The modular decomposition of countable graphs : Definition and construction in monadic second-order logic

Bruno Courcelle, Christian Delhommé

THE MODULAR DECOMPOSITION OF COUNTABLE GRAPHS. DEFINITION AND CONSTRUCTION IN MONADIC SECOND-ORDER LOGIC

BRUNO COURCELLE AND CHRISTIAN DELHOMMÉ

Abstract. We consider the notion of modular decomposition for countable graphs. The modular decomposition of a graph given with an enumeration of its set of vertices, can be defined by formulas of Monadic Second-Order logic. Another result is the definition of a representation of modular decompositions by low degree relational structures. Such relational structures can also be defined in the considered graph by monadic second-order formulas.

1. Introduction

Finite and infinite trees and graphs can be studied from different points of view : as *binary relations*, as generalized words to which the methods of *formal language theory* can be applied or as *algorithmic objects*. These perspectives are developped respectively in the book on *theory of relations* by Fraïssé [29], in the book chapter by Courcelle [14] for the extension of language theory to finite graphs, in many books on graph algorithms, among which we quote the one by Downey and Fellows on *parametrized complexity* [25]. The study of infinite trees and graphs is motivated by the need of *formalizing the semantics* of programs and of processes in order to verify them [9, 12, 20], and by research on combinatorial group theory [39, 43]. However it is an interesting chapter of graph theory on its own, see for instance the book by Diestel [24].

Algorithms and decidability questions are meaningful for finite graphs and for infinite graphs *described in finitary ways*. Many types of finite descriptions of countable graphs have been proposed : by pushdown automata and more generally by languages [42, 5], by equation systems [10, 1], by automatic structures [33, 38, 3], by formulas of monadic second-order logic [10, 4]. The doctoral dissertation of T. Colcombet [6] presents a detailed comparison of these different specification methods.

Two importants tools for all these studies are *graph decompositions* and *monadic second-order logic*. Graphs, finite as well as infinite, can be decomposed in several ways. By a *decomposition* of a graph, we mean a construction of this graph that uses a specified set of basic graphs and graph composition operations. For instance a graph is obtained from its biconnected components by one-vertex gluings. Another

Date: November 8, 2007.

Key words and phrases. Modular decomposition, Tree, Linear order, Monadic second-order logic, Monadic second order transduction.

Acknowledgement : This work has been initiated during a stay of B. Courcelle in the ERMIT group, in 2004, supported by the University of La Réunion. It has also been supported by the project GRAAL of the "Agence Nationale pour la Recherche".

example is the very well-known notion of *tree-decomposition*. The global structure of a decomposition is always a tree.

Graph decompositions are very useful for the understanding of the structure of certain types of graphs and also for algorithmic purposes. In particular, the notions of tree-decomposition and of *tree-width* are fundamental for the construction of *fixed parameter tractable* algorithms for many problems, in particular for NP-complete ones. Many such problems have polynomial time algorithms on graphs of tree-width at most k. This use of tree-decompositions is developped in the book [25]. *Clique-width* is another graph complexity measure on graphs, based on the expression of a graph from one-vertex graphs by means of certain graph operations, and that is interesting for algorithmic purposes ([21]). For graphs of bounded tree-width or of bounded clique-width, efficient algorithms can be obtained for problems expressed in *monadic second-order logic*.

The present article investigates the *modular decomposition of infinite (mainly countable) graphs*. The notion of modular decomposition of a finite graph has been studied extensively in many articles, and under various names. Mörhing and Radermacher give in [41] a survey of this frequently rediscovered notion. The *composition operation* underlying its definition is the *substitution* of a graph $\mathbf{H}$ for a vertex v in a graph $\mathbf{G}$. In the resulting graph, denoted by $\mathbf{G}[\mathbf{H}/v]$, the vertex v is replaced by $\mathbf{H}$, and all its neighbours in $\mathbf{G}$ are linked to all vertices of $\mathbf{H}$. A subgraph $\mathbf{H}$ of a graph $\mathbf{K}$ is a *module* if $\mathbf{K}$ is of the form $\mathbf{G}[\mathbf{H}/v]$. Among those are distinguished the *strong* modules, namely those modules that are comparable for inclusion with every module that they meet. The strong modules form a finite tree where the ancestor relation is inclusion ($X \subseteq Y$ if and only if Y is an ancestor of X). This tree is called its *modular decomposition*. It corresponds to a *canonical* expression of the graph in terms of (nested) substitutions. Furthermore, this decomposition can be constructed from the given graph by monadic second-order formulas, as proved in [13]. This result enriches the logical tool box and helps to express graph properties in monadic second-order logic, which yields, ultimately, polynomial algorithms and decidability results (see [42, 10, 14] for decidability results based on monadic second-order logic).

The notion of modular decomposition is essential, not only for algorithmic purposes, but also for establishing structural properties of graphs and related objects, in particular of partial orders and their *comparability graphs*. For instance, one can compute the number of transitive orientations of a finite comparability graph from its modular decomposition. Motivated by the investigation of comparability graphs, and in particular for proving that the dimension of a partial order depends only on its comparability graph, Kelly [37] reviews Gallai's fundamental analysis of the properties of modules of finite undirected graphs [30] and extends it to infinite ones. The strong modules of an infinite graph are pairwise disjoint or comparable for inclusion, but they do not form a tree in the usual sense where a tree is defined as a connected and directed graph without circuits such that every vertex is reachable by a unique directed path from the *root* (the unique vertex of indegree 0). They form a tree provided one defines a *tree* as a partial order such that the set of elements larger than any element is linearly ordered, which we will do in this article. Such trees may have no root. The ordered set $\mathbb{Q}$ of rational numbers, like any linear order, is a tree in this sense. It has no root, and no element has a father because no rational number has a successor.

Our aims are to define and to study modular decompositions of countable graphs, to represent them by relational structures, and to use *Monadic Second-Order logic (MS logic in short)* to construct from a graph certain representations of its modular decomposition by relational structures. MS logic has already been used for such constructions by Courcelle in [13], in the case of finite graphs, and our constructions build upon those of this article, with the necessity of new features for handling infinite graphs.

For defining the modular decomposition of a countable graph, we do not take all strong modules, but only some of them. Doing so we obtain a countable object associated with a countable graph. By defining the modular decomposition as the tree of all strong modules, we would obtain in certain cases an uncountable tree associated with a countable graph. This would not be satisfactory because we consider that the modular decomposition must be a *synthetic representation* of a graph : so it is not acceptable that it is of larger cardinality than the considered graph. There are two key tools for our study of modular decomposition of infinite graphs. On the one hand, the notion of *robust* module (already considered in [37]), defined as the intersection of all strong modules containing two vertices. And on the other hand, the *characterization of graphs with no non-trivial strong modules*[1] ; the finite and undirected case appears in [30], and the extension to infinite directed graphs, for which we provide a somewhat simpler proof, is in Ehrenfeucht and al. [27].

Another concern is to describe *dense* graphs, *i.e.* graphs having "lots of edges", by relational structures, actually vertex- and edge-labelled graphs, which are as sparse as possible. A finite partial order can be represented by its Hasse diagram, and this representation can be used for expressing its properties in monadic second-order logic. This is no longer the case of infinite partial orders. Consider for instance the ordered set $\mathbb{Q}$. However, it can be defined as a certain ordering of the nodes of the complete infinite binary tree (here "tree" is taken in the usual sense of computer science). This observation extends to all countable linearly ordered sets. Furthermore, first-order formulas using an auxiliary enumeration of the considered linear order A, *i.e.* an auxiliary ordering of A isomorphic to the ordinal ω can define the representing binary tree (these formulas do not depend on A). It follows that this representation of linear orders by binary trees fits with the expression of their properties by monadic second-order formulas. We use this fact to represent the modular decomposition of a countable graph $\mathbf{G}$ by a countable graph of maximum degree $m+3$ where m is the least upper bound of the degrees of the *prime subgraphs* of $\mathbf{G}$. It may happen that m is finite, even if $\mathbf{G}$ has vertices of infinite degree. This is the case for cographs.

This article is an expanded version of [19]. It is organized in nine sections :

(1) Introduction.
(2) Review of the modular decomposition for finite graphs.
(3) Basics about trees.
(4) Strong and robust modules.
(5) The modular decomposition.
(6) Construction of the modular decomposition in monadic second order logic.

[1]Those are the prime graphs, the undirected complete or edge-free graphs, and the linear orders.

(7) Representation of modular decompositions by low degree relational structures.
(8) Concluding remarks.
(9) Appendix : Monadic second order logic and monadic second order transductions.

2. Prologue : Graph substitutions and the Modular decomposition of finite graphs

This section is a quick review of the notion of modular decomposition for finite graphs and a presentation of our view of this notion, in the perspective of its extension to countable graphs.

Graphs are loop-free without multiple edges. A graph $\mathbf{G}$ is handled as a relational structure $\langle V_{\mathbf{G}}, edg_{\mathbf{G}}\rangle$. Its domain is the vertex set $V_{\mathbf{G}}$ and $edg_{\mathbf{G}}$ is a binary relation such that $edg_{\mathbf{G}}(x,x)$ never holds. We write also $x \xrightarrow{\mathbf{G}} y$ if $edg_{\mathbf{G}}(x,y)$ holds. A graph is *undirected* if for every vertices x and y, $x \xrightarrow{\mathbf{G}} y$ implies $y \xrightarrow{\mathbf{G}} x$; it is *directed* if $x \xrightarrow{\mathbf{G}} y$ and $y \xrightarrow{\mathbf{G}} x$ never hold simultaneously. It is *total* if any two distinct vertices are linked by an edge. It is *complete* if it is total and undirected. We say that a graph is a *linear order* if for some strict linear ordering $<$ on $V_{\mathbf{G}}$, $x \xrightarrow{\mathbf{G}} y$ if and only if $x < y$.

We denote by $\mathbf{G}[X]$ the induced subgraph of $\mathbf{G}$ with vertex set $X \subseteq V_{\mathbf{G}}$. A graph $\mathbf{H}$ *embeds* into a graph $\mathbf{G}$ if $\mathbf{H}$ is isomorphic to an induced subgraph of $\mathbf{G}$.

2.1. Graph substitution and modules.

Definition 2.1. (*Graph substitution.*) Let $\mathbf{K}$ be a graph and let $(\mathbf{H}_v)_{v\in V_{\mathbf{K}}}$ be a family of pairwise disjoint graphs (they have no vertex in common). We denote by $\mathbf{K}[\mathbf{H}_v/v; v \in V_{\mathbf{K}}]$ the graph $\mathbf{G}$ resulting from the simultaneous *substitution* of $\mathbf{H}_v$ for $v \in V_{\mathbf{K}}$. It is defined as follows

- $V_{\mathbf{G}} = \cup\{V_{\mathbf{H}_v} : v \in V_{\mathbf{K}}\}$
- $u \xrightarrow{\mathbf{G}} u'$ if and only if
  - either $u \xrightarrow{\mathbf{H}_v} u'$ (with $u, u' \in V_{\mathbf{H}_v}$) for some v,
  - or there are two distinct $v, v' \in V_{\mathbf{K}}$ such that $v \neq v'$, $u \in V_{\mathbf{H}_v}$, $u' \in V_{\mathbf{H}_{v'}}$ and $v \xrightarrow{\mathbf{K}} v'$.

Notation 2.1. If, in that definition, $\mathbf{K}$ is an edge-free graph we write $\mathbf{G} = \oplus_{v\in V_{\mathbf{K}}}\mathbf{H}_v$ and we say that $\mathbf{G}$ is the *free sum* of the family $(\mathbf{H}_v)_{v\in V_{\mathbf{K}}}$.

If $\mathbf{K}$ is a complete graph we write $\mathbf{G} = \otimes_{v\in V_{\mathbf{K}}}\mathbf{H}_v$ and we say that $\mathbf{G}$ is the *complete sum* of the family $(\mathbf{H}_v)_{v\in V_{\mathbf{K}}}$.

If $\mathbf{K}$ is a linear order $(V_{\mathbf{K}}, <)$ (thus $v \xrightarrow{\mathbf{K}} v'$ if and only if $v < v'$), we write $\mathbf{G} = \overrightarrow{\otimes}_{v\in V_{\mathbf{K}}}\mathbf{H}_v$ and we say that $\mathbf{G}$ is the *linear sum* of the family $(\mathbf{H}_v)_{v\in V_{\mathbf{K}}}$ where the set of indices $V_{\mathbf{K}}$ is linearly ordered by $<$.

As binary operations, $\oplus$ and $\otimes$ are associative and commutative, and the operation $\overrightarrow{\otimes}$ is only associative.

Definition 2.2. (*Modular partition, module.*) In order to characterize the ways in which a graph $\mathbf{G}$ can be expressed as $\mathbf{K}[\mathbf{H}_v/v; v \in V_{\mathbf{K}}]$, one defines the notion of *modular partition* : it is a partition $\mathcal{C}$ of the vertex set $V_{\mathbf{G}}$ (the members of $\mathcal{C}$ are pairwise disjoint non-empty sets of vertices, of which the union is the vertex set) such that for any two distinct M and N in $\mathcal{C}$ and any $u, u' \in M$, $v, v' \in N$

$$u \xrightarrow{\mathbf{G}} v \text{ if and only if } u' \xrightarrow{\mathbf{G}} v'$$

Each member M of $\mathcal{C}$ is a *module*, *i.e.* a set of vertices such that for any $u \in V_{\mathbf{G}}\backslash M$ for any $v, v' \in M$

$$(u \xrightarrow{\mathbf{G}} v \text{ if and only if } u \xrightarrow{\mathbf{G}} v') \text{ and } (v \xrightarrow{\mathbf{G}} u \text{ if and only if } v' \xrightarrow{\mathbf{G}} u)$$

The sets $\varnothing$, $V_{\mathbf{G}}$ and $\{v\}$ for each $v \in V_{\mathbf{G}}$ are modules and are called the *trivial* modules.

If $\mathcal{C}$ is a modular partition, then the *quotient* is the graph $\mathbf{K} = \mathbf{G}/\mathcal{C}$ defined by :

(1) $V_{\mathbf{K}} = \mathcal{C}$
(2) $M \xrightarrow{\mathbf{K}} N$ if and only if for some u, v we have $u \in M$, $v \in N$ and $u \xrightarrow{\mathbf{G}} v$.

From the definitions it follows immediately :

Proposition 2.1. (1) *If* $\mathbf{G} = \mathbf{K}[\mathbf{H}_v/v; v \in V_{\mathbf{K}}]$, *then the sets* $V_{\mathbf{H}_v}$ *form a modular partition of* $\mathbf{G}$.
(2) *If* $\mathcal{C}$ *is a modular partition of* $\mathbf{G}$, *then* $\mathbf{G} = (\mathbf{G}/\mathcal{C})[\mathbf{H}[M]/M; M \in \mathcal{C}]$.

We may also consider as modular partition any indexed partition $\mathcal{C} = (V_i)_{i\in I}$ (in particular the V_i's are pairwise distinct) such that $\{V_i : i \in I\}$ is a modular partition in the sense above ; thus the set of vertices of $\mathbf{G}/\mathcal{C}$ is I, and $\mathbf{G} = (\mathbf{G}/\mathcal{C})[\mathbf{H}[V_i]/i; i \in I]$.

2.2. **Decompositions.** Let $\mathbf{G}$ be a graph expressed as $\mathbf{K}[\mathbf{H}_v/v; v \in V_{\mathbf{K}}]$. We say that this expression defines a decomposition of $\mathbf{G}$ into the graphs $\mathbf{H}_v$, $v \in V_{\mathbf{K}}$. The graphs $\mathbf{H}_v$ may themselves be decomposed into smaller graphs, which themselves may be decomposed similarly. Thus we obtain a notion of graph decomposition, that can be defined as follows.

Definition 2.3. (*Decompositions.*) Two sets A and B *meet* if $A \cap B \neq \varnothing$; they *overlap* if $A \cap B \neq \varnothing$, $A \backslash B \neq \varnothing$ and $B \backslash A \neq \varnothing$.

A *decomposition*[2] of a graph $\mathbf{G}$ is a family $\mathcal{D}$ of subsets of $V_{\mathbf{G}}$ such that

(1) $\varnothing \notin \mathcal{D}$, $V_{\mathbf{G}} \in \mathcal{D}$, $\{v\} \in \mathcal{D}$ for each $v \in V_{\mathbf{G}}$;
(2) no two members of $\mathcal{D}$ overlap ;
(3) for every $M \in \mathcal{D}$, letting $mp(M, \mathcal{D})$ denote the set of maximal proper subsets of M that belong to $\mathcal{D}$, if $mp(M, \mathcal{D})$ is non-empty then it forms a modular partition of $\mathbf{G}[M]$ denoted by $\pi(M)$ (or $\pi_{\mathcal{D}}(M)$ when $\mathcal{D}$ need to be specified).

Hence if $\mathbf{G} = \mathbf{K}[\mathbf{H}_v/v; v \in V_{\mathbf{K}}]$, the family

$$\mathcal{D} = \{V_{\mathbf{G}}\} \cup \{V_{\mathbf{H}_v} : v \in V_{\mathbf{K}}\} \cup \{\{v\} : v \in V_{\mathbf{G}}\}$$

is a decomposition of $\mathbf{G}$. If furthermore $\mathcal{D}_v$ is a decomposition of $\mathbf{H}_v$ for each $v \in V_{\mathbf{K}}$, then

$$\mathcal{D}' = \cup\{\mathcal{D}_v : v \in V_{\mathbf{K}}\} \cup \{V_{\mathbf{G}}\}$$

is a decomposition of $\mathbf{G}$ that *refines* $\mathcal{D}$, that is, such that $\mathcal{D} \subseteq \mathcal{D}'$.

Let us conversely assume that $\mathcal{D}$ is a decomposition of a graph $\mathbf{G}$. By the first two conditions, $(\mathcal{D}, \subseteq)$ is a tree, denoted by $Tree(\mathcal{D})$: its nodes are the elements of $\mathcal{D}$, $X \subsetneq Y$ if and only if Y is an ancestor of X, $V_{\mathbf{G}}$ is the root and the singletons $\{v\}$ are the leaves.

Because of the finiteness of $\mathcal{D}$, a set $mp(M, \mathcal{D})$ is not empty except when M is a leaf. It follows that for each $M \in \mathcal{D}$ that is not a leaf,

$$\mathbf{G}[M] = (\mathbf{G}[M]/\pi(M))[\mathbf{G}[N_1]/1, \ldots, \mathbf{G}[N_n]/n]$$

where N_1, ..., N_n are the sons of M in $Tree(\mathcal{D})$. In particular

$$\mathbf{G} = (\mathbf{G}/\pi(V_{\mathbf{G}}))[\mathbf{G}[N_1]/1, \ldots, \mathbf{G}[N_n]/n]$$

[2]From now on, the word "decomposition" will be used in the sense of this definition.

where $N_1, \ldots, N_n$ are the sons of the root.

Lemma 2.1. *Let $\mathcal{D}$ be a decomposition of a finite graph $\mathbf{G}$, let $N_1, \ldots, N_n$ be the sons of $V_{\mathbf{G}}$ in the tree of $\mathcal{D}$. Then for each i, $\{X \in \mathcal{D} : X \subseteq N_i\}$ is a decomposition of $\mathbf{G}[N_i]$.*

It follows that every decomposition $\mathcal{D}$ of a finite graph $\mathbf{G}$ yields an expression $\mathcal{E}_{\mathcal{D}}$ of that graph in terms of nested substitutions of the graphs $\mathbf{G}[M]/\pi(M)$ for $M \in \mathcal{D}$. The graphs $\mathbf{G}[M]/\pi(M)$ are called the *factors* of the decomposition.

Conversely with every expression $\mathcal{E}$ of a finite graph $\mathbf{G}$ in terms of nested substitutions of graphs is associated a decomposition $\mathcal{D}$ such that $\mathcal{E}_{\mathcal{D}} = \mathcal{E}$.

A graph has several decompositions. However a canonical one, called the modular decomposition, can be defined.

Definition 2.4. *Strong modules. The modular decomposition of a finite graph.* A module in a graph is *strong* if it is non-empty and overlaps no module. The set of strong modules of a finite graph $\mathbf{G}$ is a decomposition $\mathcal{D}$ called its *modular decomposition*. We will identify that set of strong modules and the corresponding tree, that we will denote $mdec(\mathbf{G})$.

The quotient graphs $\mathbf{G}[M]/\pi(M)$ for $M \in \mathcal{D}$, M not singleton have a particular structure described by the following *Fundamental Theorem of Modular Decomposition*. A graph is *prime* if it has no non-trivial module and at least three vertices.

