) for each q in Q

$$q_T = \{((d_1, i_1), \cdots, (d_t, i_t)) \in D_T^t \mid (S, \gamma, d_1, \cdots, d_t) \models \theta_{(q, \vec{j})}\} ,$$

where $\vec{j} = (i_1, \cdots, i_t)$ and $t = \rho(q)$.

The notation $S \models \psi$ means that the logical formula ψ holds true in the structure S. By $(S, \gamma, d_1, \cdots, d_t) \models \theta_{(q, \vec{j})}$, we mean $(S, \gamma') \models \theta_{(q, \vec{j})}$, where γ' is the assignment extending γ, such that $\gamma'(x_i) = d_i$ for all $i = 1, \cdots, t$; a similar convention is used for $(S, \gamma, d) \models \psi_i$).

Since T is associated in a unique way with S, γ and Δ whenever it is defined, i.e., whenever $(S, \gamma) \models \varphi$, we can use the functional notation $def_\Delta(S, \gamma)$ for T. The *transduction defined by* Δ is the binary relation

$$\mathcal{D}_\Delta := \{(S, T) \mid T = def_\Delta(S, \gamma) \text{ for some } W\text{-assignment } \gamma \text{ in } S\} .$$

Hence $\mathcal{D}_\Delta \subseteq \mathcal{STR}(R) \times \mathcal{STR}(Q)$. A transduction $f \subseteq \mathcal{STR}(R) \times \mathcal{STR}(Q)$ is an *MS transduction* if it is equal, up to isomorphism of structures, to $\mathcal{D}_\Delta$ for some (Q, R)-definition scheme Δ.

An MS-transduction is defined as a binary relation. Hence it can be seen as a "nondeterministic" partial function associating with an R-structure one or more Q-structures. However, it is not really nondeterministic because the different outputs come from different choices of parameters. In the case where $W = \emptyset$, we say that the transduction is *parameterless* ; it defines a partial function. It may also happen that different choices of parameters yield isomorphic output structures. This is the case in the example of edge contraction detailed below.

We will refer to the integer k by saying that Δ and $\mathcal{D}_\Delta$ are *k-copying*; if $k = 1$ we will say that they are *noncopying*. A noncopying definition scheme can be written more simply: $\Delta = (\varphi, \psi, (\theta_q)_{q \in Q})$. We will say that an MS transduction is *domain extending*, if the formula ψ_1 of its definition scheme Δ is the Boolean constant $True$. In this case, if $T = def_\Delta(S, \gamma)$, then D_T contains $D_S \times \{1\}$, an isomorphic copy of D_S. This transduction defines the domain of T as an extension of that of S. If in the definition scheme Δ we only use FO formulas, then we will say that $\mathcal{D}_\Delta$ is an *FO transduction*.

Example. *Edge contraction.* We consider a graph G with two types of edges, the ordinary edges and the ε-edges. It is represented by a structure $\langle V_G, edg_G, \varepsilon - edg_G \rangle$ where the binary relation $\varepsilon - edg_G$ represents the ε-edges. We want to define from G the graph H obtained by the contraction of all ε-edges.

It is formally defined as $\langle V_H, edg_H \rangle$ where $V_H = V_G / \sim$, $\sim$ is the equivalence relation such that $x \sim y$ if and only if x and y are linked by an undirected path made of ε-edges, and $edg_H([u], [v])$ holds if and only if $x \in [u]$, $y \in [v]$ for some (x, y) in edg_G ($[u]$ denotes the equivalence class of u). The MS formula $\xi(x, y)$ defined as

$$\forall X[(x \in X \wedge \forall u, v\{u \in X \wedge (\varepsilon - edg(u, v) \vee \varepsilon - edg(v, u)) \Longrightarrow v \in X\}) \Longrightarrow y \in X]$$

expresses $x \sim y$. For defining V_H we must select a set containing one and only one vertex of each equivalence class. This can be done with a set variable Y that will be a parameter of the MS transduction, satisfying the formula $\varphi(Y)$ defined as $\forall x \exists ! y [y \in Y \wedge \xi(x, y)]$.

Edge contraction can be defined by the transduction with noncopying definition scheme $\Delta = (\varphi, \psi, \theta_{edg})$ where $\psi(Y, x)$ is $x \in Y$ and $\theta_{edg}(Y, x, y)$ is $\exists u, v[x \in Y \wedge y \in Y \wedge edg(u, v) \wedge \xi(x, u) \wedge \xi(y, v)]$.

