The Modular Decomposition of Countable Graphs: Constructions in Monadic Second-Order Logic

Bruno Courcelle[1] and Christian Delhommé[2]

[1] Université Bordeaux 1, LaBRI (CNRS)
courcell@labri.fr
[2] Université de La Réunion, ERMIT
delhomme@univ-reunion.fr

Abstract. We show that the modular decomposition of a countable graph can be defined from this graph, given with an enumeration of its set of vertices, by formulas of Monadic Second-Order logic. A second main result is the definition of a representation of modular decompositions by a low degree relational structures, also constructible by Monadic Second-Order formulas.

1 Introduction

The present article investigates the *modular decomposition of countable graphs* and more precisely, its construction by *Monadic Second-Order* (*MS* in short) formulas. The notion of modular decomposition of a finite graph has been studied extensively in many articles, and under various names. Möhring and Radermacher give in [17] a survey of this frequently rediscovered notion. It is important, not only for algorithmic purposes, but also for establishing structural properties, in particular of partial orders and their *comparability graphs* (see Kelly [15] who discusses finite and infinite comparability graphs and their *modules*); for instance, one can determine the transitive orientations of a comparability graph from its modular decomposition.

The modular decomposition of a finite graph is the finite tree of its *strong modules*, with inclusion as ancestor relation, together with some structure attached to the nodes of the tree. Each node is a *graph operation*, either the *disjoint union*, the *complete product*, the *sequential product* or the *substitution* to the vertices of a *prime* graph, i.e., a graph that is not expressible in terms of these operations. The strong modules of an infinite graph can be defined in the very same way as for finite graphs. They are either pairwise disjoint or comparable for inclusion, but they do not form a tree, defined as a connected and directed graph without circuits. They form a *generalized tree* defined as a partial order such that the set of elements larger than any element is linearly ordered (and called below simply a tree). Such trees may have no root. The linearly ordered set $\mathbf{Q}$ of rational numbers is a tree in this sense. For defining the modular decomposition of a countable graph, we *do not take all strong modules*, but only some of them called *robust*. Doing so we obtain a countable tree associated with a countable graph. The basic definitions are reviewed in Section 2.

L. Ong (Ed.): CSL 2005, LNCS 3634, pp. 325–338, 2005.

Our goals here are to represent modular decompositions of countable graphs by relational structures, and to use *MS logic* to construct from a graph its modular decomposition. 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. The linearly ordered set $\mathbf{Q}$ has an empty Hasse diagram. One may think that one *must* represent it by a *complete infinite graph*. However, it can be defined as a certain ordering of the nodes of the complete infinite binary tree (here in the usual sense). A similar binary tree can be constructed from any linearly ordered set A by first-order formulas using an auxiliary enumeration of A (i.e., an ordering of A isomorphic to the ordinal ω). By using these facts, we can represent the modular decomposition of a countable graph G by a countable graph of maximum degree $m + 3$ where m is the least upper bound of the degrees of the prime induced subgraphs of G. It may happen that m is finite, even if G has vertices of infinite degree. This is the case for countable cographs, defined as the graphs without induced P_4 (i.e., without induced path of length 3).

Because of space limitations, proofs are sketched or omitted. Complete proofs can be found in [11].

2 Robust Modules and Modular Decomposition

Unless otherwise specified, trees, forests, graphs and relational structures are countably infinite. A linear order isomorphic to ω is called an *ω-order*. An ω-ordering of a set is equivalent to an enumeration $x_0, x_1, ..., x_n, ...$ of this set.

Definitions 1. (*Trees, $\vee$-trees and leafy trees.*) A *forest* is a partial order $(T, \leq)$ such that for every element x, called a *node*, the set $T^x = \{y \in T \mid x \leq y\}$ is linearly ordered. A *tree* is a forest that is *directed*, i.e., such that every two nodes have an upper bound. A forest is the disjoint union of the trees which are the connected components of its comparability graph.

A tree is a *$\vee$-tree* (read a "sup-tree") if any two nodes x and y have a least upper bound denoted by $x \vee y$. A *sub-$\vee$-tree* must preserve the function $\vee$.

A *leaf* 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 it is a $\vee$-tree and every internal node is the least upper bound of two leaves. A finite tree is a finite rooted tree in the usual sense, and its root is the unique maximal element.

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 the (unique) minimal node among those $> x$. We say in this case that x is a *son* of y.