Theorem 2.1. *For every finite graph $\mathbf{G}$ with at least two vertices, if π is its modular partition into maximal proper strong modules, then $\mathbf{G}/\pi$ has at least two vertices and has one of the following forms :*

(1) *either it is edge-free*
(2) *or it is complete*
(3) *or it is a linear order*
(4) *or it is prime*

The graph $\mathbf{G}$ has one and only one of the following forms, respectively :

(I) $\mathbf{G} = \mathbf{H}_1 \oplus \cdots \oplus \mathbf{H}_n$, $n \geq 2$, *where no $\mathbf{H}_i$ is a free sum.*
(II) $\mathbf{G} = \mathbf{H}_1 \otimes \cdots \otimes \mathbf{H}_n$, $n \geq 2$, *where no $\mathbf{H}_i$ is a complete sum.*
(III) $\mathbf{G} = \mathbf{H}_1 \overrightarrow{\otimes} \cdots \overrightarrow{\otimes} \mathbf{H}_n$, $n \geq 2$, *where no $\mathbf{H}_i$ is a linear sum.*
(IV) $\mathbf{G} = \mathbf{P}[\mathbf{H}_1/v_1, \ldots, \mathbf{H}_n/v_n]$ *where $\mathbf{P}$ is a prime graph and $n \geq 3$.*

It follows that each node M of the modular decomposition which is not a leaf has type I, II, III or IV, corresponding respectively to the expression of $\mathbf{G}[M]$ of one of the above forms with $\mathbf{H}_i = \mathbf{G}[N_i]$ where $N_1, \ldots, N_n$ are the sons of M in the tree $mdec(\mathbf{G})$. If $\mathbf{G}$ is undirected, then case III does not occur and furthermore $n \geq 4$ in case IV because the smallest prime undirected graph is the undirected path with 4 vertices.

That theorem seems to have been rediscovered many times. For finite undirected graphs, Kelly [37] attributes it to Gallai [30]. For the case of directed graphs, generalizations and references, we refer the reader to [28, 32, 40, 41]. Möhring and Radermacher [41] call "substitution decomposition" the modular decomposition and thus emphasize its relation with graph substitution. They show that analogous decompositions can be defined for other discrete structures like hypergraphs and Boolean functions. The structure of prime graphs is investigated by Ille in [36, 35], which improve previous results by Ehrenfeucht and Rozenberg [26] and Schmerl and Trotter [44].

The graphs for which all nodes of the modular decomposition are of the forms (I) or (II) are called *cographs* ; those are precisely the simple undirected graphs without induced path with 4 vertices.

2.3. **Representing decompositions by binary structures.** Relational structures and logic are reviewed in the appendix. A *binary structure* is a relational structure all relations of which are unary or binary. As said above, a graph $\mathbf{G}$ is considered as the binary structure $\langle V_{\mathbf{G}}, edg_{\mathbf{G}}\rangle$. It is undirected if and only if $edg_{\mathbf{G}}$ is symmetric. We consider only graphs without loops, hence $edg_{\mathbf{G}}(x,x)$ never holds.

We have defined a decomposition $\mathcal{D}$ of a graph $\mathbf{G}$ as a family of subsets of the vertex set $V_{\mathbf{G}}$. Such an object is not a relational structure. However, a decomposition is a tree (for inclusion as ancestor relation). Hence we can represent it by a relational structure $\langle N_T, \leq_T\rangle$ where N_T is in bijection by, say h, with $\mathcal{D}$ and $x \leq_T y$ iff $h(x) \subseteq h(y)$. Hence T is the tree $Tree(\mathcal{D})$. The vertex set $V_{\mathbf{G}}$ is then in bijection with the set of leaves of T, because $\{v\} \in \mathcal{D}$ for every $v \in V_{\mathbf{G}}$. This bijection is defined by $h(x) = \{v\}$ for $x \in N_T$ and $v \in V_{\mathbf{G}}$.

The graph $\mathbf{G}$ cannot be determined from this binary structure. We will expand it into a richer binary structure by adding unary and binary relations representing the structure of nodes, *i.e.* the graphs $\mathbf{G}[M]/\pi_{\mathcal{D}}(M)$ for the nodes M of $Tree(\mathcal{D})$. We obtain in this way a binary structure that can be considered as a labeled graph.

Definition 2.5. *The graph representation of the modular decomposition.* Let $\mathbf{G}$ be a finite graph and $\mathcal{D}$ its modular decomposition. The *graph representation* of $\mathcal{D}$ is the binary structure

$$Gdec(\mathbf{G}) = \langle N_T, \leq_T, lab_{\oplus}, lab_{\otimes}, lab_{\overrightarrow{\otimes}}, fedg\rangle$$

where $\langle N_T, \leq_T\rangle$ is as above, $T = Tree(\mathcal{D})$,

- $lab_{\oplus}(x)$ holds if and only if x is a node of type (I), *i.e.*, defines a free sum,
- $lab_{\otimes}(x)$ holds if and only if x is a node of type (II), *i.e.*, defines a complete sum,
- $lab_{\overrightarrow{\otimes}}(x)$ holds if and only if x is a node of type (III), *i.e.*, defines a linear sum,
- $fedg(x,y)$ holds if and only if x and y are two sons of a node z of type (III) or (IV) such that $x \to y$ in the graph $\mathbf{G}[M(z)]/\pi_{\mathcal{D}}(z)$, where $M(z)$ is the corresponding module.

The prefix f in $fedg$ recalls that we deal with the edges of certain factors of the modular decomposition and not with the edges of the graph $\mathbf{G}$.

If z has label $\overrightarrow{\otimes}$, *i.e.* is of type III, then $fedg$ linearly orders its set of sons. If z has no label and is not a leaf, then it is of type IV and $fedg$ represents the edges of the corresponding prime factor $\mathbf{G}[M(z)]/\pi_{\mathcal{D}}(z)$.

The structure $Gdec(\mathbf{G})$ is somewhat redundant. One could delete the labelling of nodes of type III, because the distinction between types III and IV can be made from the relation $fedg$. However, we think more clear to have a specific labelling for these nodes.

For a cograph, the structure $Gdec(\mathbf{G})$ reduces to $\langle N_T, \leq_T, lab_{\oplus}, lab_{\otimes}\rangle$ which represents a term over the operation symbols $\oplus$ and $\otimes$ and the constant $\mathbf{1}$. Since $\oplus$ and $\otimes$ are associative and commutative, they are handled as functions of variable arity (at least 2) and unordered set of arguments.

Proposition 2.2. *Every graph can be defined from the graph representation of its modular decomposition.*

Proof. Let $\mathbf{G}$ be a finite graph and

$$Gdec(\mathbf{G}) = \langle N_T, \leq_T, lab_\oplus, lab_\otimes, lab_{\overrightarrow{\otimes}}, fedg\rangle.$$

Then $\mathbf{G} = \langle V_\mathbf{G}, edg_\mathbf{G}\rangle$ can be defined as follows :

$$V_\mathbf{G} = \{x \in N_T : y <_T x \text{ for no } y \in N_T\}$$

$edg_\mathbf{G}(x, y)$ holds if and only if $x, y \in V_\mathbf{G}$, $x \neq y$ and, letting z be their least common ancestor in the tree T,

- either $lab_\otimes(z)$ holds
- or $\neg(lab_\oplus(z) \vee lab_\otimes(z))$ holds and $fedg(x', y')$, where x', y' are the (necessarily) distinct sons of z such that $x \leq_T x'$, $y \leq_T y'$

The correctness of this definition follows immediately form the definitions □

From that proof it is clear that $\mathbf{G}$ can be reconstructed from $Gdec(\mathbf{G})$ by first-order formulas. It is proved by Courcelle [13] that $Gdec(\mathbf{G})$ can be constructed from $\mathbf{G}$ by monadic second-order formulas, with the help of an auxiliary linear ordering $\preccurlyeq$ of $V_\mathbf{G}$.

Our objectives will be the following ones :

- to extend the definition of modular decomposition to infinite graphs ;
- to extend the above result of [13] to countable graphs given with an auxiliary linear ordering of type ω of the vertex set ;
- to replace the binary structure $Gdec(\mathbf{G})$ by an alternative binary structure, called its sparse representation, which, considered as a graph, has vertices of "low degree".

3. Trees

In the present Section, no particular assumptions of cardinality are made.

Definitions 3.1. (*Trees, join-trees and leafy trees.*) Our terminology borrows from R. Fraïssé [29], with some variations.

Given a partially ordered set $(P, \leq)$, two elements x and $y \in P$ are *comparable* if $x \leq y$ or $y \leq x$, they are *compatible* if the pair $\{x, y\}$ has an upper-bound. A *chain* is a set of pairwise comparable vertices. A set $Q \subseteq P$ of vertices is (upwards) *directed* if

$$\forall x, y \in Q,\ \exists z \in Q,\ x \leq z \wedge y \leq z$$

A set Q is an *up-set* (resp. *down-set)* if

$$\forall x \in Q,\ \forall y \in P,\ x \leq y \Rightarrow y \in Q \text{ (resp. } \forall x \in Q,\ \forall y \in P,\ x \geq y \Rightarrow y \in Q).$$

We will use the following notation : $P^x = \{y \in P : x \leq y\}$, $P^{>x} = \{y \in P : y > x\}$, $P_x = \{y \in P : y \leq x\}$ and $P_{<x} = \{y \in P : y < x\}$.

A forest is a partial order $(T, \leq)$ such that for every $x \in T$, the set T^x is a chain ; the elements of T are called *nodes*. A *tree* is a forest that is *directed*. In a forest, the relation of compatibility is an equivalence whose classes are called the *components* of T. The components are the maximal directed sets ; they are also the connected components of the comparability graph.

A tree is a *join-tree* if any two nodes x and y have a least upper bound, called their *join* and denoted by $x \vee y$. A join-tree can be defined as a relational structure $(T, \leq)$ or as an algebraic structure $(T, \vee)$, with $x \leq y$ if and only if $y = x \vee y$. A *sub-join-tree* of $(T, \leq)$ is a tree $(T', \leq')$ with $T' \subseteq T$ and, for any x and y in T', $x \vee' y = x \vee y$ (so in particular $x \leq' y$ if and only if $x \leq y$).

A *leaf* in a forest is a minimal node, a *root* is a maximal one. An *internal node* is one that is not a leaf. A forest may have one or several roots, or no root at all. It may have no leaf. A tree has at most one root. We say that a tree is *leafy* if every internal node is the least upper bound of two leaves. Notice that every leafy tree is a join-tree. A finite tree is a finite rooted tree in the usual sense, and its root is the unique maximal element. A finite forest is a finite free sum of finite trees.

If $x \leq y$, we say that the node y is an *ancestor of* x. We say that y is the *father* of x if it is the least node among those greater than x ; in that case, we say that x is a *son* of y.

Definition 3.2. (*Directions in forests.*) Let T be a forest. For every node x, $T_{<x}$ ordered by the induced ordering, is a forest, hence a free sum of trees. Each of these trees D is called a *direction relative to* x.[3] For $y \in D$, we say that D is the *direction of* y *relative to* x. We denote it by $dir_x(y)$. We denote by $Dir(x, T)$ the set of directions relative to x. Thus the directions relative to x are the components of $T_{<x}$.

The *degree* of a node x is the cardinality of $Dir(x, T)$. A tree is *binary* if every node has degree at most 2. If a node is $y \vee z$ where y and z are incomparable, it has degree at least 2. If T is finite, this definition of the degree of a node yields the number of its sons.

Here are some easy facts listed for later reference :

Lemma 3.1.

[3]We identify a direction and the corresponding set of nodes.

(1) *In a tree, given two nodes in distinct directions relative to a node x, then x is their join ; conversely, if two incomparable nodes have a join, then they lie in distinct directions relative to it.*
(2) *In a tree every directed set of nodes has a cofinal chain*[4].
(3) *In a join-tree, the least-upper bound of any three-element set is the join of a pair of these elements, indeed of at least two pairs of these elements.*
(4) *In a join-tree, the least-upper bound of any finite set is the join of two elements of that set.*
(5) *In a join-tree, of the three least upper-bounds of the pairs of a three-element set, at least two equal the greatest.*
(6) *In a leafy tree, if v is an upper-bound of a set X of nodes, but not the least one, then there is a leaf y such that for every $x \in X$, $x \vee y = v$.*
(7) *In a non-empty forest, the components are the non-empty directed simultaneously up- and down-sets, and the maximal chains are the non-empty chains that are up-sets and have no strict lower bound.*[5]
(8) *A tree is leafy if and only if every inner node has at least two directions and every non-empty down-set contains a leaf.*
(9) *In a tree, any two directed down-sets that meet are comparable for inclusion.*

Proof.

(2) If the directed set is empty then the empty chain suits. If the directed set D is not empty, then, given any $v \in D$, consider $C := \{u \in D : v \leq u\}$.
(3) Given three nodes x, y and z, if $x \vee y < x \vee y \vee z$, then x, y and $x \vee y$ lie in the same direction of $x \vee y \vee z$, but $x \vee y \vee z = (x \vee y) \vee z \not< x \vee y \vee z$, thus z cannot lie to the same direction (*i.e.* either it equals $x \vee y \vee z$ or it lies in an other direction), so that both $x \vee z$ and $y \vee z$ equal $x \vee y \vee z$.
(4) Induction using (3) and the associativity of $\vee$.
(5) Consequence of Point 3.
(6) Notice that X is included in a direction of v ; any leaf y lying in a different direction suits.
(9) Assume that D and D' are two distinct directed down-sets that meet. So let $y \in D \cap D'$ and, without loss of generality, $x \in D \setminus D'$, and then an upper bound z of $\{x, y\}$ in the directed set D. Now given any $x' \in D'$, consider an upper bound z' of $\{x', y\}$ in the directed set D' ; observe that $z' \leq z$: z' and z are comparable since both $\geq y$, but $z \not\leq z'$ since $x \leq z$ and z' belongs to the down-set D' that excludes x ; hence $x' \leq z' \leq z \in D$, so that $x' \in D$, that is a down-set.

□

Lemma 3.2. *Let x and y be two nodes of a tree such that $y < x$, and let D denote the direction of y relative to x. The node y is a son of x if and only if it is the greatest element of D. If D has no greatest element, then it admits a cofinal chain containing y, and x is the least upper-bound of any such chain.*

[4]A *cofinal* (resp. *coinitial*) set of a set P of vertices of a partially ordered set is any $Q \subseteq P$ with the property that for every element x of P, there is an element y of Q such that $x \leq y$ (resp. $y \leq x$).

[5]In a partially ordered set, a *strict lower-bound* of a set of vertices is a vertex that is strictly smaller than every element of that set ; in other words, it is a lower bound not belonging to the set.

Proof. Observe that, if z is a node such that $y \leq z < x$, then $z \in D$. □

Remarks 3.1. For a partially ordered set $(P, \leq)$, we let $HD(P)$ denote its *Hasse diagram*, *i.e.* the directed graph with set of vertices P and edges $x \longrightarrow y$ such that $x < y$ and there is no z with $x < z < y$. We say that P *is diagram-connected* if P is the transitive closure of $HD(P)$ and $HD(P)$ is connected. A tree is *diagram-connected* if the graph of the father-son relation is connected ; any two nodes are then at finite distance in this graph. A diagram-connected tree may have no root.

The infinite trees representing *infinite algebraic terms over finite signatures* (Courcelle [8] or [9]) and the *genealogies* (F. Gire, M. Nivat [31]) are diagram-connected join-trees. Some infinite trees as defined in Definition 3.1 represent neither infinite trees in the sense of [8], nor genealogies.

4. Strong and robust modules

4.1. **The tree of strong modules.** Although a forest is a graph or can be considered as a graph, we use the special 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.

Modules and modular partitions are defined (Definition 2.2) in Section 2 ; recall that a module is strong if it is non-empty and it overlaps no module.

Notation 4.1. We denote by $sdec(\mathbf{G})$ the set of strong modules of any graph $\mathbf{G}$.

The results of Section 2 do not extend immediately to infinite graphs, because it may happen that a graph has no maximal proper strong module (see Example 4.3 below). In such a case, Theorem 2.1 does not extend. Besides, a countable graph may have uncountably many strong modules (see Example 4.4) ; still the tree of strong modules has countably many non-limit nodes. These particular nodes will be sufficient to reconstruct the tree by a kind of "completion".

Let us first mention the following easy facts :

Lemma 4.1. *In a graph,*

1. *the intersection of a non-empty set of modules is a module (possibly empty) ;*
2. *the union of two modules that meet is a module, and more generally, the union of a set of modules is a module as soon as the* meeting *relation on that set is connected ;*
3. *for two modules M and N, if $M \backslash N$ is non-empty, then $N \backslash M$ is a module.* □

Example 4.1. *The modules of a chain are its intervals ; in particular the strong modules of a chain are trivial.* The first assertion is clear. As for the remaining one, given a chain $(C, \leq)$, if I is a proper interval with at least two elements $a < b$, then at least one of the two intervals $\{x \in C : x > a\}$ and $\{x \in C : x < b\}$ overlaps I.

Example 4.2. *Every connected component of a graph is a strong module.*

Example 4.3. A *bicoloring* of a chain $\mathbf{C} = (C, \leq)$ is a mapping $\chi : C \to \{\oplus, \otimes\}$. The graph associated with a bicolored chain $\mathbf{C}$ as above is the undirected graph on the set C such that two distinct vertices are linked if and only if the greater one is colored by $\otimes$. (Such graphs are sometimes called *linear cographs*.) A bicoloring χ of $\mathbf{C}$ is *good* if for $x < y$ and $i \in \{\oplus, \otimes\}$, there is z such that $x \leq z \leq y$ and $\chi(z) = i$. In particular, no two consecutive vertices have the same color.

The modules of the graph associated with a good bicoloring of a chain are the singletons and the down-sets ; in particular all its modules are strong.

Any down-set I is a module, even when the bi-coloring fails to be good. Conversely, consider the graph $\mathbf{G}$ associated with a good bicoloring of a chain $(C, \leq)$. If A is a subset of C failing to be an interval, then it is not a module of $\mathbf{G}$: letting $a < b < c$ with a and c in A and $b \notin A$, in case $\chi(b) \neq \chi(c)$, b is linked in different ways to a and c, and in case $\chi(b) = \chi(c)$, there must be a vertex d in between with a different color. If $d \in M$ then let it play the role of c and otherwise let it play the role of b. A non-singleton interval A failing to be a down-set is not a module either : given $a < b$ with $b \in A$ and $a \notin A$, consider some other element c of A, since A is convex, $a < c$; if b and c have different labels then a is linked to them in different ways, and if they have the same label, then given some d in between with a different label, that d must lie in A, which is assumed to be convex, and it is linked to a differently from c.

The chain $\mathbb{Z}$ of integers has exactly two good bicolorings, they are isomorphic. The associated graph $\mathbf{G}_\zeta$ is represented on Figure 1.

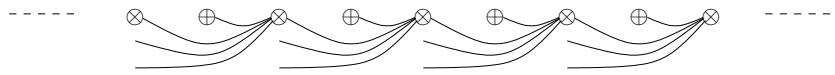

FIGURE 1. The graph $\mathbf{G}_\zeta$

Example 4.4. The good bicolorings of the chain $\mathbb{Q}$ of rational numbers are isomorphic with one another ; let $\mathbf{G}_\eta$ denote the associated graph. (As for the existence, one can label by $\oplus$ the rationals of the form $\frac{m}{2^n}$ for some $m \in \mathbb{Z}$ and $n \in \mathbb{N}$, and by $\otimes$ the others.)

Since $\mathbb{Q}$ has uncountably many down-sets, the graph $\mathbf{G}_\eta$ is a countable graph with uncountably many strong modules.

Let us consider the basic properties of the tree of strong modules.

Lemma 4.2.

(1) *The intersection of any set of strong modules is empty or is a strong module.*
(2) *The union of any directed set of strong modules is a strong module.*

Notice that, in that statement, the intersection of the empty set is the vertex set of $\mathbf{G}$.

Proof.

(1) Let $\mathcal{E} \subseteq sdec(\mathbf{G})$ with a non-empty intersection. First $\cap\mathcal{E}$ is a module, like any intersection of a set of modules. If a set $M \subseteq V_{\mathbf{G}}$ overlaps $\cap\mathcal{E}$, then it overlaps some member of $\mathcal{E}$: Indeed since $M \setminus \cap\,\mathcal{E} \neq \varnothing$ and $M \setminus \cap\,\mathcal{E} = \cup\{M \setminus E : E \in \mathcal{E}\}$, there is some $E \in \mathcal{E}$ such that $M \setminus E \neq \varnothing$, while, for any such E, $E \setminus M \supseteq (\cap\mathcal{E}) \setminus M \neq \varnothing$. Thus $\cap\mathcal{E}$ overlaps no module, since no member of $\mathcal{E}$ overlaps any module. Incidentally notice that $\mathcal{E}$ is a chain whenever its intersection is non-empty.
(2) Let $\mathcal{E}$ be a directed subset of $sdec(\mathbf{G})$. If a set $M \subseteq V_{\mathbf{G}}$ overlaps $\cup\mathcal{E}$, then it overlaps some member of $\mathcal{E}$: Indeed since $(\cup\mathcal{E}) \setminus M \neq \varnothing$ and $(\cup\mathcal{E}) \setminus M = \cup\{E \setminus M : E \in \mathcal{E}\}$, there is some $E \in \mathcal{E}$ such that $E \setminus M \neq \varnothing$, while, for any such E, $M \setminus E \supseteq M \setminus (\cup\mathcal{E}) \neq \varnothing$. Thus again $\cup\mathcal{E}$ belongs to $sdec(\mathbf{G})$.

□

Notation 4.2. For any non-empty $A \subseteq V_{\mathbf{G}}$, we let $S(A)$ denote the intersection of all strong modules including A ; thus $S(A)$ is the least strong module including A.

From Lemma 4.2, it immediately follows :

Corollary 4.1.