Notice that the structures associated with all values of the parameter Y satisfying $\varphi(Y)$ are isomorphic. They only differ regarding the concrete subsets Y of V_G used as sets of vertices of H.

Lemma A.1.1. Let $\tau : \mathcal{STR}(R) \longrightarrow \mathcal{STR}(Q)$ be an MS (or FO) transduction. Let $\leq$ be a binary relation symbol not in $R \cup Q$. One can transform τ into an MS (or FO) transduction $\tau' : \mathcal{STR}(R \cup \{\leq\}) \longrightarrow \mathcal{STR}(Q \cup \{\leq\})$ such that, for every S in $\mathcal{STR}(R)$ and every linear order $\preceq$ on its domain, $\tau'(S, \preceq) = (\tau(S), \preceq')$ where $\preceq'$ is a linear order on the domain of $\tau(S)$.

Proof. Let τ be k-copying. For $w = (\leq, \vec{j}) \in \{\leq\}^* k$ it is easy to define FO formulas θ_w belonging to $MS(R \cup \{\leq\}, W \cup \{x_1, x_2\})$ such that, in $\tau'(S, \preceq)$

$$(d_1, i) \preceq' (d_2, j) \quad \text{if and only if either } d_1 \prec d_2 \text{ or } (d_1 = d_2 \text{ and } i \leq j) .$$

It is clear that $\preceq'$ is a linear order on the domain of $\tau(S)$ if $\preceq$ is one on S. $\qquad \square$

The fundamental property of MS transductions. The following proposition says that if $T = def_\Delta(S, \gamma)$, then the monadic second-order properties of T can be expressed as monadic second-order properties of (S, γ). The usefulness of definable transductions is based on this proposition.

Let $\Delta = (\varphi, \psi_1, \cdots, \psi_k, (\theta_w)_{w \in Q^* k})$ be a (Q, R)-definition scheme, written with a set of parameters W. Let V be a set of set variables disjoint from W. For every variable X in V, for every $i = 1, \cdots, k$, we let X_i be a new variable. We let $V' := \{X_i / X \in V, i = 1, \cdots, k\}$. Let S be a structure in $\mathcal{STR}(R)$ with domain D. For every mapping $\eta : V' \longrightarrow \mathcal{P}(D)$, we let $\eta^k : V \longrightarrow \mathcal{P}(D \times [k])$ be defined by $\eta^k(X) = \eta(X_1) \times \{1\} \cup \cdots \cup \eta(X_k) \times \{k\}$. With this notation we can state

Proposition A.1.2. *For every formula β in $MS(Q, V)$ one can construct a formula $\beta^\#$ in $MS(R, V' \cup W)$ such that, for every S in $\mathcal{STR}(R)$, for every assignment $\gamma : W \longrightarrow S$, for every assignment $\eta : V' \longrightarrow S$ we have*

$$(S, \eta \cup \gamma) \models \beta^\# \quad \text{if and only if} \quad def_\Delta(S, \gamma) \text{ is defined,}$$
$$\eta^k \text{ is a } V\text{-assignment in } def_\Delta(S, \gamma),$$
$$\text{and} \quad (def_\Delta(S, \gamma), \eta^k) \models \beta . \qquad \square$$

If the definition scheme and the formula β are FO the formula $\beta^\#$ is also FO. Note that, even if $T = def_\Delta(S, \gamma)$ is well-defined, the mapping η^k is not necessarily a V-assignment in T, because $\eta^k(X)$ may not be a subset of the domain of T which is a possibly proper subset of $D_S \times \{1, \ldots, k\}$. We call $\beta^\#$ the *backwards translation* of β relative to the transduction def_Δ.

The composition of two transductions is defined as the composition of the corresponding binary relations. If they are both partial functions, then one obtains the composition of

these functions. The composition of two domain extending MS (or FO) transductions is domain extending.

Proposition A.1.3.
(1) *The composition of two MS (or FO) transductions is an MS (or an FO) transduction.*
(2) *The inverse image of an MS-definable class of structures under an MS transduction is MS-definable. A similar statement holds with FO instead of MS.* □

Appendix 2: Clique-width

Clique-width is, like tree-width a graph complexity measure. It is defined and studied by Courcelle and Olariu in [15], and also in [10, 27]. Graphs are simple, directed or not, and loop-free.