For a partial order $(P, \leq)$ we use the notations $P^x = \{y \in P \mid x \leq y\}$, $P^{>x} = \{y \in P \mid y > x\}$, $P_x = \{y \in P \mid y \leq x\}$ and $P_{<x} = \{y \in P \mid y < x\}$. We let $HD(P)$ denote its *Hasse diagram*, i.e. the directed graph with set of vertices P and edges $x \to y$ whenever $x < y$ 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.

Thus a tree is diagram-connected iff every node which is not the root has a father, and the graph of the father relation is connected. Any two nodes are thus at finite distance in the graph $HD(P)$. A diagram-connected tree may have no root.

Definitions 2. (*Directions in $\vee$-trees.*) Let T be a $\vee$-tree. For every x, $T_{<x}$ ordered by the induced ordering is a forest, hence a union of trees. Each of these trees D is called a *direction relative to* x. If $y \in D$, 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. 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.

A $\vee$-tree $(T, \leq)$ is *ordered* if it is equipped with a linear order $\trianglelefteq_x$ on each set $Dir(x,T)$.

Definitions 3. (*Graph substitutions.*) Graphs are simple, directed, loop-free. Undirected graphs are those where each edge has an opposite edge. We denote by $x \to y$ the existence of an edge from x to y. Although a forest is a graph or can be considered as a graph, we will use the special term "nodes" for the vertices of a tree or a forest. We denote by $V_G$the set of vertices of a graph G.

If G is a graph and $X \subseteq V_G$, we denote by $G[X]$ the *induced* subgraph of G consisting of X and all the edges, the two ends of which are in X. If E is a set of edges of G, we denote by $G[E]$ the subgraph of G consisting the edges of E and all their end vertices.

If G and H are graphs with disjoint sets of vertices, and u is a vertex of G, we denote by $G[H/u]$ the graph resulting of the *substitution of H for u in G*. Its set of vertices is $V_G \cup V_H - \{u\}$, its edges are those of H, those of G that are not incident with u, the edges $x \to y$ whenever $x \in V_G - \{u\}$, $x \to u$ in G, $y \in V_H$, and the edges $y \to x$ whenever $x \in V_G - \{u\}$, $u \to x$ in G, $y \in V_H$.

If G and H are not disjoint, we replace H by an isomorphic copy disjoint with G. If $u_1, ..., u_n$ are vertices of G and $H_1, ..., H_n$ are graphs, we define $G[H_1/u_1, ..., H_n/u_n]$ as $G[H_1/u_1]...[H_n/u_n]$. The order in which substitutions are done is irrelevant, hence we can consider they are done simultaneously.

If H_v is a graph associated with each $v \in V_G$, we denote by $G[H_v/v, v \in V_G]$ the graph resulting from *the simultaneous substitution in G* of H_v for $v \in V_G$. It can be defined as the graph with vertices (v,w) for $v \in V_G$ and $w \in V_{H_v}$ and edges $(v,w) \to (v',w')$ iff either $v = v'$ and $w \to w'$ (in H_v), or $v \to v'$ (in G).

We will also use the graph operations $\oplus, \otimes$ and $\overrightarrow{\otimes}$: $G \oplus H$ is the disjoint union of G and H, $G\overrightarrow{\otimes}H$ is $G \oplus H$ augmented with edges from each vertex of G to each vertex of H, and $G \otimes H$ is $G\overrightarrow{\otimes}H$ augmented with edges from each vertex of H to each vertex of G. The graphs $G \oplus H$, $G\overrightarrow{\otimes}H$ and $G \otimes H$ can be defined as $K[G/u, H/v]$ for graphs K with two vertices u and v, and, respectively, no edge, an edge from u to v, edges between u and v in both directions. They are associative. We will consider them as operations of variable arity in the usual way. The operations $\oplus, \otimes$ are also commutative.

More generally, every graph can be turned into a graph operation. With a finite graph G, with vertices $v_1, ..., v_n$ we associate an n-ary graph operation, denoted by σ_G (where σ stands for *substitution*) defined by $\sigma_G(H_1, ..., H_n) = G[H_1/v_1, ..., H_n/v_n]$. If G is infinite, then σ_G is defined similarly as an operation of countably infinite arity.

Definition 4. (*Modules.*) Let G be a graph. A *module* of G is a subset M of its set of vertices V_G such that for every vertices x, y in M and every vertex z not in M: $x \to z$ implies $y \to z$ and: $z \to x$ implies $z \to y$. In words this means that every vertex not in M "sees" all vertices of M in the same way. (This notion is studied in several works in particular [15], [17] and [13], in different formal frameworks and using different terminologies). If M is a module then $G = H[G[M]/v]$ for some H and v. Hence the notion of a module identifies a way of expressing a graph as the result of a substitution.