(1) *$sdec(\mathbf{G}) \cup \{\varnothing\}$ is a complete lattice. The greatest-lower bound of a subset $\mathcal{E}$ of $sdec(\mathbf{G})$ is its intersection $\cap\mathcal{E}$; its least upper-bound is the intersection of all members of $sdec(\mathbf{G})$ including its union : $\bigvee \mathcal{E} = \cap\{S \in sdec(\mathbf{G}) : \cup\mathcal{E} \subseteq S\}$. In particular for every non-empty subset X of $V_{\mathbf{G}}$, the least strong module including X is $S(X)$. The tree $sdec(\mathbf{G})$ of strong modules of $\mathbf{G}$ is a join-tree.*

(2) *The least upper-bound of a directed set of strong modules is its union.*

(3) *If $\mathcal{D}$ is a direction relative to some $M \in sdec(\mathbf{G})$ then, either $\mathcal{D}$ has a greatest member N in which case N is a son of M in $sdec(\mathbf{G})$, or M itself is the least upper bound of $\mathcal{D}$ in $sdec(\mathbf{G})$.*

Proof. Only (3) requires some comment : If $\bigvee \mathcal{D} \subsetneq M$, then for any $N \in sdec(\mathbf{G})$ such that $\bigvee \mathcal{D} \subseteq N \subsetneq M$, $\mathcal{D} \cup \{N\}$ is a directed set of nodes all lesser than M, hence $\mathcal{D} \cup \{N\} \subseteq \mathcal{D}$ by maximality of $\mathcal{D}$. □

The following lemma about modules of subgraphs and quotient graphs is easy to establish.

Lemma 4.3. *Consider a graph $\mathbf{G}$.*

(1) (a) *Every module of $\mathbf{G}$ included in a set of vertices A is a module of the induced graph $\mathbf{G}[A]$. Those are the only modules of $\mathbf{G}[A]$ if and only if A is a module of $\mathbf{G}$.*

(b) *Every strong module of $\mathbf{G}$ included in a module M is a strong module of $\mathbf{G}[M]$. Those are the only ones if and only if M is a strong module of $\mathbf{G}$.*

(2) (a) *Let $\mathcal{C}$ be a modular partition of $\mathbf{G}$. The modules of the quotient graph $\mathbf{G}/\mathcal{C}$ are the subsets of $\mathcal{C}$ whose union is a module of $\mathbf{G}$.*

(b) *Let $\mathcal{C}$ be a modular partition of $\mathbf{G}$ formed of strong modules. The modules of the graph $\mathbf{G}$ are its sets of vertices that are the union of a module of the quotient graph $\mathbf{G}/\mathcal{C}$.* □

4.2. **The tree of robust modules.**

Notation 4.3. For possibly equal vertices x and y of a graph $\mathbf{G}$, let $S(x, y)$ denote the least strong module $S(\{x, y\})$ containing x and y. We call *robust* any such strong module and we denote by $rdec(\mathbf{G})$ the set $\{S(x, y) : x, y \in V_{\mathbf{G}}\}$. It is the tree, ordered by inclusion, of robust modules of $\mathbf{G}$.

For any distinct vertices x and y, let $A(x, y)$ denote the union of all strong modules containing x but not y.

Notice that a module is robust if and only if it is of the form $S(F)$ for some non-empty finite set F of vertices (see Lemma 3.1-4). It follows that that the tree $rdec(\mathbf{G})$ of robust modules is a sub-join-tree of the tree of strong modules, and it is also a leafy tree.

Example 4.5. Consider the graph $\mathbf{G}_\eta$ of Example 4.5. For any two distinct rationals x and y, the robust module $S(x, y)$ is the least down-set of $\mathbb{Q}$ containing both x and y, thus the down-set of $\mathbb{Q}$ admitting $\max\{x, y\}$ as greatest element. It follows that the robust modules of $\mathbf{G}_\eta$ are the singletons and the initial intervals of $\mathbb{Q}$ with a greatest element. Incidentally notice that for any two distinct rationals $x < y$, $A(x, y) = \mathbb{Q}_{<y}$ and $A(y, x) = \{y\}$.

For any rational number x, the robust module $\mathbb{Q}_x$ has two sons in the tree of strong modules, namely the singleton $\{x\}$ and $\mathbb{Q}_{<x}$, which is not robust. The strong module corresponding to irrational cuts, *i.e.* those of the form $\{y \in \mathbb{Q} : y < x\}$ for some irrational number x, are the strong modules that are neither a father nor a son in the tree of strong modules.

We use also this graph in Example 5.1.

Lemma 4.4. *Consider a graph.*

(1) *For two distinct vertices x and y, $A(x, y)$ is a strong module ; it is the greatest strong module containing x but not y.*
(2) (a) *For two distinct vertices x and y, the strong module $A(x, y)$ is the son of the strong module $S(x, y)$ containing x.*
    (b) *For two comparable strong modules M and N with $N \subsetneq M$, for any vertices $x \in N$ and $y \in M \backslash N$, $N \subseteq A(x, y) \subsetneq S(x, y) \subseteq M$, in particular, when N is a son of M in the tree of strong modules, then $N = A(x, y)$ and $M = S(x, y)$.*
(3) *Every strong module is the union of a chain of robust modules.*

Proof.

(1) The strong modules containing x form a chain, thus $A(x, y)$ is the union of a chain of strong modules, and therefore it is a strong module.
(2) (a) The strong modules $S(x, y)$ and $A(x, y)$ both contain x, hence they are comparable. Therefore $A(x, y) \subsetneq S(x, y)$ since y belongs to $S(x, y)$ but not to $A(x, y)$. Now assume that M is a strong module such that $A(x, y) \subseteq M \subseteq S(x, y)$; in particular $x \in M$; if $y \in M$, then $S(x, y) \subseteq M$ (and then $M = S(x, y)$), and if $y \notin M$, then $M \subseteq A(x, y)$ (and then $M = A(x, y)$). Thus $A(x, y)$ is a son of $S(x, y)$.
    (b) The inclusions $N \subseteq A(x, y) \subsetneq S(x, y) \subseteq M$ hold by definition of $A(x, y)$ and $S(x, y)$. Thus, since $A(x, y)$ and $S(x, y)$ are strong modules, $N = A(x, y)$ and $M = S(x, y)$ whenever N and M are consecutive.
(3) Given a strong module M and any $x \in M$, consider $\{S(x, y) : y \in M\}$.

□

Corollary 4.2. *For a strong module M of a graph $\mathbf{G}$, the following are equivalent :*

(1) *M is a non-singleton robust module,* i.e. *M is of the form $S(x, y)$ for two* distinct *vertices.*
(2) *M is a father in the tree of strong modules.*
(3) *The degree of M is greater than one in the tree of strong modules.*
(4) *The induced graph $\mathbf{G}[M]$ has a maximal proper strong module.*

If those conditions hold :

(1') *For two elements x and y of M, $M = S(x, y)$ if and only if their directions relative to M are distinct ; and the sons of M are the sets $A(x, y)$ for all such x and y in M.*

(2') *All directions relative to M in the tree of strong modules have greatest elements, which are the sons of M ; in other words (*cf. *Lemma 3.2), each strong module $N \subsetneq M$ is included in a son of M.*

(4') *The maximal proper strong modules of $\mathbf{G}[M]$ partitionate M and are the sons of M.*

Proof. (1) implies (3), since x and y belong to distinct directions relative to $S(x, y)$ (Lemma 3.1-1), and the converse holds, since a strong module M of degree more than one is $S(x, y)$ for any x and y belonging to distinct directions, which also yields the first part of (1').

From Lemma 4.4-2 it follows that (1) and (2) are equivalent, and also that (1) implies the second part of (1'), as well as (2').

Recall that the strong modules of $\mathbf{G}[M]$ are precisely the strong modules of $\mathbf{G}$ included in M (Lemma 4.1-1). It follows that (2) and (4) are equivalent, and that (2') and (4') are equivalent. □

Corollary 4.3. *For a strong module M, the following are equivalent*

(1) *M is not robust,*
(2) *M is a limit node in the tree of strong modules,*
(3) *M is the union of a chain of smaller strong (resp. robust) modules.*

Proof. 1⇒3 by Lemma 4.4-3.

3⇒2 Clear.

2⇒1 Assuming that M is the least upper bound of a directed set $\mathcal{D}$ of strictly lesser strong modules, let us check that M has only one direction, namely the down set generated by $\mathcal{D}$ (then, having degree one, M will not be robust by Corollary 4.2 above) : That down-set is clearly a directed set of lesser nodes, it remains then to check that every node lesser than M is less than or equal to a member of $\mathcal{D}$. Recall that indeed $M = \cup\mathcal{D}$ (Corollary 4.1-2). Now given a strong module $N \subsetneq M$, consider some vertex $x \in N$ and $y \in M \backslash N$, then letting $D' \in \mathcal{D}$ containing x and $D'' \in \mathcal{D}$ containing y, any upper bound of $\{D', D''\}$ in $\mathcal{D}$ meets N but is not included in N, and then since it cannot overlap N, it includes N.

□

Remark 4.1. The article [27], that sudies the modular decomposition of infinite graphs, defines as *fully decomposable* the graphs[6] all non-singleton strong modules of which satisfy Property 4 of Corollary 4.2 above, thus, according to that corollary, those graphs all strong modules of which are robust. As it is mentioned there, they are the graphs whose tree of strong modules has no infinite increasing sequence (indeed the union of such a sequence is a non-singleton limit strong module, hence a strong non-robust module by Corollary 4.3-2 ; and conversely if there is such a strong non-robust module then there is a chain of strong modules with no greatest member by Corollary 4.3-3, and then there is an increasing sequence of strong modules). Among those are the graphs whose tree of strong modules is rooted and

[6][27] deals with 2-structures, which generalize binary relations. See Section 4.3.2 below.

diagram-connected. Indeed these fully decomposable graphs are those whose tree of strong modules is well-founded for the reverse ordering.

Definition 4.1. (*Canonical partition and skeleton of a robust graph.*) A graph $\mathbf{G}$ is *robust* if its vertex set is a robust module. In that case, we call *canonical* its partition into its maximal proper strong modules, and we call *skeleton* of $\mathbf{G}$ the corresponding quotient graph (actually it embeds into $\mathbf{G}$). For every non-singleton robust module M of $\mathbf{G}$, we let $\mathcal{C}_M$ denote its set of sons in the tree of strong modules ; it is also the canonical partition of the induced graph $\mathbf{G}[M]$, call it the canonical partition of M, and likewise, call skeleton of M the quotient graph $\mathbf{G}[M]/\mathcal{C}_M$.

Example 4.6. (See Example 4.2.) *Every non-connected graph is robust and its maximal proper strong modules are its connected components.* Dually, since a graph and its edge-complement graph have the same modules, a graph whose edge-complement graph is not connected is robust and its maximal proper strong modules are the connected components of the edge-complement graph.

The connected components of a non-connected graph are maximal among proper strong modules : if a proper module M is included in no connected component, then, since the connected components are strong modules, M includes any component it meets, and thus it is the union of at least 2 but not all connected components, but then it overlaps a module (namely the union of a connected component included in M and of the complement of M). Finally the components are the only maximal proper strong modules since they already cover the vertex set.

4.3. **Basic and elementary graphs.** For every robust module M of a graph $\mathbf{G}$, the maximal proper strong modules of the induced graph $\mathbf{G}[M]$ form a partition $\mathcal{C}$ of M and the quotient graph $\mathbf{G}[M]/\mathcal{C}$ has no non-trivial strong module. Besides the prime graphs, which have no non-trivial module at all, the linear orders, the complete graphs and the edge-free graphs have no non-trivial strong module. It turns out that they are the only ones. The finite case is well known. The general case is Theorem 4-2 of [27]. The proof that we give here relies on a direct proof of the following observation : *in a graph admitting non-trivial modules but no non-trivial strong ones, every two distinct vertices are separated by a partition into two modules* (Proposition 4.2). The result is stated below in the framework of graphs (loop-free directed graphs). We reformulate our proof in [27]'s framework of labeled 2-structures in Proposition 4.3 of Section 4.3.2. Notice that both the statement and the proof of the characterization formulated by Proposition 4.2 are the same in the two frameworks.

Definition 4.2 (Basic and elementary graphs)**.** Say that a graph is *basic* if it has at least two vertices and no non-trivial strong module. Say that a graph is *elementary* if it is basic and non-prime, thus if it has non-trivial modules but no strong one, or if it has two vertices.[7]

The term *basic* comes from the fact that the graphs at stake are precisely those from which all other graphs are built (such graphs are called *special* in [27] and [28]).

Definition 4.3 (Modular bi-partition)**.** A *bi-partition* of a set is a partition in two classes ; two elements of the set are *separated* if they lie in two different classes of

[7]In the closely related theory of graph decomposition by Cunningham [22], see also [18], such graphs are called *brittle* because they are decomposable in many ways.

the partition. A modular bi-partition of a graph is a partition of its vertex set into two (non-empty) modules.

4.3.1. *The elementary graphs.*

Proposition 4.1. *(*Cf. [27]*.) A graph with at least two vertices is elementary if and only if it is edge-free, or is complete or is a linear order.*

That proposition follows from the technical one :

Proposition 4.2. *A graph with at least two vertices is elementary if and only if any two distinct vertices are separated by a modular bi-partition.*

Proof of Proposition 4.1. That $\mathbf{G}$ is elementary whenever it is free or complete or a linear order follows from Examples 4.2 and 4.1. Conversely, assume that the graph $\mathbf{G}$ is elementary.

Say that a pair $\{x, y\}$ of distinct vertices has type

(I) if there is no edge between these vertices,
(II) if there are edges from x to y and from y to x,
(III) if there is exactly one directed edge between them.

If M is a module and y a vertex outside M, then all pairs of the form $\{x, y\}$ for $x \in M$ have the same type. In other words, for each $t \in$\{I,II,III\}, a module of $\mathbf{G}$ is also a module of the undirected graph $\mathbf{G}_t$ with vertex set $V_{\mathbf{G}}$ and edges the pairs of type t.

Given any type $t \in$\{I,II,III\} such that $\mathbf{G}_t$ has at least one edge, consider two vertices x and y such that $\{x, y\}$ has type t and a partition into two modules X containing x and Y containing y ; then the edge relation of $\mathbf{G}_t$ contains the complete bipartite graph between X and Y ; in particular it is connected and no other graph $\mathbf{G}_s$ $(s \neq t)$ can be connected. It follows that only one type can occur.

If that type is II then $\mathbf{G}$ is a complete graph, if it is I then $\mathbf{G}$ is edge-free. Now assume that it is III. Then $\mathbf{G}$ is total and oriented, and it remains to check that it is transitive : Given $x \to y \to z$, consider a partition into two modules X containing x and Z containing z, and observe that, wherever y lies, $x \to z$: if $y \in Z$ then $x \to z$ and $x \to y$; if $y \in X$ then $z \leftarrow x$ and $z \leftarrow y$; in either case $x \to z$. □

Proof of Proposition 4.2. If a graph with at least three vertices satisfies the stated separation property, then it is not prime ; still it is basic, since for any non-trivial module M and any partition of the vertex set into two modules separating two elements of M, at least one of the classes of the partition must overlap M.

Now let us prove the converse : Assume that $\mathbf{G}$ is an elementary graph with at least three vertices (the case of two vertices being obvious). Let V denote its vertex set.

(1) First we prove that *every vertex x belongs to a non-trivial module* : Let A denote a non-trivial module. Assume that A does not already contain x. Consider B the union of all modules including A but excluding x. B is a non-trivial module thus it must overlap some (non-trivial) module C ; any such C must contain x, otherwise $B \cup C$ would be a module including A, excluding x and strictly including B.
(2) Second we prove that *for every two distinct vertices there is a non-trivial module containing one and only one of them* : Assume not. By the fact

above, there would be a non-trivial module containing both, thus the smallest such module C (the intersection of all of them) would be non-trivial. Let D be a module overlapping C. By assumption D contains none or both of them ; in the first case $C\backslash D$, and in the second case $C \cap D$, contradicts the minimality of C.

(3) Now consider two distinct vertices x and y. Then let X denote the greatest module containing x but not y, and let Y denote the greatest module containing y but not x. Notice that $Y\backslash X$ is a module (Lemma 4.1) since $X\backslash Y$, which contains x, is not empty. Then the proof will be complete once we check that $X \cup Y = V$, since then $\{X, Y\backslash X\}$ will be the desired partition. So let us check that $X \cup Y = V$:

First let us observe that *if a module C overlaps X then $y \in C \subseteq X \cup Y$* (and the same statement with X, y and Y, x interchanged) : Indeed $y \in C$ otherwise the consideration of $C \cup X$ would contradict the maximality of X ; in particular the module $C\backslash X$ also contains y, thus $Y \cup (C\backslash X)$ is a module containing y but not x, hence it is included in Y ; finally $C = (C \cap X) \cup (C\backslash X) \subseteq X \cup Y$.

It follows that $X \cup Y$ is a module : Indeed by 2 above, X or Y is a non-trivial module, and hence, by assumption of elementarity, it overlaps some module C ; then such a C also meets the other one, by the observation above, hence $X \cup C \cup Y$ is a module, but it equals $X \cup Y$ still by that observation.

Besides observe the following general fact : *if a set overlaps the union of two sets but none of these two sets, then it strictly includes one and is disjoint from the other.* (Indeed it meets at least one of these sets, and since it is not included in it but does not overlap it either, then it strictly includes it ; now if it met the other one then it would also have to include it and then it would include their union and therefore would not overlap it.)

Finally $X \cup Y = V$: Otherwise the module $X \cup Y$ would overlap some module C ; that module C cannot strictly include one and be disjoint from the other, by the maximality property of the one it would be including ; thus it follows from the last observation that C overlaps X or Y, and then it follows from the preceding observation that $C \subseteq X \cup Y$, contradicting their overlapping.

□

4.3.2. *Labeled 2-structures.* The purpose of this subsection is to relate the present definitions and proofs to the setting of [27]. It will not be used elsewhere in the article.

The proposition below is Theorem 4-2 of [27]. The proof we give relies on Proposition 4.2 above.

Given a set of labels Λ endowed with an involution $\lambda \mapsto \lambda^{-1}$, a *reversible labeled 2-structure* is a mapping $\mathbf{S} : (V_{\mathbf{S}})^2_* \to \Lambda$ from the set $(V_{\mathbf{S}})^2_*$ of ordered pairs of distinct elements of $V_{\mathbf{S}}$, its *domain*, with the property that for every pair of vertices, $\mathbf{S}(y, x) = (\mathbf{S}(x, y))^{-1}$. For such a structure $\mathbf{S}$, a subset M of its domain $V_{\mathbf{S}}$ is a module if for any $y \in V_{\mathbf{S}}\backslash M$, the mapping $x \mapsto \mathbf{S}(x, y)$ is constant on M. Then the notions of strong or robust modules are derived, as well as all related results.

Proposition 4.3. [27] *Consider a reversible labeled 2-structure* $\mathbf{S} : (V_{\mathbf{S}})^2_* \to \Lambda$, *having at least one non-trivial module but no non-trivial strong module. Then there is a label* λ *such that for every ordered pair* (x, y) *of (distinct) vertices,* $\mathbf{S}(x, y) \in \{\lambda, \lambda^{-1}\}$. *Furthermore if* $\lambda \neq \lambda^{-1}$, *then the relation* $\mathbf{S}(x, y) = \lambda$ *(written* $x \xrightarrow{\lambda} y$*) defines a (strict) linear ordering on* $V_{\mathbf{S}}$.

Proof. Assume without loss of generality that $\mathbf{S}$ has at least two vertices. Given any label λ that labels at least one pair, consider two vertices x and y such that $x \xrightarrow{\lambda} y$ and, with Proposition 4.2, a partition into two modules X containing x and Y containing y ; then the relation $\xrightarrow{\lambda}$ contains the complete bipartite graph from X to Y ; in particular it is connected and no other $\xrightarrow{\mu}$, except $\xrightarrow{\lambda^{-1}}$ can be connected. It follows that only λ and λ^{-1} can occur.

Now, if $\lambda \neq \lambda^{-1}$, then the total relation[8] $x \xrightarrow{\lambda} y$ is oriented, thus it remains to check that it is transitive : Given $x \xrightarrow{\lambda} y \xrightarrow{\lambda} z$, consider a partition into two modules X containing x and Z containing z, and observe that, wherever y lies, $x \xrightarrow{\lambda} z$: either $y \in Z$ and then $x \xrightarrow{\lambda} z$ and $x \xrightarrow{\lambda} y$, or $y \in X$ and $z \xleftarrow{\lambda} x$ and $z \xleftarrow{\lambda} y$; in either case $x \xrightarrow{\lambda} z$. □

All our definitions and results extend in an obvious way to labeled 2-structures.

4.4. **Skeletons of robust modules.**

Corollary 4.4. *The skeleton of a robust non-singleton graph is a basic graph, and therefore is*

(I) *either edge-free,*
(II) *or complete,*
(III) *or a linear order,*
(IV) *or prime.*

Proof. It follows from Corollary 4.2 that the skeleton is a basic graph ; it is then either a prime graph or an elementary graph. □

We say that the robust graph has type (I) , (II) , (III) or (IV) according to the case. We call the first three types *the elementary types.* Also we call type of a non-singleton robust module the type of the corresponding induced graph.