Let C be a set of k labels. A *C-graph* is a graph G given with a total mapping from its vertices to C, denoted by lab_G. Hence G is defined as a triple (V_G, edg_G, lab_G). We call $lab_G(v)$ the *label* of a vertex v. The operations on C-graphs are the following ones

(i) For each $i \in C$, we define a constant **i** for denoting an isolated vertex labelled by i.
(ii) For $i, j \in C$ with $i \neq j$, we define a unary function $add_{i,j}$ such that

$$add_{i,j}(V_G, edg_G, lab_G) = (V_G, edg'_G, lab_G) \ ,$$

where edg'_G is edg_G augmented with the set of pairs (u, v) such that $lab_G(u) = i$ and $lab_G(v) = j$.

In order to add undirected edges, we take: $add_{i,j}(add_{j,i}(V_G, edg_G, lab_G))$.
(iii) We let also $ren_{i \to j}$ be the unary function such that

$$ren_{i \to j}(V_G, edg_G, lab_G) = (V_G, edg_G, lab'_G) \ ,$$

where $lab'_G(v) = j$ if $lab_G(v) = i$, and $lab'_G(v) = lab_G(v)$, otherwise. This mapping renames into j every vertex label i.
(iv) Finally, we use the binary operation $\oplus$ that makes the union of disjoint copies of its arguments. Hence $G \oplus G \neq G$ and its size is twice that of G.

A well-formed expression t over these symbols will be called a *C-expression*, or a *k-expression* if we are only concerned with the size k of C. Its *value* is a C-graph $G = val(t)$. The set of vertices of $val(t)$ is (or can be defined as) the set of occurrences of the constant symbols in t. However, we will also consider that an expression t designates any graph isomorphic to $val(t)$. The context specifies whether we consider concrete graphs or graphs up to isomorphism.

A graph is considered as a graph all vertices of which are labelled in the same way. The *clique-width* of a graph G, denoted by $cwd(G)$ is the minimal k such that $G = val(t)$ for some k-expression t. A graph with at least one edge has clique-width at least 2. The graphs K_n, S_{n-1} have clique-width 2, for $n \geq 3$.

If we need to define graphs with vertex labels from a set L, then we use constant symbols $\mathbf{i}_a$ for i in C and a in L. The labels from L are not changed, and do not affect the other operations. The clique-width of a graph does not depend on the possible labelling of its vertices. By contrast, it depends strongly on edge directions. Cliques and transitive tournaments have clique-width 2 but tournaments have unbounded clique-width ([8]). To build a graph with labelled edges we use the operation $add_{a,i,j}$ to add edges labelled by a from the vertices labelled by i to those labelled by j.

Lemma A.2.1. [15] (1) *The clique-width of a graph is equal to the maximum clique-width of its induced subgraphs.*
(2) *The clique-width of $G[H/u]$ is equal to the maximum of $cwd(G)$ and $cwd(H)$.*
(3) *The clique-width of a graph is equal to $Max\{m, 2\}$, where m is the maximum clique-width of the prime graphs of its modular decomposition.* □

Lemma A.2.2. [10] *For every MS transduction τ from graphs to graphs there exists a fonction f such that $T \in \tau(S)$ implies $cwd(T) \leq f(cwd(S))$.* □

Lemma A.2.3. *Let G be a graph let u be a vertex of indegree 0, and v be a vertex of outdegree 0. Let G' be obtained from G by fusing u and v. Then $cwd(G') \leq 4\,cwd(G)$.*

Proof. Let $k = cwd(G)$ and E be a $\{1, \ldots, k\}$-expression for G, considered as a $\{1\}$-graph. For every x in $V_G - \{u, v\}$, we let its *type* be 1 if $u \longrightarrow x$ and $x \longrightarrow v$, be 2 if $u \longrightarrow x$ and $x \longrightarrow v$ does not hold, be 3 if $x \longrightarrow v$ and $u \longrightarrow x$ does not hold, and 0 otherwise.

We let H be the graph $G[V_G - \{u, v\}]$ where every vertex has label $(1, i)$ (instead of 1) and i is its type. We let $C = \{1, \ldots, k\} \times \{0, 1, 2, 3\}$. From E, by deleting the constants which define u and v, and by modifying the graph operations so that every label a of a vertex is replaced by (a, i) where i is its type, one can construct a C-expression E' defining H. Let p be a label, e.g., $(2,0)$, which does not label any vertex of H. The graph G' with all its vertices labelled by p is the value of

$$ren_{(1,0) \to p} \circ ren_{(1,1) \to p} \circ ren_{(1,2) \to p} \circ ren_{(1,3) \to p} \circ add_{p,(1,1)} \circ add_{p,(1,2)} \circ add_{(1,1),p} \circ add_{(1,3),p}$$

at $E' \oplus \mathbf{p}$. Hence G' has clique-width at most $4k$. □