4.4.1. *Prime quotient.*

Lemma 4.5. *Consider a modular partition* $\mathcal{C}$ *of a graph* $\mathbf{G}$. *For each set* A *of vertices of* $\mathbf{G}$, *let* $\check{A}$ *denote the set of members of* $\mathcal{C}$ *included in* A *and let* $\hat{A}$ *denote the set of classes meeting* A. *Then given any module* M *and any subset* $\mathcal{A}$ *of* $\mathcal{C}$, *if* $\check{M} \subseteq \mathcal{A} \subseteq \hat{M}$ *then* $\mathcal{A}$ *is a module of* $\mathbf{G}/\mathcal{C}$.

Proof. Assuming that $\check{M} \subseteq \mathcal{A} \subseteq \hat{M}$, for any A and A' in $\mathcal{A}$ and $C \in \mathcal{C} \backslash \mathcal{A}$ (thus $C \notin \check{M}$), one can consider some $a \in M \cap A$, $a' \in M \cap A'$ and $c \in C \backslash M$. Then

$$C \xrightarrow{\mathbf{G}/\mathcal{C}} A \Leftrightarrow c \xrightarrow{\mathbf{G}} a \Leftrightarrow c \xrightarrow{\mathbf{G}} a' \Leftrightarrow C \xrightarrow{\mathbf{G}/\mathcal{C}} A'$$

and likewise $A \xrightarrow{\mathbf{G}/\mathcal{C}} C \Leftrightarrow A' \xrightarrow{\mathbf{G}/\mathcal{C}} C$. □

[8] We discuss $\xrightarrow{\lambda}$ as the edge relation of a graph ; "total" and "complete" are defined in Section 2.

Corollary 4.5. *Consider a modular partition $\mathcal{C}$ of a graph $\mathbf{G}$ and assume that the corresponding quotient graph $\mathbf{G}/\mathcal{C}$ is prime. Then*

(1) *Any proper module of $\mathbf{G}$ is included in a member of $\mathcal{C}$.*
(2) *The graph $\mathbf{G}$ is robust and $\mathcal{C}$ is its canonical partition.*

In particular, for any two non-equivalent vertices a and b, $S(a,b) = V_{\mathbf{G}}$.

Proof. Since $\mathbf{G}/\mathcal{C}$ is prime, $\mathcal{C}$ has at least 3 members. Consider a proper module M of $\mathbf{G}$. Then $\check{M}$ is a proper subset of $\mathbf{G}/\mathcal{C}$ and it is also a module of $\mathbf{G}/\mathcal{C}$ (Lemma 4.5), thus empty or a singleton. Then

- Either $\check{M} = \hat{M}$, in which case M is empty or a member of $\mathcal{C}$.
- Or $\check{M} \subsetneq \hat{M}$. In that case, since every intermediate subset must be a module of $\mathbf{G}/\mathcal{C}$ (Lemma 4.5) whereas $\mathbf{G}/\mathcal{C}$ has no module of size 2, $\check{M} = \varnothing$ and $\hat{M}$ is a singleton. Then M is a proper non-empty subset of some member of $\mathcal{C}$.

This establishes the first assertion. In particular the members of $\mathcal{C}$ are the maximal proper modules and also the maximal proper strong modules. □

4.4.2. *Types of adjacent nodes.*

Lemma 4.6. *Assume that M is a robust module of an elementary type of $\mathbf{G}$, and that N is a non-singleton robust module of $\mathbf{G}$ and also a son of M* i.e. *is maximal among strong modules strictly included in M. Then the type of N is distinct from that of M.*

Proof. Let $\mathcal{C}$ denote the canonical partition of $\mathbf{G}[M]$, *i.e.* its set of sons.

First assume that the skeleton $\mathbf{G}[M]/\mathcal{C}$ is edge-free. From Example 4.6 (and Lemma 4.3-1), we know that the sons of M are the connected components of $\mathbf{G}[M]$. Thus on the one hand the graph induced on M is not connected, and on the other hand the graphs induced on its sons are connected. In particular, no son of N can also be of that type.

The case where $\mathbf{G}[M]/\mathcal{C}$ is complete is similar and can be deduced from the previous one by edge-complementation.

Finally assume that $\mathbf{G}[M]/\mathcal{C}$ is a linear order. Then let $\mathcal{D}$ and $\mathcal{E}$ denote the two intervals of the quotient $M/\mathcal{C}$, formed by the classes strictly less (resp. greater) than N ; so $\mathcal{D} \xrightarrow{\mathbf{G}[M]/\mathcal{C}} \{N\} \xrightarrow{\mathbf{G}[M]/\mathcal{C}} \mathcal{E}$. If N were also of type III, then given any partition of its canonical quotient into two complementary non-empty intervals $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A} \to \mathcal{B}$, the two sets $(\cup\mathcal{D}) \cup (\cup\mathcal{A})$ and $(\cup\mathcal{B}) \cup (\cup\mathcal{E})$ would be two complementary modules of $\mathbf{G}[M]$; but at least one of those sets overlaps N, contradicting the hypothesis that N is a strong module of $\mathbf{G}[M]$ (Lemma 4.3-1). □

4.4.3. *Prime factors.*

Definition 4.4. We call *prime factors* of a graph $\mathbf{G}$ the skeletons of its robust modules that are prime graphs.

Obviously, every prime factor of a graph embeds in this graph. Moreover :

Lemma 4.7. *Every prime graph embedding in a graph embeds in a prime factor of this graph.*

Proof. Assume that P is a set of vertices of a graph $\mathbf{G}$ such that the induced graph $\mathbf{G}[P]$ is prime.

Then the least strong module $S(P)$ including P is a robust module : consider any two elements a and b of P, and observe that, since $S(a,b) \cap P$ is a module of the prime graph $\mathbf{G}[P]$ and therefore is trivial, then the module $S(a,b)$ includes P, thus it includes also $S(P)$, hence $S(P) = S(a,b)$.

Now each maximal proper strong module of $S(P)$ shares at most one vertex with P, because its intersection with P is a proper module of the prime graph $\mathbf{G}[P]$; thus $\mathbf{G}[P]$ embeds into the skeleton of $\mathbf{G}[S(P)]$. Finally that quotient, which is a basic graph admitting a prime (induced) subgraph, must be prime. □

It is well known that every prime (undirected) graph has an induced prime subgraph of three or four vertices.

5. Modular decomposition

Before proceeding with a definition of the modular decomposition, we collect in Proposition 5.1 below the main facts from Section 4.

Definition 5.1. (*Subrobust modules.*) A module of a graph $\mathbf{G}$ is *subrobust* if it is a maximal proper strong submodule of a robust module of $\mathbf{G}$.

According to Corollary 4.2, the subrobust modules are precisely the sets of the form $A(x,y)$. Then Corollary 4.4, Example 4.6, Corollary 4.5, Lemma 4.6 and Lemma 4.7 sum up to :

Proposition 5.1. *Let* $\mathbf{G}$ *be a graph.*

(1) *For every non-singleton robust module* M*, the induced graph* $\mathbf{G}[M]$ *is of one and only one of the following types :*
   - (I) *it is the free sum (denoted by* $\oplus$*) of a family of graphs* $(C_i : i \in I)$ *with card* $I \geq 2$*, and no* C_i *has type I,*
   - (II) *or it is the complete sum (denoted by* $\otimes$*) of a family of graphs* $(C_i : i \in I)$ *with card* $I \geq 2$*, and no* C_i *has type II,*
   - (III) *or it is the linear sum (denoted by* $\overrightarrow{\otimes}$*) of a linearly ordered family of graphs* $(C_i : i \in I)$ *with card* $I \geq 2$*, and no* C_i *has type III,*
   - (IV) *or it is* $\mathbf{P}[C_i/u_i; i \in I]$ *for some (unique) prime graph* $\mathbf{P}$*.*

(2) *The graphs* C_i *are the the maximal proper strong modules of* $G[M]$*. They are not necessarily robust. Their common father in the tree of strong modules of* $\mathbf{G}$ *is* M*.*

(3) *A prime graph embeds in* $\mathbf{G}$ *if and only if it embeds into a prime factor,* i.e. *in a graph* $\mathbf{P}$ *of Case IV.* □

By decomposing in this way all robust modules, we will obtain a hierarchical structure yielding the modular decomposition. That structure is unique by (1) of the proposition.

Remark 5.1. Case III does not occur when $\mathbf{G}$ is undirected. The special case of the proposition for undirected graphs is Theorem 4.6 of [37]. His proof relies on considerations of connectedness for the graph and for its edge-complement, which is specific to that particular framework. There, the module $S(X)$ is called the *strongly autonomous closure* of X, and the subrobust modules of $\mathbf{G}$ are called its *quasimaximal strongly autonomous subsets*. For his purpose the tree of all strong modules is considered implicitly as the modular decomposition.

Definition 5.2. (*Modular decomposition.*) We define the *modular decomposition* of a graph $\mathbf{G}$ as the tree $mdec(\mathbf{G})$ of its robust and subrobust modules. It is at most countable, when $\mathbf{G}$ is. For finite graphs, the notions of a strong and of a robust module coincide ; hence this notion of modular decomposition is equivalent to the usual one which is the finite rooted tree of strong modules. The tree $mdec(G)$ has a root if and only if V_G is a robust module. Otherwise G is the union of a chain of robust modules. One could of course make it rooted by adding V_G as root.

We extend to infinite, and in particular to countable graphs, what has been defined for finite graphs in Section 2 (and in [13]).

Definition 5.3. (*Graph representations of modular decompositions.*) The structure $Gdec(\mathbf{G})$ consists of the tree $mdec(\mathbf{G}) = (T, \leq)$, augmented with edges between the sons of each node M of T (which is a module of $\mathbf{G}$), in order to represent the edges between the submodules corresponding to the sons of M. It is a straightforward generalization of the similar notion defined in [13].

Formally, we define $Gdec(\mathbf{G})$ from $mdec(\mathbf{G})$ as follows :

For each node M of $mdec(\mathbf{G})$ which is neither a limit node nor a leaf, whence has at least two sons, we do the following according to its type (cf. Proposition 5.1) :

- if $\mathbf{G}[M]$ is a free sum (I), we label M by $\oplus$,
- if $\mathbf{G}[M]$ is a complete sum (II), we label M by $\otimes$,
- if $\mathbf{G}[M]$ is a linear sum (III), we label M by $\overrightarrow{\otimes}$, and we define a strict linear ordering of the sons of M (which corresponds to the linear ordering of the strong modules C_i, *cf.* Proposition 5.1), denoted by $\lhd_M$,
- if $\mathbf{G}[M]$ is a substitution in a prime graph (IV), we create edges between the sons of M corresponding to the edges of $\mathbf{P}$ in an obvious way.

By extending Definition 2.5, we obtain the structure $Gdec(\mathbf{G})$ defined as :

$$(T, \leq, lab_{\oplus}, lab_{\otimes}, lab_{\overrightarrow{\otimes}}, fedg)$$

where $(T, \leq)$ is the tree $mdec(\mathbf{G})$, $lab_{\oplus}, lab_{\otimes}, lab_{\overrightarrow{\otimes}}$ are unary predicates defining the labels $\oplus, \otimes, \overrightarrow{\otimes}$ of the nodes of types I, II, III, $fedg$ is a binary relation representing the edges created between sons of nodes of type IV, and also the linear orderings on the sons of father nodes of type III : $fedg(x, y)$ if and only if $x \lhd_{x \vee y} y$ when x, y are sons of $x \vee y$, which is a node labeled by $\overrightarrow{\otimes}$.[9] We can consider $Gdec(\mathbf{G})$ as a graph with two types of edges, corresponding to the binary relations $\leq$ and $fedg$. The symbols $\oplus, \otimes, \overrightarrow{\otimes}$ are thus vertex labels.

Lemma 5.1. *A graph* $\mathbf{G}$ *can be defined from* $Gdec(\mathbf{G})$ *as a graph the vertices of which are the leaves of* $mdec(\mathbf{G})$.

Proof. As in Proposition 2.2. □

Example 5.1. Consider the graph $\mathbf{G}$ of Figure 2. The thick edges stand for all edges between the copy of $\mathbf{G}_\eta$ (from Example 4.4) and their end vertices to the right. The graph $\mathbf{G}$ is robust of type (IV). It has a unique prime factor $\mathbf{P}$ shown on Figure 3. It has a unique elementary factor of type (III), and countably many elementary factors of type (I) and (II) (those of the copies of $\mathbf{G}_\eta$), and it has one

[9]The linear order on the sons of a node of type III is encoded by $fedg$. If in this ordering, every element has as successor and a predecessor (unless it is minimal or maximal), it is enough to encode by $fedg$ the successor of each node.

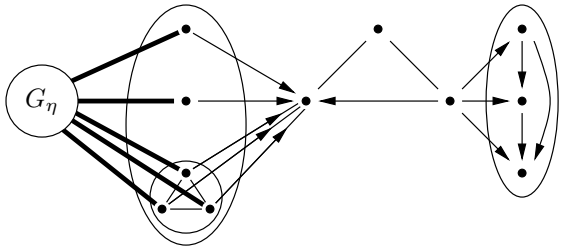

FIGURE 2. The graph **G**

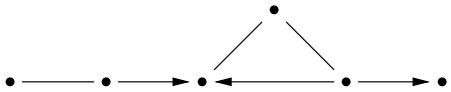

FIGURE 3. The skeleton of the robust graph **G**

factor of type (I) and arity 3 as well as one factor of type (II) and arity 3. The tree $mdec(\mathbf{G})$ and the graph representation $Gdec(\mathbf{G})$ are shown on Figure 4. A strong

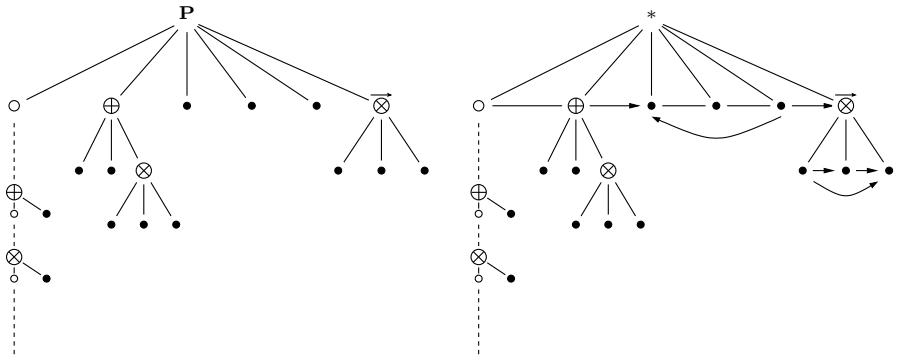

FIGURE 4. The tree $mdec(\mathbf{G})$ and the graph representation $Gdec(\mathbf{G})$

module that is a son but not a father in the tree of strong modules is marked $\circ$; the singleton robust modules are marked $\bullet$.

Definition 5.4. (*Modular trees.*) A *limit node* in a tree is a node which is the least upper bound of a directed set of strictly smaller elements. A *father node* is a node that has at least one son. In a join-tree, a father node may also be a limit node.

A join-tree is said to be *modular* if it satisfies the following conditions :

(1) No node is both a limit node and a father node.
(2) Every father node is the join of two leaves.
(3) Every limit node has degree one

Hence, a tree is modular when a non-leaf node is a father if and only if it is not a limit node, if and only if it is the join of two leaves, if and only if it has degree more than 1.

Proposition 5.2. *The tree of the modular decomposition of a graph is a modular tree.*

Proof. Corollaries 4.2 and 4.3. □

Proposition 5.3. *Every leafy tree* $\mathbf{T}$ *is the tree induced on the set of leaves and father nodes of a unique modular tree denoted by* $\widehat{\mathbf{T}}$*. (Unicity is understood up to isomorphisms preserving* $\mathbf{T}$ *pointwise.)*

Sketch of proof. The construction of $\widehat{\mathbf{T}}$ from $\mathbf{T}$ is a *completion*, where we add only the elements needed as greatest elements of certain directions. (Similar but different completions are used in semantics of recursive program schemes, see [9]).

Let $\widehat{T}$ consist of the following sets : of all directions relative to the inner nodes, and of the sets of the form T_u ($=\{w \in T : w \leq u\}$) for all nodes u (a direction can be of the form T_u). We claim that $(\widehat{T}, \subseteq)$ is a join-tree (*cf.* Lemma 3.1-9) and that the mapping that associates T_u with u is a join-embedding of T into $\widehat{T}$. The "new" elements in $\widehat{T}$ are the directions which have no greatest element. □

Proposition 5.4. *For every graph* $\mathbf{G}$*,* $mdec(\mathbf{G}) = \widehat{rdec(\mathbf{G})}$*, where* $rdec(\mathbf{G})$ *is the leafy tree of robust modules of* $\mathbf{G}$*.*

Proof. A module in $mdec(\mathbf{G})$ not in $rdec(\mathbf{G})$ is a subrobust module, hence is the union of the robust modules forming the unique direction D relative to it. We use here Corollary 4.1-3. It can be identified with $\widehat{D}$ in $\widehat{rdec(\mathbf{G})}$. Conversely, the union of the modules forming a direction D is a subrobust module. □

Proposition 5.5. *Every countable modular tree is the tree of the modular decomposition of some countable graph, whose nodes are all of type (I) or (II).*

We omit the proof because we will not need this result. It justifies the terminology "modular tree".

6. The monadic second-order construction of modular decompositions

From now on, we only consider countable graphs, *i.e.* finite or countably infinite graphs.

The objectives of this section are to prove that $Gdec(\mathbf{G})$ (Definition 5.3) and $\mathbf{G}$ can be defined from each other by transformations of relational structures specified by monadic second-order (MS in short) formulas, and thus to obtain that the monadic second-order properties of the modular decomposition of a graph $\mathbf{G}$ are monadic second-order expressible in $\mathbf{G}$. *Monadic second-order logic* and monadic second-order transformations of structures (called *MS transductions*) are reviewed in the appendix.

We only recall here that an MS transduction 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 different relational signatures) such that the domain D_T of T is a subset of $D_S \times \{1, ..., k\}$. (The numbers $1, ..., k$ 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 MS-transduction is *order-invariant* if it uses an auxiliary linear ordering of the domain of the input structure that is of type ω if the domain is infinite, such that for any two such orderings the output structures are isomorphic. These orderings will be denoted by $\preccurlyeq$.

A graph $\mathbf{G}$ is handled as the binary structure $\langle V_{\mathbf{G}}, edg_{\mathbf{G}} \rangle$.

6.1. **The monadic second-order definition of modules.** In this lemma, the types I to IV of robust modules are as in Proposition 5.1, with the same notation.

Lemma 6.1. (1) *There exist MS formulas $\varphi_1(X, Y)$ (resp. $\varphi_2(X, Y)$) such that, for all sets of vertices M, M' of a graph $\mathbf{G}$, $\varphi_1(M, M')$ (resp. $\varphi_2(M, M')$) holds in $\mathbf{G}$ if and only if M is a robust module of type I, (resp. of type II), and M' is one of the corresponding modules C_i.*

(2) *There exists an MS formula $\varphi_3(X, Y, Z)$ such that for all sets of vertices M, M', M'' of a graph $\mathbf{G}$, $\varphi_3(M, M', M'')$ holds in $\mathbf{G}$ if and only if M is a robust module of type (III), M' is a module C_i, M'' is a module C_j and $i < j$.*

(3) *There exists an MS formula $\varphi_4(X, Y, Z)$ such that for all sets of vertices M, M', M'' of a graph $\mathbf{G}$, $\varphi_4(M, M', M'')$ holds in $\mathbf{G}$ if and only if M is a robust module of type (IV), M' is a module C_i, M'' is a module C_j and $u_i \longrightarrow u_j$ in the graph $\mathbf{P}$.*

Proof. Straightforward constructions from the definitions. □

6.2. **Reconstructing trees from their leaves.** Our first objective will be to define by MS formulas a subset W of the set V of vertices of the considered graph $\mathbf{G}$ and a bijection of W onto the set of robust modules that are not singletons. This will give us a definition inside a graph $\mathbf{G}$, and by MS formulas, of the leafy tree

of its robust modules. Then we will complete this tree by using the completion of Proposition 5.3 which is an MS transduction. By Proposition 5.4, we will obtain $mdec(\mathbf{G})$ in this way. The following definitions are adapted from Section 5 of [13].

Definition 6.1. (*The structure $\lambda(T)$ on the leaves of a tree T.*) For a join-tree T, we let $\lambda(T) = \langle Leaves(T), R_T\rangle$ where $R_T(x,y,z)$ is defined to hold if and only if $x \leq y \vee z$, *i.e.* if x belongs to $T_{y\vee z}$.

It is proved in [13] that for a finite leafy tree T, if $Leaves(T)$ is linearly ordered by some auxiliary order $\preccurlyeq$, then T is definable from $(\lambda(T), \preccurlyeq)$[10] by a domain extending MS-transduction. The resulting tree T does not depend on the linear order $\preccurlyeq$. We will generalize this result.

Proposition 6.1. *There exists a domain extending MS transduction that transforms $(\lambda(T), \preccurlyeq)$ into T, whenever T is a leafy tree and $\preccurlyeq$ is an ω-order on the set $Leaves(T)$.*

Hence the mapping which associates a leafy tree T with $\lambda(T)$ is an order-invariant MS-transduction.

Proof. The proof simplifies and generalizes that of Theorem. 5.3 of [13]. Let T be a leafy tree and $\preccurlyeq$ be an ω-order the set of its leaves.

For every *internal* (*i.e.*, non-leaf) node x of T we define :

- $fl1(x)$ as the $\preccurlyeq$-smallest leaf below x, called the *first leaf below x*,
- $D1(x)$ as $dir_x(fl1(x))$, called the *first direction relative to x*,
- $rep(x)$ as the $\preccurlyeq$-smallest leaf below x and not in $D1(x)$ (this is well-defined because in a leafy tree, every internal node has degree at least two).

We call $rep(x)$ the *leaf representing x*. We have $fl1(x) < x$, $rep(x) < x$, and $fl1(x) \prec rep(x)$.

Claim 6.1. Let x, y be two internal nodes. If $rep(x) = rep(y)$ then $x = y$.

Proof. 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 in $D1(x)$. Hence $fl1(x) \prec u$. Also $fl1(x) < x$. Hence u and $fl1(x)$ are in the same direction relative to y. It follows that $rep(y)$ is the $\preccurlyeq$-smallest leaf in a set containing u and $fl1(x)$. Hence $rep(y) \preccurlyeq fl1(x)$ and $rep(y)$ cannot be equal to u. □

The proof of this claim is illustrated in Figure 5.

One defines (up to a bijection) the nodes of T as the elements of a subset of $Leaves(T) \times \{1, 2\}$ (*cf.* the definition of MS transductions in the appendix). Each leaf u is mapped to $(u, 1)$, hence the transduction we construct is domain extending. Each internal node u is mapped to $(rep(u), 2)$.

Claim 6.2. One can write a first-order formula $\alpha(x, y, z)$ such that $(\lambda(T), \preccurlyeq) \models \alpha(x, y, z)$ if and only if $x \neq y$ and $z = rep(x \vee y)$

Proof. We will express the definition of rep by an FO formula. We first observe that an FO formula $\beta(x, y, z)$ can express that $x \neq y$ and $z = fl1(x \vee y)$. Then a formula $\gamma(x, y, u, v)$ can express that $x \neq y$, $u < x \vee y$, $v < x \vee y$, and $u \vee v = x \vee y$, which means that u and v are not in the same direction relative to $x \vee y$. The construction of $\alpha(x, y, z)$ follows easily. □

[10] $(\lambda(T), \preccurlyeq)$ denotes the structure $\langle Leaves(T), R_T, \preccurlyeq\rangle$.

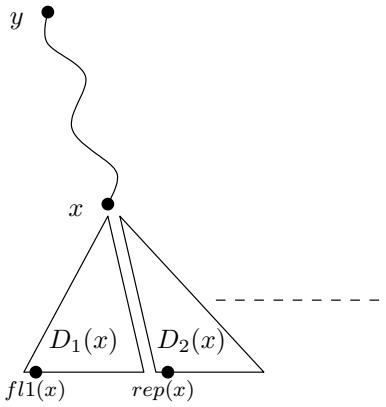

FIGURE 5. The leaf representing of x is the first element of the second direction of x, in other words, the second element of the lexicographically first pair (u, v) of leaves for which $u \vee v = x$ and $u < v$.

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

- $(x, 1) \leq (y, 1)$ if and only if $x = y$,
- $(x, 2) \leq (y, 1)$ is always false,
- $(x, 1) \leq (y, 2)$ if and only if there exist leaves u, v such that $y = rep(u \vee v)$ and $R_T(x, u, v)$ holds,
- $(x, 2) \leq (y, 2)$ if and only if there exist leaves u, v, w, z such that $x = rep(u \vee v)$, $y = rep(w \vee z)$, $R_T(u, w, z)$ and $R_T(v, w, z)$ hold.

Claim 6.3. Ordered in this way, N is isomorphic to T where a leaf u of T corresponds to $(u, 1)$, and an internal node x corresponds to $(rep(x), 2)$.

Proof. The four above clauses 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 $u \vee v$ iff $R_T(x, u, v)$ holds, and that an internal node $u \vee v$ is below (possibly equal to) $w \vee z$ if and only if u and v are both below $w \vee z$. The result follows then from the definition of R_T and the representation of the internal nodes $x \vee y$ by leaves. □

The proof of Proposition 6.1 is then immediate. □

Proposition 6.2. *The mapping rdec associating with a graph* $\mathbf{G}$*,* ω*-ordered by* $\preccurlyeq$*, the leafy tree of its robust modules is a domain extending MS-transduction. The vertices of* $\mathbf{G}$ *are in bijection with the leaves of the tree* $rdec(\mathbf{G})$*.*

The function *rdec* is thus an order-invariant MS-transduction.

Proof. In the leafy tree $T = rdec(\mathbf{G})$, a leaf $S(x, x)$ corresponds to the vertex x of the graph $\mathbf{G}$, an internal node $S(x, y)$ is nothing but $S(x, x) \vee S(y, y)$. The relation $R_T(z, x, y)$ is "$z \in S(x, y)$" which is expressible by an MS formula. Hence $\lambda(T) = \lambda(rdec(\mathbf{G}))$ is definable from $\mathbf{G}$ by an MS-transduction. If in addition $\mathbf{G}$, hence the set of leaves of T, is ω-ordered by $\preccurlyeq$ we can obtain T, *i.e.* $rdec(\mathbf{G})$ from $(\lambda(T), \preccurlyeq)$, by a domain extending MS-transduction, whence from $(\mathbf{G}, \preccurlyeq)$ also

by a domain extending MS-transduction because the composition of two domain extending MS transductions is a domain extending MS transduction. □

6.3. **The monadic second-order definition of modular decompositions and their graph representations.** We know from Proposition 5.4 that $mdec(\mathbf{G})$ is the completion of $rdec(\mathbf{G})$. Our next aim is now to prove that the completion operation defined in Proposition 5.3 is a domain extending MS-transduction, using again an auxiliary ω-ordering.

Proposition 6.3. *There is an order-invariant domain extending MS-transduction that associates with a leafy tree $(T, \leq)$ the modular tree $\widehat{T}$.*

Proof. The technique is similar to the one used in Proposition 6.1. The idea is to represent by some leaf in a well-defined way each direction to be completed (*cf.* the definition of $\widehat{T}$, in Proposition 5.3). The following facts are straightforward to check.

Let $\preccurlyeq$ denote an auxiliary ω-ordering of T.

- There exists an FO formula $\delta(X, x)$ expressing that x is an internal node and X is a direction relative to x.
- There exists an FO formula $\epsilon(X, u)$ expressing that u is the $\preccurlyeq$-smallest leaf in a set of nodes X. We will denote this by $u = fl(X)$.
- There exists an FO formula $\zeta(X, Y, x)$ intended to order linearly the directions relative to x ; we construct it so as it expresses :

$$\delta(X, x) \wedge \delta(Y, x) \wedge \text{"}fl(X) \preccurlyeq fl(Y)\text{"}$$

Since two directions relative to a same node are disjoint sets, this defines a linear ordering of each set $Dir(x, T)$ which is finite or of type ω. A direction is *maximal* if it is the greatest element for this order in its set $Dir(x, T)$.

For every direction X in $Dir(x, T)$ that is not maximal (the existence of such a set X implies that x has degree at least 2), we let $rep - dir(X, x)$ be $fl(Y)$ where Y is the union of the directions relative to x strictly larger than X. Note that $rep - dir(X, x) \notin X$ and $fl(X) \prec rep - dir(X, x)$.

Claim 6.4. The mapping $rep - dir$ is one-to-one.

Proof. Assume $z = rep - dir(X, x) = rep - dir(Y, y)$.
Case 1 : $x = y$. We must have $X = Y$ from the definition of $rep - dir$.
Case 2 : $x \neq y$. We have $z < x$ and $z < y$, hence x and y are comparable. Without loss of generality let us assume that $x < y$. The nodes x and z are in the same direction relative to y, say Z. This direction is strictly larger than Y by the definition of z as $rep - dir(Y, y)$. Hence $z \preccurlyeq fl(Z)$.

From the definition of z as $rep - dir(X, x)$, $fl(X) \prec z$, and $fl(Z) \preccurlyeq fl(X)$ since $X \subseteq Z$. This gives$fl(Z) \prec z$ contradicting a previous observation. Hence we cannot have $x \neq y$. □

In Figure 6, r_i is the leaf representing the direction D_{i-1}. The maximal direction D_{max} is represented by x.

It is then clear that there exists an MS formula $\eta(X, x, y)$ expressing that $X \in Dir(x, T)$, X is not maximal, and $y = rep - dir(X, x)$. We are now ready to construct $\widehat{T}$ from $(T, \leq, \preccurlyeq)$ by an MS transduction.

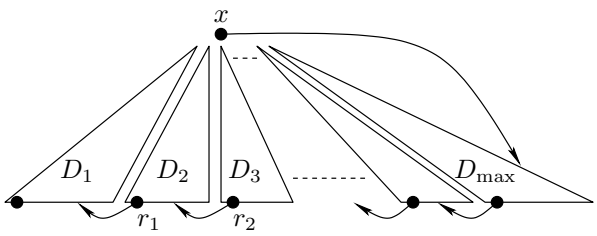

FIGURE 6

We let $N = (T \times \{1\}) \cup (Max_T \times \{2\}) \cup (Dir_T \times \{3\})$, where Max_T is the set of nodes of finite degree at least 2, having a maximal direction such that this maximal direction, say X, has no greatest element, and Dir_T is the set of leaves of the form $rep-dir(X, x)$ such that x has degree at least 2 and the direction X has no greatest element.

The directions X involved in this definition are exactly those for which $\widehat{X}$ must be added to T in order to construct $\widehat{T}$. Since the mapping $rep - dir$ does not represent maximal directions, we treat them separately, as pairs $(x, 2)$. The others are defined as pairs $(y, 3)$ where y is the leaf $rep-dir(Y)$ representing the direction Y.

The order on $\widehat{T}$ defined by Proposition 5.3 is then straightforward to express by MS formulas. This completes the proof of the Proposition. □

Theorem 6.1. *There is an order-invariant domain extending MS transduction γ constructing $Gdec(\mathbf{G})$ from $\mathbf{G}$. There is an FO transduction δ such that $\delta(Gdec(\mathbf{G})) = \mathbf{G}$ for every graph $\mathbf{G}$.*

Proof. We begin with the definition of δ as an FO transduction. The vertices of $\mathbf{G}$ are the leaves of the tree underlying $Gdec(\mathbf{G})$, hence can be identified by FO formulas. Given two vertices x and y of $\mathbf{G}$, whether there is in $\mathbf{G}$ an edge $x \longrightarrow y$ can be determined from the label of $x \vee y$ in $Gdec(\mathbf{G})$ as follows : there is no edge if the label is $\oplus$; there is an edge if the label is $\otimes$, or if it is $\overrightarrow{\otimes}$ and there exist sons u, v of $x \vee y$ such that $u \lhd_{x\vee y} v$[11], $x \leq u$, and $y \leq v$; when $x \vee y$ satisfies case IV, the existence of an edge in $\mathbf{P}$ between the submodules containing x and y is determined by the condition "there exist sons u, v of $x \vee y$ such that $fedg(u, v)$, $x \leq u$, and $y \leq v$".

For proving the first assertion, we express the mapping γ that transforms $(\mathbf{G}, \preccurlyeq)$ into $Gdec(\mathbf{G})$ as the composition of several MS transductions :

(1) The first one transforms :

$$(V_{\mathbf{G}}, edg_{\mathbf{G}}, \preccurlyeq) \text{ into } (V_{\mathbf{G}}, edg_{\mathbf{G}}, \preccurlyeq, R)$$

where R is the ternary relation such that $R(x, y, z) \Leftrightarrow x \in S(y, z)$. Since $V_{\mathbf{G}} = Leaves(rdec(\mathbf{G}))$ and $S(y, z) = y \vee z$ in the tree $rdec(\mathbf{G})$, R is nothing but $R_{rdec(\mathbf{G})}$ (*cf.* Section 6.2). The definition of R by an MS formula is a straightforward translation of the definitions.

[11]The strict linear ordering $\lhd_{x\vee y}$ is defined in Definition 5.3. It is represented in $Gdec(G)$ by the relation $fedg$.

(2) The second MS transduction is based on Proposition 6.1, which gives a domain extending MS transduction transforming $(V_{\mathbf{G}}, \preccurlyeq, R)$ into $rdec(\mathbf{G})$. Hence, by using it we can build an MS transduction that transforms :

$$(V_{\mathbf{G}}, edg_{\mathbf{G}}, \preccurlyeq, R) \text{ into } (V_{\mathbf{G}} \cup N^{int}_{rdec(\mathbf{G})}, edg_{\mathbf{G}}, \preccurlyeq, \leq_{rdec(\mathbf{G})})$$

where $N^{int}_{rdec(\mathbf{G})}$ is the set of internal nodes of $rdec(\mathbf{G})$ and $rdec(\mathbf{G}) = (V_{\mathbf{G}} \cup N^{int}_{rdec(\mathbf{G})}, \leq_{rdec(\mathbf{G})})$.

(3) The third transformation uses Proposition 5.3 and Lemma 4.4-3. Since $mdec(\mathbf{G}) = \widehat{rdec(\mathbf{G})}$, we have an MS transduction that transforms :

$$(V_{\mathbf{G}} \cup N^{int}_{rdec(\mathbf{G})}, edg_{\mathbf{G}}, \preccurlyeq, \leq_{rdec(\mathbf{G})}) \text{ into } (V_{\mathbf{G}} \cup N^{int}_{mdec(\mathbf{G})}, edg_{\mathbf{G}}, \leq_{mdec(\mathbf{G})})$$

The output does not contain the linear order $\preccurlyeq$ because it is not used in the next step.

(4) The last MS transduction transforms $(V_{\mathbf{G}} \cup N^{int}_{rdec(\mathbf{G})}, edg_{\mathbf{G}}, \leq_{mdec(\mathbf{G})})$ into $Gdec(\mathbf{G})$. Its definition is a straightforward translation from the definition using Lemma 6.1.

By composing these four MS transductions, one gets the desired domain extending MS transduction that transforms $(V_{\mathbf{G}}, edg_{\mathbf{G}}, \preccurlyeq)$ into $Gdec(\mathbf{G})$. □

Corollary 6.1. (1) *The first-order (resp. monadic second-order) properties of a graph* $\mathbf{G}$ *are first-order (resp. monadic second-order) properties of the structure* $Gdec(\mathbf{G})$.

(2) *For any graph* $\mathbf{G}$*, the monadic second-order properties of* $Gdec(\mathbf{G})$ *are order-invariant monadic second-order properties of* $\mathbf{G}$.

Proof. Immediate from Theorem 6.1 and Proposition 9.1 from the Appendix. □

7. Representing modular decompositions by low degree relational structures

Our objective in this section is to represent trees and modular decompositions by relational structures of *lowest possible degree* (the notion of degree is as for graphs) by generalizing the observation that the *dense* structure $(\mathbb{Q}, \leq)$ is isomorphic to the set of nodes of the complete infinite binary tree, a graph of degree 3 ordered appropriately. We will extend to countable linear orders a construction of [17] that builds on the elements of a finite linear order a structure of binary tree such that the associated inorder on nodes is the given linear order. Furthermore, this construction can be done by an MS-transduction using an auxiliary ω-ordering of the given set.

Let us give some motivations for this investigation.

For finite objects like graphs and partial orders, space efficient representations are of interest. For an example, every finite partial order can be represented by its Hasse diagram, which may contain $O(m^{1/2})$ edges whereas its directed graph has m edges. The modular decomposition of a finite graph has a graph representation with at most $3n + m$ edges where n is the number of vertices and m is the total number of edges of its prime factors. In both cases the original partial order (or graph) can be determined from its Hasse diagram (or its modular decomposition) by computations of transitive closures, hence by MS transductions.

The motivation of getting space efficient representations does not apply to infinite graphs, but bounds on degrees of infinite structures are nevertheless interesting because they yield structural properties. For example, every "equational graph" of bounded degree is "prefix recognizable", see Caucal [5], (or Barthelmann [1] for a similar result). As another result, MS logic with edge set quantifications is as powerful as MS logic without them on relational structures with sparse relations, see [16].

In the present section, we define mutual transformations of relational structures : trees, graphs, binary rooted trees, and in each case we prove they are MS transductions.

7.1. Representing linear orders by standard binary trees.

We first consider the case of linearly ordered sets.

Definition 7.1. (*Standard binary trees.*) By a *standard binary tree*, we mean a simple directed edge-labeled graph $T = (N_T, lson_T, rson_T)$ where N_T is the finite or countable set of nodes, $lson_T$ and $rson_T$ are two binary functional relations defining for each node its *left son* and its *right son.* A node may have no son, two sons, or only a right son or only a left son. The *root* is the unique node of indegree 0 and every node is reachable from it by a unique directed path. Whether a structure $A = (D_A, lson_A, rson_A)$ is a standard binary tree can be expressed in MS logic.

For a standard binary tree T, and $x, y \in N_T$, we will write $x \longrightarrow_l y$ if y is the left son of x, $x \longrightarrow_r y$ if y is the right son of x, and $x \longrightarrow y$ if y is the left or the right son of x.

A linear order, the *in-order*, on N_T can be defined as follows.

$x \leq_{in,T} y$ if and only if $x = y$ or $x \longrightarrow_r z \longrightarrow^* y$ or $y \longrightarrow_l z \longrightarrow^* x$ for some z, or $t \longrightarrow_l z \longrightarrow^* x$ and $t \longrightarrow_r z' \longrightarrow^* y$ for some t, z, z'.

We let $\Omega(T)$ denote the linearly ordered set $(N_T, \leq_{in,T})$. It is clear that the mapping Ω is an MS-transduction because the transitive closure of given binary relation is expressible by an MS formula. Our objective is to construct T from $\Omega(T)$ by an MS transduction.

Proposition 7.1. *([17]) There exist two first-order formulas $\lambda(x,y)$ and $\rho(x,y)$ that define in every structure $(N, \sqsubseteq, \preccurlyeq)$ such that $\sqsubseteq$ is a linear order and $\preccurlyeq$ is an ω-order, two binary relations $lson$ and $rson$ such that $(N, lson, rson)$ is a standard binary tree T, the root of which is the $\preccurlyeq$-least element of N and such that $\Omega(T) = (N, \sqsubseteq)$. This tree T is defined from $(N, \sqsubseteq, \preccurlyeq)$ by an FO transduction.*

The formulas λ and ρ do not depend on N.

Proof. Let $(N, \sqsubseteq, \preccurlyeq)$ be given as in the statement. We leave out the case where N is finite, for which the construction is immediate, without using $\preccurlyeq$.

We will take r, the $\preccurlyeq$-least element of N as root of the tree T we are constructing.

For every x in N, $x \neq r$, we let

- $m(x)$ be the $\sqsubseteq$-largest element y such that $y \sqsubset x$ and $y \prec x$,
- $M(x)$ be the $\sqsubseteq$-smallest element y such that $x \sqsubset y$ and $y \prec x$.

We have $m(x) \sqsubset x$ and $x \sqsubset M(x)$ whenever $m(x)$ or $M(x)$ is defined.

(a) If $M(x)$ is undefined, we let $(m(x), x)$ belong to $rson$.
(b) If $m(x)$ is undefined, we let $(M(x), x)$ belong to $lson$.
If $M(x)$ and $m(x)$ are both defined, we have $m(x) \sqsubset x \sqsubset M(x)$ and
(c) if $m(x) \prec M(x)$ we let $(M(x), x)$ belong to $lson$ and finally
(d) if $M(x) \prec m(x)$ we let $(m(x), x)$ belong to $rson$.

This can be formalized by first-order formulas λ and ρ defining $lson$ and $rson$.

We make some observations to help the understanding of the forthcoming proof. For every pair (y, x) in $lson$ or in $rson$, y is before x in the enumeration defined by $\preccurlyeq$. This guarantees the absence of circuits.

The construction consists in putting in a tree the elements of N in the order defined by $\preccurlyeq$. There are four ways to add a node x :

- as right son of the rightmost node $m(x)$, by clause a) above.
- as left son of the leftmost node $M(x)$, by clause b) above ;
- in cases c) and d) the node x must be placed between $m(x)$ and $M(x)$ which are the elements of N before x with respect to $\preccurlyeq$ which are closest to x with respect to $\sqsubseteq$ (they already exist in the tree). Depending on whether $m(x) \prec M(x)$ or $M(x) \prec m(x)$, x is defined as left son of $M(x)$ (case c)), or right son of $m(x)$ (case d) ;

It remains to prove that $(N, lson, rson)$ is actually a tree T, and that $\Omega((N, lson, rson)) = (N, \sqsubseteq)$.

For every x in N we let $T(x)$ be the restriction of the structure $(N, lson, rson)$ to the set $A(x) = \{y : y \preccurlyeq x\}$.

Claim 7.1. $T(x)$ is a finite standard binary tree and $\Omega(T(x)) = (A(x), \sqsubseteq)$.

Proof. By induction on the order $\preccurlyeq$. The least element of N is r. The tree $T(r)$ is reduced to r and the assertion holds.

Consider $x \neq r$ and x', its predecessor with respect to $\preccurlyeq$. Hence $T(x')$ satisfies the property.

From the definitions, x is the second component of a unique pair (y, x) either in $lson$ or in $rson$, and furthermore, $y \prec x$.

We review the different cases.

(1) In Case a), $y = m(x)$, $M(x)$ is undefined, we have $m(x) \sqsubset x$. Then y has no right son in $T(x')$ because otherwise, if it had one say z, then either $x \sqsubset z$ and $M(x)$ would be defined or $z \sqsubset x$ and $m(x)$ would not be y (y would not be the $\sqsubset$ -predecessor of x in $A(x)$)

Hence by setting x as right son of y, we get a standard binary tree $T(x)$ satisfying $\Omega(T(x)) = (A(x), \sqsubseteq)$.

(2) In Case b) the argument is the same by exchanging left and right, and $m(x)$ and $M(x)$.

In the next two cases $M(x)$ and $m(x)$ are both defined and we have $m(x) \sqsubset x \sqsubset M(x)$. This means that x must be inserted "between" $m(x)$ and $M(x)$ which are consecutive in $(A(x'), \sqsubseteq)$. Furthermore, in $T(x')$, $m(x)$ is an ancestor of $M(x)$ or vice-versa, because otherwise, they have a common ancestor, say z, $m(x) \sqsubset z \sqsubset M(x)$ hence they are not consecutive in $(A(x'), \sqsubseteq)$.

(3) If $m(x) \prec M(x)$ (case c)), we have $y = M(x)$, x is set as left son of y, $m(x)$ is an ancestor of $M(x)$. (This cannot be the reverse because $m(x) \prec M(x)$). Assume y has already a left son, say z, in $T(x')$. Either $x \sqsubset z$ and $M(x)$ would not be y (because $x \sqsubset z \sqsubset y$) or $z \sqsubset x$ but then $m(x) \sqsubset z$ (because $m(x)$ is an ancestor of $M(x)$, so that $M(x)$ is, or is below, the right son of $m(x)$) but this contradicts the definition of $m(x)$. Hence we can set x as left son of y. And we get thus a tree $T(x)$ satisfying $\Omega(T(x))= (A(x), \sqsubseteq)$.

(4) If $M(x) \prec m(x)$ (case d)) the proof is fully similar.

□

We now complete the proof of Proposition 7.1. One takes for T the union of the standard binary trees $T(x)$ which extend one another. Every x in N is a node of this tree, because the isomorphism type of $\preccurlyeq$ is ω, hence it is added at some step. The tree T is thus $(N, lson, rson)$. We have noted that $lson$ and $rson$ are definable by first-order formulas. This completes the proof of the Proposition. □

This definition is illustrated on Figure 7. We show the tree associated with the linear order $N = (9, 8, 1, 6, 0, 7, 2, 5, 3, 4)$, where the numbers define the enumeration.

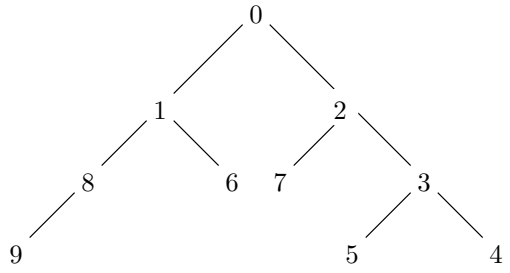

FIGURE 7. The tree of a linear order N

The transduction of Proposition 7.1 is not order-invariant.

This proposition proves that one can represent a linear order by a suitable ordering of the set of all nodes of a standard binary tree that is not necessarily complete. We now prove that one can represent it by the same ordering on the set of *leaves* of a standard binary tree. Furthermore, this tree is definable by an FO transduction, as in Proposition 7.1.

Lemma 7.1. *There exists a domain extending FO transduction that transforms a standard binary tree T into a standard binary tree U such that $(Leaves(U), \leq_{in,U})$ is isomorphic to $(N_T, \leq_{in,T})$.*

Proof. We first describe the construction in set theoretical terms. Then we will show that it can be performed by an FO transduction.

Given T, we construct U such that $N_T \subseteq N_U$ and a mapping h from N_T to $Leaves(U)$ intended to be the desired isomorphism.

For constructing U, we consider each node x of T. There are several cases (see the first two columns of Figure 8).

- Case 1 : x is a leaf. It remains a leaf in U. We let $h(x) = x$.
- Case 2 : x has a left son y and no right son. We add a new node x' as right son in U of x, and we let $h(x) = x'$.
- Case 3 : x has a right son z and no left son. We add a new node x' as left son in U of x, and we let $h(x) = x'$.
- Case 4 : x has a left son y and a right son z. We add two nodes, an internal node x'' and a leaf x' such that in U :
  - x'' is the right son of x,
  - z is the right son of x'',
  - y is the left son of x, as in T,
  - x' is the left son of x'',

  and we let $h(x) = x'$.

These transformations can be done simultaneously for all nodes x of T, giving U and h is a bijection of N_T onto $Leaves(U)$.

It is clear that $h(x) \leq_{in,U} h(y)$ if and only if $x \leq_{in,U} y$.

It remains to define U from T by an FO transduction. This transduction will specify N_U as a subset of $N_T \times \{1, 2, 3\}$. Furthermore, we will do that in such a way that the above defined mapping h is actually $h(x) = (x, 1)$ for every x in N_T. In order to reach this goal, the new nodes x'' in case 4 will be defined as pairs $(x, 3)$. The nodes x' in cases 2,3,4 will be defined as pairs $(x, 1)$. The nodes x in cases 2,3,4 will be defined as pairs $(x, 2)$ as nodes of U. The nodes x in case 1 will be defined as pairs $(x, 1)$ as nodes of U. (Note in particular that, if r is the root of T, the root of U is $(r, 1)$ if T is reduced to a single node, and $(r, 2)$ otherwise.) See the last column of the table of Figure 8. We denote by $\overline{y}$ and $\overline{z}$ the results of this transformation applied to y and to z.

There exist first-order formulas $\gamma_i(x)$, $i = 1, ..., 4$ such that $T \models \gamma_i(x)$ if and only if the node x is of the type of Case i.

We show how first-order formulas can be constructed which specify

$$rson_U((u, i), (v, j)) \text{ and } lson_U((u, i), (v, j))$$

in the desired way.

Let us consider which pairs $((u, i), (v, j))$ must be put in $rson_U$ and in $lson_U$ by the specification of Case 4.

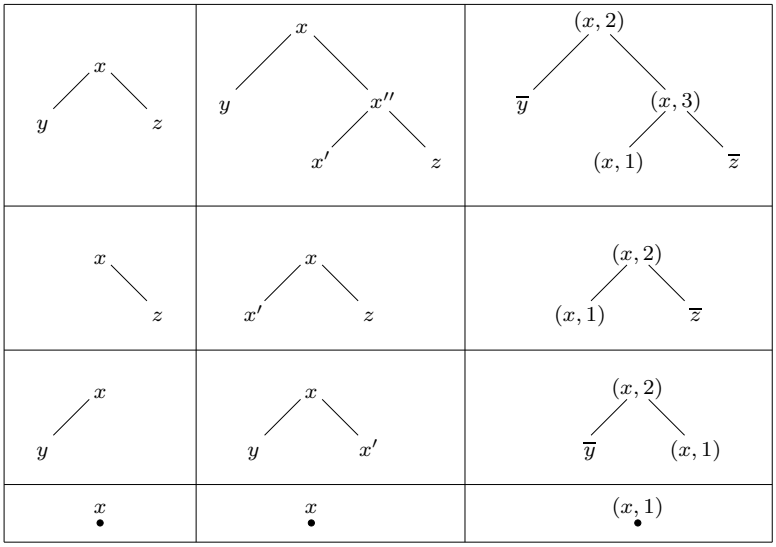

FIGURE 8

Let x satisfy γ_4 (hence is of the type of Case 4) with left son y and right son z in T. Then we must have

(x, x'') and (x'', z) in $rson_U$, and (x, y) and (x'', x') in $lson_U$. From the way nodes of U are defined as pairs (u, i), this gives :

$((x, 2), (x, 3))$ and $((x, 3), \overline{z})$ in $rson_U$, and $((x, 2), \overline{y})$ and $((x, 3), (x, 1))$ in $lson_U$. If Case 1 applies to y, then $\overline{y}$ is $(y, 1)$, otherwise $\overline{y}$ is $(y, 2)$, and similarly for z.

Collecting all cases, we get the following description of $rson_U$: $rson_U((u, i), (v, j))$ holds if and only if

- either $u = v \wedge i = 2 \wedge j \in \{1, 3\}$,
- or $u \neq v, rson_T(u, v)$ and we have the following cases :
  - either : $\gamma_3(u) \wedge \gamma_1(v) \wedge i = 2 \wedge j = 1$,
  - or : $\gamma_3(u) \wedge (\gamma_2(v) \vee \gamma_3(v) \vee \gamma_4(v)) \wedge i = j = 2$,
  - or : $\gamma_4(u) \wedge \gamma_1(v) \wedge i = 3 \wedge j = 1$,
  - or : $\gamma_4(u) \wedge (\gamma_2(v) \vee \gamma_3(v) \vee \gamma_4(v)) \wedge i = 3 \wedge j = 2$.

The characterization of $lson_U$ is similar. The translation into first-order formulas specifying an FO transduction is then a routine task. □

To conclude this subsection we apply these two results to build an MS transduction that transforms a diagram-connected ordered tree into a standard binary tree. This transformation is classical, but our effort consists in defining it as an MS transduction.

Definition 7.2. (*Ordered trees.*) A join-tree $(T, \leq)$ is *ordered* if it is equipped with a strict linear ordering $\lhd_x$ on each set $Dir(x, T)$. These strict linear orderings yield a linear ordering $\sqsubseteq$ extending $\leq$ given by $x \sqsubseteq y$ if and only if $x \leq y$ or x and y are incomparable and $dir_{x \vee y}(x) \lhd_{x \vee y} dir_{x \vee y}(y)$.

According to Definition 5.3, $\lhd_x$ is defined only for nodes representing linear sums. For simplifying certain constructions, we will choose for $\lhd_x$ an arbitrary linear order

for all other father nodes x. All our logical constructions use a ω-ordering $\preccurlyeq$ on the given graph. This (arbitrary) ω-order can be used to specify non-ambiguoulsy the linear orderings $\lhd_x$ for nodes of types I, II and IV. We denote in the same way by $\lhd_x$ the ordering of the sons of x and of its directions.

Definitions 7.3. (*Compressing and stretching trees.*) Let T be a diagram-connected ordered tree. Its *son* relation is sufficient to define (by taking the transitive closure) the ancestor relation because T is diagram-connected. Hence, without loss of information, T can be represented by the logical structure $(N_T, son_T, order_T)$ where N_T is its set of nodes, son_T and $order_T$ are binary relations such that $son_T(x, y)$ holds if and only if y is a son of x, and $order_T(x, y)$ holds if and only if x and $y \neq x$ have the same father, say z, and $dir_z(x) \lhd_z dir_z(y)$ (*cf.* Definitions 5.3 and 7.2).[12]

Our intention is to represent T by a structure $U = (D_U, node_U, lson_U, rson_U)$ which is a standard binary tree (equipped with a unary predicate *node*) such that :

- N_T is the set of elements x of D_U that satisfy $node_U(x)$,
- $son_T(x, y)$ holds if and only if $node_U(x)$, $node_U(y)$, and $x \longrightarrow^+ y$ (we use the notation of Definition 7.1) hold, and the elements on the path $x \longrightarrow^+ y$ are not in N_T except x and y.
- and $order_T(x, y)$ holds if and only if $node_U(x)$, $node_U(y)$, and $u \longrightarrow_l w \longrightarrow^* x$, $u \longrightarrow_r w' \longrightarrow^* y$ hold for some u, w, w', where the elements on the paths $w \longrightarrow^* x$, $w' \longrightarrow^* y$ are not in N_T, except x and y.

In such a situation, we will say that U is a *stretching* of T and that T is *the compression* of U. We will write $T = Compress(U)$.

Proposition 7.2. *There exists a domain extending MS transduction σ that defines an ω-ordered stretching of any rooted, diagram-connected ordered and ω-ordered tree T. The mapping Compress is an MS transduction.*

Proof. Our aim is to construct an MS transduction σ such that $\sigma(T, \preccurlyeq)$ is a stretching of T and $Compress(\sigma(T, \preccurlyeq)) = T$ for every rooted, diagram-connected ordered tree T ω-ordered by $\preccurlyeq$. From its definition it is clear that *Compress* is an MS transduction.

We now define σ. By composing the transductions of Proposition 7.1 and Lemma 7.1, we get a domain extending MS transduction that associates with a linear order $(N, \sqsubseteq)$ that is also ω-ordered by $\preccurlyeq$, a standard binary tree U such that the structure $(Leaves(U), \leq_{in,U})$ is isomorphic to $(N, \sqsubseteq)$. In this construction, the root of U is the pair $(t_0, 2)$ where t_0 is the $\preccurlyeq$-smallest element of N, because this is the root of the tree constructed by Proposition 7.1 and the root r of a tree T transformed into $(r, 2)$ by Lemma 7.1. See Figure 8. We will denote this tree by $\tau(N, \sqsubseteq, \preccurlyeq)$. Consider a diagram-connected ordered tree T given by the structure $(N_T, son_T, order_T)$.

The structure U is constructed from T as follows :

(1) to each element x of N_T corresponds the element $(x, 1)$ of D_U and we let it belong to $node_U$,
(2) for each internal node x of T, its set of sons $Sons_T(x)$ is linearly ordered by $order_T$, and we construct the tree $\tau(Sons_T(x), order_T, \preccurlyeq)$, by using the domain extending MS transduction τ, with the slight modification that the

[12]The relation $order_T$ is the union of the relations $\lhd_z$. It is usually not an order relation.

root of $\tau(Sons_T(x), order_T, \preccurlyeq)$ is defined as $(x, 1)$ and not as $(y, 2)$ where y is the $\preccurlyeq$-smallest son of x.

Hence, the construction of U consists in introducing new nodes and in replacing the edges from fathers to their sons in T by paths through these new nodes. The edges of these paths are of types left or right, in such a way that the in-order on these paths represents the left-right order of sons. If a node has out-degree 1 or 2, then no node is inserted between it and his son or his sons. In particular, a standard binary tree is not modified. The root of U is that of T.

It is clear that this construction is a domain extending MS transduction.

The statement of the proposition demands also an ω-order on the constructed structure. But Lemma 9.1 in the appendix shows that an MS transduction taking as input ω-ordered structures can be equipped so as to define also an ω-order on the output structures.

As final remark, consider a structure $U = (D_U, node_U, lson_U, rson_U)$. It is of the form $\sigma(T, \preccurlyeq)$ for some ω-ordered diagram-connected rooted ordered tree T if and only if it is a standard binary tree, and every infinite path in U starting at the root contains infinitely many nodes in $node_U$. This characterization is MS expressible. □

Figure 9 shows a tree T and one of its stretchings U. The nodes of T, and their copies in U are indicated by •. The "new nodes" are indicated by ∘. The order type of the set of sons of the root is ζ, that of integers.

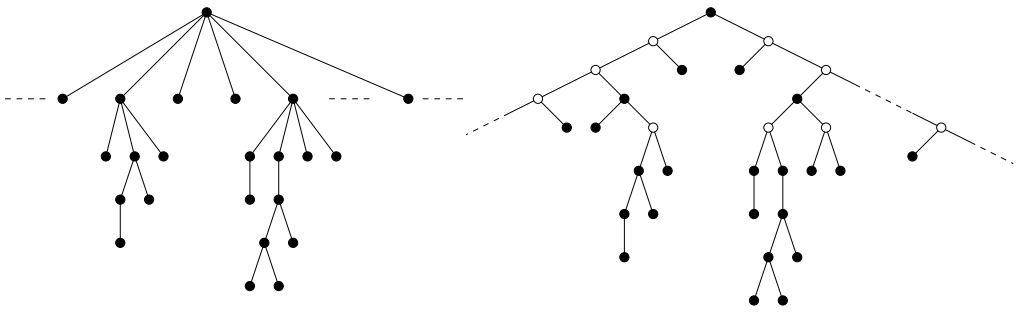

FIGURE 9

7.2. **A universal ordered join-tree.** It is well known that the linearly ordered set $\mathbb{Q}$ is *universal for finite and countable linear orders* : it embeds each of them and is itself countable. We will construct a *universal ordered tree*, where universality is relative to join-embeddings. We will use it in the next subsection to represent modular decompositions by low degree relational structures.

Our objective is to define a universal ordered tree. As one may expect, the set $\mathbb{Q}$ will play a prominent role in this construction.

Definition 7.4. (*Ordered trees constructed from linear orders.*) We let S and D be two nonempty disjoint linearly ordered sets. We let $Aseq(S, D)$ denote the set of alternating sequences of the form :

$$s_1 d_1 s_2 d_2 ... d_k s_{k+1}$$

for $k \geq 0$, with s_i in S and d_j in D for all i, j. These sequences have at least one occurrence of an element in S.

We order $Aseq(S, D)$ by $\leq_{S,D}$ defined as follows :

$w \leq_{S,D} u$ if and only if $u = u's, w = u's'w'$ for some u' in $(SD)^*$, some w' in $(DS)^*$, some s, s' in S with $s' \leq_S s$.

In particular, $w \leq_{S,D} u$ whenever $u \leq_{pref} w$ (where $\leq_{pref}$ denotes the prefix order on sequences). The linear ordering on D will be used in Definition 7.5 below.

Lemma 7.2. *The ordered set $(Aseq(S, D), \leq_{S,D})$ is a join-tree, denoted by $T(S, D)$.*

Proof. That $(Aseq(S, D), \leq_{S,D})$ is a tree is easy. We check the existence of joins.

Let x and y be incomparable. They have a longest common prefix u, and we have two cases :

- either $x = udw, y = ud'w', d, d' \in D, d \neq d', w, w' \in (SD)^*S$ and then $x \vee y = u$,
- or $x = usw$, $y = us'w', s, s' \in S, s \neq s'$, $w, w' \in (DS)^*$, and then $x \vee y = uMax\{s, s'\}$, as one checks from the definitions.

□

The following lemma is easy to prove from the definitions.

Lemma 7.3. *The directions in $T(S, D)$ relative to a node us (for $u \in (SD)^*$, $s \in S$) are the nonempty sets of the following forms :*

$$D(0, us) = \{us'w : s' \in S, s' <_S s, w \in (DS)^*\}, \text{ for } s \neq Min(S) \text{ and}$$
$$D(d, us) = \{usdw : w \in S(DS)^*\} \text{ for } d \in D.$$

The direction $D(0, us)$ is called the main direction *relative to us.*[13]

Definition 7.5. (*Making $T(S, D)$ into an order-invariant tree.*) For disjoint linearly ordered sets $(A, \leq_A)$ and $(B, \leq_B)$, we denote by $A + B$ the linearly ordered set $(A \cup B, \leq)$ where $x \leq y$ if and only if $x \leq_A y$ or $x \leq_B y$ or $x \in A$ and $y \in B$.

We let D_+ and D_- be two disjoint linearly ordered sets and $D = D_- + D_+$, and we make the join-tree $T(S, D)$ into an ordered tree by ordering directions as follows :

$D(d, us) \lhd_{us} D(0, us) \lhd_{us} D(d', us)$ for $d \in D_-, d' \in D_+$, for $s \neq Min(S)$, and

$D(d, us) \lhd_{us} D(d', us)$ for $d, d' \in D$, where $d < d'$.

We obtain an ordered tree denoted by $UT(S, D_-, D_+)$. We denote by $Pref(L)$ the set of prefixes of a subset L of $Aseq(S, D)$. With these notations :

Lemma 7.4. *For every nonempty subset L of $Aseq(S, D)$ such that $Pref(L) \cap Aseq(S, D) \subseteq L$, the triple $(L, \leq_{S,D}, \lhd)$ is an ordered tree join-embeddable into $UT(S, D_-, D_+)$.*

(In the triple $(L, \leq_{S,D}, \lhd)$, $\leq_{S,D}$ denotes actually the restriction to L of the order $\leq_{S,D}$ on $Aseq(S, D)$ and similarly for $\lhd$.)

Proof. Easy □

[13] We denote by $Min(S)$ the smallest element of S if it exists. If $s = Min(S)$, then $D(0, us)$ is undefined.

We let $\mathbb{Q}_-$ be the set of negative rational numbers and $\mathbb{Q}_+$ be the set of positive ones. They are both order-isomorphic to $\mathbb{Q}$, but it is more convenient to distinguish them. The universal tree $\mathbf{U} = UT(\mathbb{Q}, \mathbb{Q}_-, \mathbb{Q}_+)$ is shown on Figure 10.

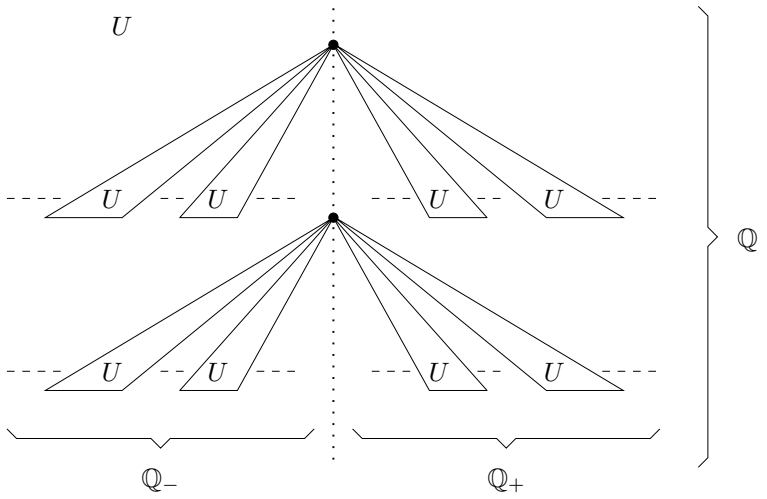

FIGURE 10. The universal tree U

Theorem 7.1. *Every ordered tree join-embeds into the tree* $\mathbf{U} = UT(\mathbb{Q}, \mathbb{Q}_-, \mathbb{Q}_+)$.

Proof. Since the case of a finite tree is trivial, we consider an infinite ordered tree $(T, \leq, \lhd)$, ω-ordered by $\preccurlyeq$, with corresponding enumeration denoted by $t_0, ..., t_n, ...$ We define a structuring of T that depends on this enumeration and associates a finite depth with each node. This structuring will be the basis of a representation of ordered trees by "usual" binary trees that will be considered in Section 7.

Hence we begin by some definitions depending on $\preccurlyeq$. First, we associate with every $x \in T$ a subset $U(x)$ characterized as follows :

(1) $U(x)$ is a maximal chain and contains x,
(2) it is lexicographically minimal with this property, which means that for every maximal chain W containing x and different from $U(x)$, the $\preccurlyeq$-smallest element of $U(x) \vartriangle W$ defined as $(U(x) \backslash W) \cup (W \backslash U(x))$ belongs to $U(x)$.

For each x, there exists such a set $U(x)$: we let $J(x)$ be the increasing sequence of integers $i_0 < i_1 < ... < i_n < ...$ defined inductively as follows :

- i_0 is the smallest j such that x and t_j are comparable,
- i_{n+1} is the smallest $j > i_n$ such that $x, t_{i_0}, ..., t_{i_n}, t_j$ form a chain.

We let $U(x) = \{x, t_{i_0}, ..., t_{i_n}, ...\}$.

Claim 7.2. $U(x)$ is a maximal chain.

Proof. Any two nodes of $U(x)$ are comparable, otherwise some set $\{x, t_{i_0}, ..., t_{i_n}\}$ is not a chain, contradicting the definition. If $U(x)$ is not a maximal chain, there is a node y such that $U(x) \cup \{y\}$ is a chain. Let $y = t_j$ we have $i_n < j < i_{n+1}$ for some n. From the definition of i_{n+1}, $i_{n+1} = j$ should hold : a contradiction. Hence $U(x)$ is a maximal chain. □

It is clear that Conditions (1) and (2) define $U(x)$ in a unique way. (Because if another set W also satisfies them, the $\preccurlyeq$-smallest element of $U(x) \triangle W$ must be in W and in $U(x)$: a contradiction.)

Claim 7.3. If $y < z$ and $y \in U(x)$, then $z \in U(x)$.

Proof. Every maximal chain in a tree is an up-set of that tree. □

Claim 7.4. If $y < x$ and $y \in U(x)$, then $U(y) = U(x)$.

Proof. $U(x)$ is a maximal chain containing y. If W is a maximal chain containing y, thus it contains also x by the proof of Claim 7.3. Hence the $\preccurlyeq$-smallest element of $W \triangle U(x)$ belongs to $U(x)$. Hence, $U(x)$ satisfies the conditions characterizing $U(y)$, and thus $U(y) = U(x)$. □

Claim 7.5. If $U(x) \neq U(y)$, then $U(x) \cap U(y) = T^z$ for some z. If x and y are incomparable, then $z = x \vee y$.

Proof. We first note that $U(x) \cap U(y) \neq \varnothing$ because, since T is a join-tree, $x \vee y \in U(x) \cap U(y)$.

Since $U(x)$ and $U(y)$ are incomparable, there are u in $U(x) \backslash U(y)$ and v in $U(y) \backslash U(x)$, which are incomparable by Claim 7.3. Also by Claim 7.3, $T^{u \vee v} \subseteq U(x) \cap U(y)$. For the converse let w belong to $U(x) \cap U(y)$. It is comparable to u and to v because $U(x)$ and $U(y)$ are chains. There are several cases : if $u, v \leq w$ then $u \vee v \leq w$, which gives $w \in T^{u \vee v}$ as wanted. The other cases are $u, v \geq w$, $u \leq w \leq v$, $v \leq w \leq u$, but u and v would be comparable. Hence they are excluded. This proves the first assertion with $z = u \vee v$.

If x and y are incomparable, then $x \in U(x) \backslash U(y)$ and $y \in U(y) \backslash U(x)$ (because if $x \notin U(x) \backslash U(y)$, then $x \in U(x) \cap U(y)$ and x and y are comparable). The first part gives the result. □

We need some more definitions.

We let $\mathcal{U} = \{U(x) : x \in T\}$. For U in $\mathcal{U}$, we let $a(U)$ be the $\preccurlyeq$-smallest x such that $U(x) = U$. By Claim 7.5, for any three sets U, U', U'' in $\mathcal{U}$, the sets $U \cap U'$ and $U \cap U''$ are comparable for inclusion.

Thus for U in $\mathcal{U}$, different from $U(t_0)$, we can define $p(U)$ (for "predecessor of U") as the set $W \in \mathcal{U}$ such that, under the condition that $a(W) \prec a(U)$, the set $U \cap W$ is largest possible, and among those, $a(W)$ is the $\preccurlyeq$-smallest.

We also define U_-, U_+ and $b(U)$ by $U_- = U \backslash p(U)$ and $U_+ = U \backslash U_- = U \cap p(U) = T^{b(U)}$.

Note that $U = U_- + U_+$ and $b(U) \in U_+, b(U) \notin U_-$.

We also define the *depth of* U as follows : $d(U) = 0$ if $U = U(t_0)$, $d(U) = d(p(U)) + 1$ otherwise.

Letting $ia(U)$ denote the integer i such that $a(U) = t_i$, we have :

Claim 7.6. The integers $ia(U)$ and $d(U)$ are well-defined and $d(U) \leq ia(U)$ for all U in $\mathcal{U}$.

Proof. Since every x in T is equal to some t_i, the integer $ia(U)$ is well-defined for all U. We now use induction on $ia(U)$. We have $d(U) = 0$ if $ia(U) = 0$. Otherwise : $d(U) = d(p(U)) + 1 \leq ia(p(U)) + 1 \leq ia(U))$, since $ia(p(U)) < ia(U)$ by the definition of $p(U)$. □

Claim 7.7. For any two distinct sets U, W in $\mathcal{U}$, we have $U_- \cap W_- = \varnothing$.

Proof. Without loss of generality, let $a(W) \prec a(U)$. We have $U \cap W \subseteq U \cap p(U)$ by the definition of $p(U)$. Hence $W_- \subseteq p(U)$ hence $U_- \cap W_- = \varnothing$. □

Finally, for every x in T, $x \notin U(t_0)$, we define $W_-(x) = W_-$, $p(x) = b(W)$, $d(x) = d(W)$, where W is the unique set (*cf.* Claim 7.7) such that $x \in W_-$. We call $d(x)$ the *depth* of x.

Claim 7.8. For all $x \notin U(t_0)$: $W_-(x)$, $p(x)$, and $d(x)$ are well-defined, $d(x) = d(p(x)) + 1$ and $x < p(x)$.

Proof. Clearly, x belongs to some W with minimal $ia(W)$. Then $x \in W_-$ because otherwise, $x \in p(W)$, and $ia(p(W)) < ia(W)$, contradicting the choice of W.

We have thus : $d(x) = d(W) = d(p(W)) + 1$ and $d(b(W)) + 1 = d(p(x)) + 1$.

We need only prove that $d(p(W)) = d(b(W))$. This is true because $b(W) \in p(W)_-$, which we prove as follows. If this were not the case, we would have $b(W) \geq b(p(W))$. But we would have $W \cap p(W) = W \cap p(p(W))$ but $ia(p(p(W))) < ia(p(W))$, hence $p(W)$ would not be of minimal index with the property that $W \cap p(W)$ is largest possible.

Hence we have $d(x) = d(p(x)) + 1$.

Finally $x < p(x)$ because $x \in W_-$ and $p(x) = b(W)$. □

For every z, we let $Main(z)$ be the main direction relative to it, *i.e.*, the unique one which meets $W_-(z)$.

Claim 7.9. For every $x, y \in T$, either $W_-(x) \cap W_-(y) = \varnothing$, or $W_-(x) = W_-(y)$, $p(x) = p(y)$ and $dir_{p(x)}(x) = dir_{p(x)}(y)$.

If $W_-(x) \cap W_-(y) = \varnothing$ and $p(x) = p(y)$ then $dir_{p(x)}(x) \cap dir_{p(x)}(y) = \varnothing$.

Proof. The first assertion follows from Claim 7.7. The fact $dir_{p(x)}(x) = dir_{p(x)}(y)$ is clear since $x, y < p(x) = p(y)$. Since x and y belong to a same chain, they are comparable hence in the same direction relative to $p(x)$.

In the second assertion, if $z \in dir_{p(x)}(x) \cap dir_{p(x)}(y)$, then $z \in W_-(x) \cap W_-(y)$. □

Claim 7.10.
(1) If x and y are incomparable, and x is not in the main direction relative to $x \vee y$ then $x \vee y = p^n(x)$ for some $n > 0$.
(2) If x and y are incomparable, and x is in the main direction relative to $x \vee y$ then $x \vee y > p^n(x)$ for some $n \geq 0$, such that $p^n(x) \in W_-(x \vee y)$.

Proof.
(1) The proof is by induction on $d(x)$.
If $d(x) = 0$, then x is in the main direction, there is nothing to do.
Otherwise there are three cases :
- $p(x) = x \vee y$: then we are done,
- $p(x) < x \vee y$: then $x \vee y = p(x) \vee y$ and the result follows by induction since $d(p(x)) < d(x)$,
- $x < x \vee y < p(x)$: in this case $W_-(x) = W_-(x \vee y)$ hence x is in the main direction, this cannot happen.

(2) If $x \in W_-(x \vee y)$ then the result holds with $n = 0$. Otherwise there is z in $W_-(x \vee y)$ incomparable with x. By 1) there is $n > 0$, such that $p^n(x) = x \vee z$. We have $x \vee z \in W_-(x \vee y)$, which completes the proof.

□

We can now define an embedding of T into $UT(\mathbb{Q}, \mathbb{Q}_-, \mathbb{Q}_+)$ by using an induction on the depth of nodes.

More precisely, we define a mapping h that associates with x of depth $d(x) = k$ a sequence of the form $s_1 d_1 s_2 d_2 ... d_k s_{k+1}$ where each s_i is in $\mathbb{Q}$ and each d_i in $\mathbb{Q}_- \cup \mathbb{Q}_+$.

First we define an order-preserving embedding $h^{U(t_0)} : U(t_0) \longrightarrow \mathbb{Q}$, and for x of depth 0, we let $h(x) = h^{U(t_0)}(x)$.

For x of depth $n+1$ we need some constructions :

(1) we define an embedding of $W \longrightarrow \mathbb{Q}$ denoted by h^W where $W = W_-(x)$ (the same embedding will be used for $W_-(x)$ and $W_-(y)$ if $W_-(x) = W_-(y)$),

(2) for $z = p(x)$, we write $Dir(z,T)$ as $Dir_-(z,T) + \{Main(z)\} + Dir_+(z,T)$ where $+$ is the concatenation of sets linearly ordered by $\lhd_z$; (in other words, $Dir_-(z,T)$ is the set of directions in $Dir(z,T)$ that are smaller than $Main(z)$ with respect to $\lhd_z$, and $Dir_+(z,T)$ is the set of directions in $Dir(z,T)$ that are larger than $Main(z)$),

(3) we define two embeddings $k^{Dir_-(z,T)} : Dir_-(z,T) \longrightarrow \mathbb{Q}_-$, and $k^{Dir_+(z,T)} : Dir_+(z,T) \longrightarrow \mathbb{Q}_+$. We can now define, letting D be the direction of x relative to z :
   - $h(x) = h(z)k^{Dir_-(z,T)}(D)h^W(x)$ if $D \lhd_z Main(z)$, and
   - $h(x) = h(z)k^{Dir_+(z,T)}(D)h^W(x)$ if $Main(z) \lhd_z D$.

   Note that since we use induction on depth, $h(z)$ is well-defined, and so is $h(x)$.

Claim 7.11. *h is a join-embedding of T into the ordered tree $\mathbf{U} = UT(\mathbb{Q}, \mathbb{Q}_-, \mathbb{Q}_+)$.*

Proof. We only check that h preserves joins.

Let x and y in T be incomparable, let $D = dir_{x \vee y}(x)$ and $D' = dir_{x \vee y}(y)$.

- *First case* : none of D and D' is the main direction relative to $x \vee y$. Using the last assertion of Claim 7.9 and Claim 7.10, we have : $d(x) = k > n = d(x \vee y)$, $d(y) = m > n = d(x \vee y)$.

  We may assume that $h(x \vee y) = s_1 d_1 s_2 d_2 ... d_n s_{n+1}$. Hence $h(x) = s_1 d_1 s_2 d_2 ... d_n s_{n+1} d_{n+1} ... d_k s_{k+1}$and $h(y) = s_1 d_1 s_2 d_2 ... d_n s_{n+1} d'_{n+1} ... d'_m s'_{m+1}$, where $d_{n+1}$and d'_{n+1} are the codings of the directions D and D'. From Lemma 7.2, we have :

  $$s_1 d_1 s_2 d_2 ... d_n s_{n+1} = h(x) \vee h(y)$$

  as was to be proved.

- *Second case* : One of d and d', say d', is the main direction relative to $x \vee y$. We have in this case $d(x) = k > n = d(x \vee y)$, $d(y) = m \geq n = d(x \vee y)$.

  Again $h(x \vee y) = s_1 d_1 s_2 d_2 ... d_n s_{n+1}$. Then

  $$h(x) = s_1 d_1 s_2 d_2 ... d_n s_{n+1} d_{n+1} ... d_k s_{k+1} \text{ and}$$
  $$h(y) = s_1 d_1 s_2 d_2 ... d_n s'_{n+1} d'_{n+1} ... d'_m s'_{m+1}$$

  with $s'_{n+1} < s_{n+1}$ By Lemma 7.2 again we have $s_1 d_1 s_2 d_2 ... d_n s_{n+1} = h(x) \vee h(y)$, because $s'_{n+1} < s_{n+1}$.

□

This completes the proof of Theorem 7.1. □

Remarks 7.1. (1) $UT(\mathbf{1}, \varnothing, \mathbf{N})$ is a tree into which each infinite tree in the sense of [8] join-embeds.

(2) Fraïssé defines in [29] (Theorem 6.2 of Chapter 10) a (countable) tree $\mathbf{W}$, which is actually the unordered tree underlying $UT(\mathbb{Q}, \varnothing, \mathbf{1})$ and proves that all finite or countable trees embed into it. His theorem concerns embeddings of trees ; here we are concerned with join-embeddings of ordered trees. The tree $\mathbf{W}$ is a binary join-tree and all binary join-trees join-embed into it. Only binary join-trees do so, since $\mathbf{W}$ is binary and joins are preserved. Corominas ([7]) defines a countable join-tree into which every countable join-tree has a join-embedding.

7.3. **Representing join-trees and modular decompositions by standard binary trees.** Along the lines of Definition 7.1, we will represent the ordered tree $\mathbf{U} = UT(\mathbb{Q}, \mathbb{Q}_-, \mathbb{Q}_+)$ as well as any ordered tree T by a node-labeled standard binary tree. As a preliminary step, we represent an ordered tree T by a *rooted, diagram-connected ordered tree*. We will then apply Proposition 7.2 to this tree and obtain immediately the result.

Definition 7.6. *The tree $\Delta(T, \preccurlyeq)$ associated with an ordered tree T, ω-ordered by $\preccurlyeq$.* Let T be an ordered tree, which is ω-ordered, and enumerated as $t_0, t_1, ...$ We will use some notions defined in the proof of Theorem 7.1, and in particular the chains $U(x)$, $W_-(x)$, $W_+(x)$, the node $p(x)$ and the depth $d(x)$, associated with every node $x \in T$ (the node $p(x)$ and the chains $W_-(x)$, $W_+(x)$ are defined only for x of positive depth).

A rooted diagram-connected ordered tree Δ can be specified from T by the following conditions :

(1) Its nodes are of types ν, δ_- or δ_+, the sons of the nodes of type ν are of type δ_- or δ_+ and those of the nodes of type δ_- or δ_+ are of type ν.
(2) The nodes of type ν are the nodes of T ; a node x of depth n in T is at distance $2n + 1$ to the root in Δ.
(3) The root of Δ is of type δ_+, and the linearly ordered set of its sons is $U(t_0)$, the set of nodes of T of depth 0.
(4) The sons of a node x of type ν are the directions (in T) relative to this node, except for the main direction relative to x, *i.e.* the one that meets $W_-(x)$; the nodes of type δ_- are those which are strictly smaller than the main direction relative to x, those of type δ_+ are the directions which are strictly larger than the main direction relative to x.
(5) A node w of type δ_- or δ_+ is a direction of the form $dir_{T,p(x)}(x)$ for some x where $p(x)$ is the father of w in Δ and $W_-(x) \subseteq w$. Then $W_-(x)$ is the set of sons of w.

Lemma 7.5. *There is a unique rooted, diagram-connected ordered tree $\Delta(T, \preccurlyeq)$ characterized by the above conditions. It can be defined from $(T, \preccurlyeq)$ by a domain extending MS transduction. This transduction can also define an ω-order $\preccurlyeq'$ on $\Delta(T, \preccurlyeq)$. The tree T can be defined from $\Delta(T, \preccurlyeq)$ by an FO transduction denoted by ∇.*

Proof. By using induction on n, one can prove that for every n, the set of nodes of $\Delta(T, \preccurlyeq)$ at distance at most n of the root is defined in a unique way.

It is clear that $\Delta(T, \preccurlyeq)$ is a rooted, diagram-connected and ordered tree.

It can be defined from $(T, \preccurlyeq)$ by an MS transduction because $U(x)$, $U_-(x)$, $U_+(x)$, $p(x)$ are definable by MS formulas : this follows immediately from the way they are defined in the proof of Theorem 7.1.

In order to define $\preccurlyeq'$ we use Lemma 9.1 of the appendix showing that every MS transduction transforming a structure T into a structure U, can be enlarged so as to transform any ω-order $\preccurlyeq$ on T into an ω-order $\preccurlyeq'$ on U. We let $\Delta'(T, \preccurlyeq)=(\Delta(T, \preccurlyeq), \preccurlyeq')$.

Let us explain how T can be defined from $\Delta(T, \preccurlyeq)$:

(1) its nodes are those of $\Delta(T, \preccurlyeq)$ of type ν,

(2) for two distinct nodes x, y of type ν in $\Delta(T, \preccurlyeq)$, we let :

$x <_T y$ if and only if either $x <_{\Delta(T,\preccurlyeq)} y$ or $x \leq_{\Delta(T,\preccurlyeq)} u$, u is a *left brother* of y ; (this latter condition is equivalent to $dir_z(x) \lhd_{\Delta(T,\preccurlyeq),z} dir_z(y)$ where z is the father of y).

Directions in T are ordered as follows :

(3) if x, y are distinct nodes of T and $z = x \vee y$, then $dir_z(x) \lhd_{T,z} dir_z(y)$ iff
- either $dir_z(x) \lhd_{\Delta(T,\preccurlyeq),z} dir_z(y)$,
- or $x <_{\Delta(T,\preccurlyeq)} w$ where w is a son of z labeled by δ_- and $y \in U(z)$ (*i.e.*, $dir_z(y)$ is the main direction relative to z),
- or $w <_{\Delta(T,\preccurlyeq)} y$ where w is a son of y labeled by δ_+ and $x \in U(z)$ (*i.e.*, $dir_z(x)$ is the main direction relative to z)

This definition is clearly expressible as an FO transduction, that we will denote by ∇. □

By combining Proposition 7.2 and Lemma 7.5 we obtain :

Corollary 7.1. *There exists a domain extending MS transduction $\Delta \circ \sigma$ that associates with every ordered tree T that is also ω-ordered a node-labeled standard binary ω-ordered tree W. An MS transduction can define T from W.*

Proof. In one direction one uses $\Delta \circ \sigma$, and in the other the mapping $Compress \circ \nabla$ which is an MS transduction, as composition of two MS transductions. □

We now apply this to the representation of modular decompositions. We recall that $Gdec(\mathbf{G})$ is a modular tree augmented with some node-labels and a binary relation $fedg$ between some nodes. Formally, $Gdec(\mathbf{G})$ is of the form $(T, \leq, lab_\oplus, lab_\otimes, lab_{\overrightarrow{\otimes}}, fedg)$.

Assuming that $(T, \leq)$ is represented by a node-labeled standard binary tree $(W, \leq node_T, lson_W, rson_W)$ in the sense of Corollary 7.1, then we define a *sparse representation of the modular decomposition of* $\mathbf{G}$ as a structure $Sdec(\mathbf{G}) = (W, \leq node_T, lson_W, rson_W, lab_\oplus, lab_\otimes, lab_{\overrightarrow{\otimes}}, sedg)$, where $sedg$ is the restriction of $fedg$ to the set of sons of nodes of type IV (corresponding to prime factors). The relation $fedg$ on sons of nodes of type III (occurrences of linear sums) is no longer necessary because the linear order on directions in T is handled by the inorder on W derived from the left and right types of sons.

Theorem 7.2. *There exists a domain extending MS transduction that associates with an ω-ordered graph $\mathbf{G}$ a sparse representation $Sdec(\mathbf{G})$ of its modular decomposition. The structure $Sdec(\mathbf{G})$ is a vertex- and edge-labeled graph of degree $m+3$ where m is the maximum degree of a vertex in a prime factor of $\mathbf{G}$ (*cf. *case IV of Proposition 5.1). There exists an MS-transduction that defines $\mathbf{G}$ from $Sdec(\mathbf{G})$.*

Proof. It suffices to combine the MS transductions of Theorem 6.1 and Corollary 7.1.

Note that the tree $(T, \leq)$ underlying $Gdec(\mathbf{G})$ is not ordered : only the sons of the nodes of type III are linearly ordered, whereas Corollary 7.1 uses ordered trees. But since an ω-order is available in $\mathbf{G}$ whence in T, we can use it to make T into an ordered tree, just by defining a linear order on the directions relative to the nodes of types I, II and IV.

The bound on the degree of $Sdec(\mathbf{G})$ follows from the definitions. □

8. Concluding remarks and questions

We have proved that the graph $Sdec(\mathbf{G})$ representing the modular decomposition of a countable graph $\mathbf{G}$ can be defined from $\mathbf{G}$ and any ω-order of its vertices by an MS transduction, and that, conversely, $\mathbf{G}$ is definable from $Sdec(\mathbf{G})$ also by an MS transduction.

Finite presentations of countable graphs of several types are studied by Blumensath and Graedel in [3]. One can thus ask whether a finite presentation of $\mathbf{G}$ yields one of same type of $Sdec(\mathbf{G})$. A graph $\mathbf{G}$ is *VR-equational* (*i.e.* is the canonical solution of a finite system of equations over so-called *VR operations*) if and only if it is the image of the standard binary tree $\mathbf{B} = (\{0,1\}^*, lson_{\mathbf{B}}, rson_{\mathbf{B}})$ under an MS transduction ([3]). If $\mathbf{G}$ is VR-equational, and if an ω-order of $V_{\mathbf{G}}$ is MS definable, then by Theorem 7.2 and Proposition 9.2-1, $Sdec(\mathbf{G})$ is also the image of $\mathbf{B}$ under an MS transduction, hence is VR-equational. (Since no ω-order on $\mathbf{B}$ is MS definable, the second assumption cannot be deleted). Conversely, if $Sdec(\mathbf{G})$ is VR-equational, so is $\mathbf{G}$.

Question 8.1. Is the former assertion true without the hypothesis that an ω-order of $V_{\mathbf{G}}$ is MS definable ?

It is possible that something weaker than an ω-order (e.g., a partial order of some kind) is sufficient for Proposition 6.1 and Theorem 7.2 to hold.

Remark 8.1. Certain of our constructions consist actually in constructing *automatic structures* (we refer the readers to [38] or [3] for definitions). The class of automatic structures contains the VR-equational graphs, characterized also as *prefix-recognizable graphs*. These structures have domains defined as regular languages and relations defined by multihead synchronized automata. The tree $\mathbf{B}$ ordered by inorder is an automatic structure. So is the universal tree $UT(\mathbb{Q}, \mathbb{Q}_-, \mathbb{Q}_+)$ with domain defined as $(L_{\mathbb{Q}}.L_{\mathbb{Q}^*})^*L_{\mathbb{Q}}$ where $L_{\mathbb{Q}} = (0 \cup 11)^*10$ represents $\mathbf{B}$[14] and $L_{\mathbb{Q}^*} = (0 \cup 1)(0 \cup 11)^*10$ represents the linear order $\mathbb{Q}_- + \mathbb{Q}_+$.

Let us detail the first construction. Consider a standard binary tree T. Every path from the root to a node can be represented by a word in $\{0,1\}^*$ where 0 represents a transition from a node to its left son and 1 represents one from a node to its right son. Hence the complete binary tree that encodes the set $\mathbb{Q}$ is isomorphic to $\mathbf{B} = (\{0,1\}^*, lson_{\mathbf{B}}, rson_{\mathbf{B}})$ where $lson_{\mathbf{B}}(u,v)$ holds if and only if $v = u0$, and $rson_{\mathbf{B}}(u,v)$ holds if and only if $v = u1$. In terms of the words coding nodes, the inorder on $\mathbf{B}$ is defined by :

$x \leq_{in,\mathbf{B}} y$ if and only if

$x = y$ or $x = t0x'$ and $y = t1y'$, or $y = x1y'$, or $x = y0x'$ for some t, x', y'.

It follows that $\mathbf{B} = (\{0,1\}^*, lson_{\mathbf{B}}, rson_{\mathbf{B}}, \leq_{in,\mathbf{B}})$ is an automatic structure.

If in the structure $Sdec(\mathbf{G})$ we replace $lson_W$ and $rson_W$ by $ldes_W$ and $rdes_W$ such that $ldes_W(x,y)$ holds iff $x \leq_T u$ where $lson_W(u,y)$ holds, and similarly for $rdes_W$, then we obtain a binary structure $Fdec(\mathbf{G})$ (that is no longer sparse) from which $\mathbf{G}$ can be constructed by an FO transduction. It follows that $\mathbf{G}$ is automatic if $Fdec(\mathbf{G})$ is, because the image of an automatic structure under an FO transduction is automatic ([3] Proposition 4.3).

[14] We have two representations of $\mathbf{B}$ as an automatic structure. In the first one, the set of words is $\{0,1\}^*$. In the second one, it is a prefix-free language. The second representation "implements" the construction of Lemma 7.1.

Question 8.2. For which graphs $\mathbf{G}$ is it true that the binary structure $Fdec(\mathbf{G})$ is automatic ?

Acknowledgement : We thank Teodor Knapik who invited B. Courcelle in La Réunion in 2004 on his research grant. We thank the referees for their helpful comments.

References

[1] K. Barthelmann, When can an equational simple graph be generated by hyperedge replacement ? *Mathematical Foundations of Computer Science*, Lec. Notes Comput. Sci. 1450 (1998) 543-552.

[2] M. Benedikt, L. Segoufin, Towards a characterization of order-invariant queries over tame structures. CSL 2005, Lec. Notes Comp. Sci. 3634 pp. 276-291, 2005.

[3] A. Blumensath, E. Grädel, Finite presentations of infinite structures : Automata and interpretations, *Theory of Computing Systems* 37 (2004) 641-674.

[4] A. Carayol and S. Wöhrle, *The Caucal Hierarchy of Infinite Graphs in Terms of Logic and Higher-Order Pushdown Automata*, Proceedings of "Foundations of Software Technology and Theoretical Computer Science", Lecture Notes in Computer Science 2914, 2003, pp. 112-123, Springer.

[5] D. Caucal, On the regular structure of prefix rewriting. *Theoretical Computer Science* 106 (1992) 61 - 86.

[6] T. Colcombet, *Properties and representation of infinite structures*, Ph.D., Université Rennes I, 2004. (Available from : http://www.irisa.fr/galion/colcombe/index_fr.html)

[7] E. Corominas, On better-quasi-ordering countable trees, *Discrete Mathematics*, 53 (1985) 35-53.

[8] B. Courcelle, Fundamental properties of infinite trees, *Theoretical Computer Science* 25 (1983) 95-169.

[9] B. Courcelle, Recursive applicative program schemes, in *Handbook of Theoretical Computer Science vol. B*, J. Van Leeuwen ed., Elsevier, 1990, pp.459-492.

[10] B. Courcelle, *The Monadic Second-Order Logic of Graphs IV : Definability Properties of Equational Graphs*, Ann. Pure Appl. Logic 49(3) : 193-255 (1990)

[11] B. Courcelle, Monadic second-order graph transductions: A survey. *Theoretical Computer Science*, 126 (1994)53–75.

[12] B. Courcelle, *The monadic second-order logic of graphs IX : Machines and their behaviours*, Theoretical Computer Science, 151 (1995) 125-162

[13] B. Courcelle, The monadic second-order logic of graphs X : Linear orderings, *Theoretical Computer Science* 160 (1996) 87-143.

[14] 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, Vol. 1: Foundations*, World Scientific, 1997, pp. 313–400.

[15] B. Courcelle, Clique-width of countable graphs: a compactness property. *Discrete Mathematics* 276 (2004) 127-148.

[16] B. Courcelle, The monadic second-order logic of graphs XIV : Uniformly sparse graphs and edge set quantifications. *Theoretical Computer Science* 299 (2003) 1-36.

[17] B. Courcelle, The monadic second-order logic of graphs XV : On a Conjecture by D. Seese, *Journal of Applied Logic*, 4 (2006) 79-114.

[18] B. Courcelle, The monadic second-order logic of graphs XVI : Canonical graph decompositions, *Logical Methods in Computer Science*, 2 (2006) 1-46.

[19] B. Courcelle, C. Delhommé, The modular Decomposition of Countable Graphs : Constructions in Monadic Second-Order Logic, CSL 2005, LNCS 3634, pp. 325-338, 2005.

[20] B. Courcelle, T. Knapik, *The evaluation of first-order substitution is monadic second-order compatible*, Theoretical Computer Science, 281 (2002) 177-206.

[21] B. Courcelle, J.A. Makowsky and U. Rotics, *On the Fixed Parameter Complexity of Graph Enumeration Problems Definable in Monadic Second Order Logic*, Discrete Applied Mathematics 108 (2001) 23-52.

[22] W. Cunningham, *Decomposition of directed graphs*, SIAM J. Alg. Discrete Methods 3 (1982) 214-228.

[23] C. Delhommé, Clique-width of countable graphs and prime subgraphs. Manuscript, 2005.

[24] R. Diestel, *Graph Decompositions: a study in infinite graph theory*, Oxford University Press, 1990

[25] R. Downey, M. Fellows, *Parametrized complexity*, Springer, 1999.

[26] A. Ehrenfeucht, G. Rozenberg, Primitivity is hereditary for 2-structures, Theoret. Comput. Sci. 70 (1990) 343-358.

[27] A. Ehrenfeucht, T. Harju, G. Rozenberg, Decomposition of infinite labeled 2-structures, Lec. Notes Comput. Sci. 812 (1994) 145-158.
[28] A. Ehrenfeucht, T. Harju, G. Rozenberg, The theory of 2-structures. A framework for decomposition and transformation of graphs, World Scientific Publishing Co., River Edge, New-Jersey, 1999.
[29] R. Fraïssé, Theory of relations, *Studies in logic* vol. 118, North-Holland, 1986 (Second edition, Elsevier, 2000).
[30] T. Gallai, Transitiv orientbare Graphen, *Acta Math. Acad. Sci. Hungar.* 18 (1967) 25-66. English translation by F. Maffray and M. Preissmann in J. J. Ramirez-Alfonsin and B. Reed eds., Perfect graphs, Wiley 2001, pp. 25-66.
[31] F. Gire, M. Nivat : Langages algébriques de mots biinfinis, *Theoret. Comput. Sci.* 86 (1991) 277-323
[32] M. Habib, Substitution des structures combinatoires, théorie et algorithmes, Thèse d'Etat, Université Paris VI, 1981.
[33] B. R. Hodgson, Théories décidables par automate fini, Thèse de doctorat, Université de Montréal, 1976, 171 pages.
[34] B. R. Hodgson, Décidabilité par automate fini, *Ann. sc. math. Québec*, 1983, vol. 7, n. 1, 39-57.
[35] P. Ille, Graphes Indécomposables Infinis, *C. R. Acad. Sci. Paris*, Série I-318 (1994) 499-503.
[36] P. Ille, Indecomposable Graphs, *Disc. Math.*, 176 (1997) 71-78.
[37] D. Kelly, Comparability graphs, in *Graphs and order*, I. Rival ed., D. Reidel Pub. Co., 1985, pp. 3-40.
[38] B. Khoussainov, A. Nerode, Automatic presentations of structures, in *Logic and Computational Complexity*, Lec. Notes Comput. Sci. 960 (1995) 367-392.
[39] R. Lyndon and P.Schupp, *Chap.III Geometric methods* of the book *Combinatorial Group Theory*, Springer-Verlag, 1977.
[40] F. de Montgolfier, Décomposition modulaire de graphes, théorie, extensions et algorithmes, Thèse de doctorat, Université Montpellier II, 2003.
[41] R. Möhring, R. Radermacher, Substitution decomposition of discrete structures and connections with combinatorial optimization, *Annals Discrete Maths* 19 (1984) 257-356.
[42] D. Muller, Paul E. Schupp, *The Theory of Ends, Pushdown Automata, and Second-Order Logic*, Theor. Comput. Sci. 37: 51-75 (1985)
[43] D. Renault, *Étude des graphes planaires cofinis selon leurs groupes de symétries*, Ph D., Bordeaux 1 University, 2004, (Available from : http://www.labri.fr/perso/renault)
[44] J. H. Schmerl, W. Trotter, Critically indecomposable partially ordered sets, graphs, tournaments and other binary relational structures, *Disc. Math.*, 113 (1993) 191-205.

9. Appendix : MS logic and MS transductions

We review background definitions and results on monadic second-order logic, on transformations of structures expressed in this language and its extensions. The reader is referred to the book chapter [14], or to the preliminary sections of any of the articles [11, 13, 16, 17]. However all necessary definitions are given in full in the present section.

9.1. **Relational structures and monadic second-order logic.** A *relational signature* is a finite set $R = \{A, B, C, ...\}$ of *relation symbols*, each of them given with a non-negative integer $\rho(A)$ called its *arity*. We denote by $\mathcal{STR}(R)$ the set of 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. Unless otherwise specified, structures have countable *domains* D_S.

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 $\mathbf{G}$ is defined as an $\{edg\}$-structure $\mathbf{G} = \langle V_{\mathbf{G}}, edg_{\mathbf{G}}\rangle$ where $V_{\mathbf{G}}$ is the set of vertices of $\mathbf{G}$ and $edg_{\mathbf{G}} \subseteq V_{\mathbf{G}} \times V_{\mathbf{G}}$ is a binary relation representing the edges. Edges are directed. An undirected edge is a pair of opposite directed edges. An ordered graph is a binary structure with two relations, the edge relation and the order relation. If in addition we need vertex labels, we represent them by unary relations. Binary structures can be seen as vertex- and edge labeled graphs. If we have several binary relations, say A, B, C, the corresponding graphs will have edges of types A, B, C.

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 denote set variables, lowercase letters denote ordinary 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 *individual as well as of set variables*. Hence, we allow first-order formulas with free set variables and written with the atomic formulas $x \in X$. In first-order formulas, only individual 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 :

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

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

A formula $\varphi \in MS(R \cap \{\preccurlyeq\}, W)$ is *order-invariant* if it describes a property of structures S in $\mathcal{STR}(R)$ with help of an auxiliary linear ordering of D_S of type ω if D_S is infinite. More precisely, we require that, for $S \in \mathcal{STR}(R)$, for every two linear orderings $\preccurlyeq_1$ and $\preccurlyeq_2$ of D_S, both of order type ω when D_S is infinite, for every assignment γ of elements of D_S to the individual variables of W, and of subsets of D_S to the set variables of W, we have :

$$(S, \preccurlyeq_1, \gamma) \models \varphi \text{ if and only if } (S, \preccurlyeq_2, \gamma) \models \varphi$$

If this is the case, φ defines the property of (S,γ) : $P(S,\gamma)$ if and only if $(S,\preccurlyeq,\gamma) \models \varphi$ for some linear ordering $\preccurlyeq$ of D_S of type at most ω, and we say that P is MS- (or FO-) *order-invariant*. Order-invariant properties are investigated in [13] and [2].

If $S = \langle D_S, (A_S)_{A\in R}\rangle$ and $\preccurlyeq$ is a linear ordering on D_S, we denote by $(S,\preccurlyeq)$ the relational structure $\langle D_S, \preccurlyeq, (A_S)_{A\in R}\rangle$.

9.2. **First-Order and Monadic Second-order transductions.** We also use FO and MS formulas to define certain graph transformations. As in language theory, a binary relation $\tau \subseteq A \times B$ is called a *transduction* : $A \to B$. It is considered as a multivalued partial mapping associating with certain elements of A (usually graphs) one or more elements of B (also graphs).

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 $D_S \times \{1, ..., k\}$. 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 [11, 14].

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

$\Delta = (\varphi, \psi_1, \cdots, \psi_k, (\theta_w)_{w\in Q*k})$
where $k > 0, Q*k := \{(q,\vec{j}) : q \in Q, \vec{j} \in [k]^{\rho(q)}\}$,
$\varphi \in MS(R,W), \psi_i \in MS(R, W \cup \{x_1\})$ for $i = 1, \cdots, k$,
and $\theta_w \in MS(R, W \cup \{x_1, \cdots, x_{\rho(q)}\})$, 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) $D_T = \{(d,i) : d \in D_S, i \in [k], (S,\gamma,d) \models \psi_i\}$
(iii) for each q in Q : $q_T = \{((d_1,i_1),\cdots,(d_t,i_t)) \in D_T^t : (S,\gamma,d_1,\cdots,d_t) \models \theta_{(q,\vec{j})}\}$, where $\vec{j} = (i_1,\cdots,i_t)$ and $t = \rho(q)$.

(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 relation

$def_\Delta := \{(S,T) : T = def_\Delta(S,\gamma)$ for some W-assignment γ in $S\} \subseteq \mathcal{STR}(R) \times \mathcal{STR}(Q)$.

A transduction $f \subseteq \mathcal{STR}(R) \times \mathcal{STR}(Q)$ is an *MS (definable) transduction* if it is equal to def_Δ for some (Q,R)-definition scheme Δ (equal up to isomorphisms of

structures). In the case where $W = \emptyset$, we say that f is *definable without parameters* (note that it is functional).

We will refer to the integer k by saying that def_Δ is *k-copying*.

We will say that the MS transduction is *domain extending*, if $k > 1$, and ψ_1 is the formula $True$. In this case, 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 def_Δ is an *FO transduction*.

A transduction $def_\Delta \subseteq \mathcal{STR}(R \cup \{\preccurlyeq\}) \times \mathcal{STR}(T)$ is *order-invariant* if for every $S \in \mathcal{STR}(R)$, every assignment γ of values in $\mathcal{P}(D_S)$ to the parameters, for every two linear orderings $\preccurlyeq_1$ and $\preccurlyeq_2$ on D_S of order type at most ω, the structures $def_\Delta(S, \preccurlyeq_1, \gamma)$ and $def_\Delta(S, \preccurlyeq_2, \gamma)$ are isomorphic. In this case Δ defines a transduction : $\mathcal{STR}(R) \to \mathcal{STR}(T)$ with help of auxiliary linear orderings.

Lemma 9.1. *Let an MS (or FO) transduction $\tau : \mathcal{STR}(R) \longrightarrow \mathcal{STR}(Q)$. Let us add to R and to Q a binary relation symbol $\preccurlyeq$ intended to represent orders on the domains of structures. One can transform τ into an MS (or FO) transduction $\tau' : \mathcal{STR}(R \cup \{\preccurlyeq\}) \longrightarrow \mathcal{STR}(Q \cup \{\preccurlyeq\})$ such that for every S in $\mathcal{STR}(R)$, for every ω-order $\preccurlyeq$ on its domain, $\tau'(S, \preccurlyeq) = (\tau(S), \preccurlyeq')$ where $\preccurlyeq'$ is an ω-order on the domain of $\tau(S)$.*

Proof. Let τ be k-copying. It is easy to define FO formulas $\theta_w \in MS(R \cup \{\preccurlyeq\}, W \cup \{x_1, x_2\})$, for $w = (\preccurlyeq, \vec{j}) \in \{\preccurlyeq\} * k$ such that, in $\tau'(S, \preccurlyeq)$:

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

It is clear that $\preccurlyeq'$ is an ω-order on the domain of $\tau(S)$ if $\preccurlyeq$ is one on S. □

9.3. **The fundamental property of definable 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 9.1. *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 $\gamma : W \longrightarrow \mathcal{P}(D_S)$ for every assignment $\eta : V' \longrightarrow \mathcal{P}(D_S)$ we have :*

$(S, \eta \cup \gamma) \models \beta^\#$ if and only if :
$def_\Delta(S, \gamma)$ is defined, η^k is a V–assignment in $def_\Delta(S, \gamma)$,
and $(def_\Delta(S, \gamma), \eta^k) \models \beta$.

If the definition scheme and β are FO, then 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 $D_S \times [k]$. We call $\beta^\#$ the *backwards translation* of β relative to the transduction def_Δ. Here are some important consequences of this proposition and of Lemma 9.1.

Proposition 9.2. (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.*

(3) *The inverse image of a class of structures defined by an order-invariant MS-formula under an order-invariant MS-transduction is definable by an order-invariant MS-formula.*

LaBRI, Université Bordeaux 1 et CNRS, 31, cours de la Libération, F-33405 Talence, France. and ERMIT, Université de La Réunion, 15, ave René Cassin, BP 7151, F-97715, Saint-Denis, Messag. cedex 9, La Réunion, France.

E-mail address: courcell@labri.fr and delhomme@univ-reunion.fr