A module is *strong* if it is non empty and does not overlap any module. (Two sets *meet* if they have a nonempty intersection. They *overlap* if they meet and are incomparable for inclusion.) The singletons and the set V_G are strong modules. We will identify frequently a module M and the subgraph $G[M]$ it induces. A graph G is *prime* if it has no trivial module, where the *trivial modules* are $\varnothing$, the singletons and V_G.

The smallest prime undirected graph is the path P_4 with 3 edges and 4 vertices. The smallest prime directed graphs have 3 vertices. The graph H is a module of $G[H/u]$ and the graphs G and H are modules of $G \oplus H$, $G \overrightarrow{\otimes} H$ and $G \otimes H$.

Fact. *A countable graph may have uncountably many strong modules.*

Given two vertices x, y in a graph G, we let $M(x, y)$ be the intersection of all strong modules containing x and y. It is a strong module. We call $M(x, y)$ a *robust module*. It may be the set of all vertices or $\{x\} = M(x, x)$. A countable graph has countably many robust modules. The *maximal proper strong (mps in short) modules of a graph* G are the maximal proper strong submodules of its robust modules.

Proposition 1. *For every graph G, the robust modules form a leafy tree, denoted by $rdec(G)$ which is a sub-$\vee$-tree of the tree of strong modules. Every strong module (in particular G) is the union of the directed set of robust modules included in it. A strong module that is not a singleton is robust iff it is the father of some strong module.*

Proposition 2. *[12] Let G be a graph.*

1. *For every non singleton robust module M, we have one and only one of the following possibilities:*
   (I) either $G[M]$ is the disjoint union (denoted $\oplus$) of a family of connected graphs $C_i, i \in I$,
   (II) or $G[M]$ is the complete product (denoted $\otimes$) of a family of graphs $C_i, i \in I$,where no C_i is of type II,

(III) or $G[M]$ is the linear product (denoted $\overrightarrow{\otimes}$) of a linearly ordered family of graphs $C_i, i \in I$,where no C_i is of type III,
(IV) or $G[M] = P[C_i/u_i, i \in I]$ where P is a prime graph ; this prime graph is unique up to isomorphism.

2. *The graphs C_i are the mps submodules of M. They are not necessarly robust. Their common father in the tree of strong modules of G is M.*
3. *The graphs P of Case IV are induced subgraphs of G.*

The graphs P of case IV are called the *prime factors* of G. If G is given as $Q[P[H_v/v, v \in V_P]/u]$, with P prime, then P is one of its a prime factors. In order to decompose G it suffices to decompose separately Q and the graphs H_v.

Definition 5. (*Modular decomposition.*) By decomposing all robust modules (using Proposition 2), we obtain a hierarchical structure yielding the modular decomposition. The *modular decomposition* of a (countable) graph G is defined formally as the *countable* tree $mdec(G)$ of its robust and mps modules. For finite graphs, the notions of a strong and of a robust module coincide. Hence this notion of modular decomposition is equivalent for finite graphs to the usual one which is the finite rooted tree of the strong modules.

We now analyze the structure of the trees $mdec(G)$. 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 $\vee$-tree, a father node may be also a limit node.

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

1. No father node is a limit node.
2. Every father node is the least upper bound of two leaves.
3. A limit node has degree one. Every limit node is the least upper bound of a directed set of non-limit nodes.

Proposition 3. *1. The tree of the modular decomposition of a graph is a modular tree.*
2. Every modular tree is the tree of the modular decomposition of some graph.

Definition 6. (*Embeddings of trees.*)
Let $(T, \leq, \trianglelefteq)$ and $(U, \leq', \trianglelefteq')$ be *ordered* trees. (We denote by $\trianglelefteq$ the family of linear orders $\trianglelefteq_x$ associated with nodes). A $\vee$-*embedding* of T *into* U is an injective mapping $h : T \to U$, such that for all x, y in T: $h(x) \leq' h(y)$ iff $x \leq y$, $h(x \vee y) = h(x) \vee' h(y)$, and if D, $E \in Dir(x, T)$ and $D \trianglelefteq_x E$, then $h(D) \trianglelefteq'_{h(x)} h(E)$, where $h(D)$ is the unique direction in $Dir(h(x), U)$ that contains $\{h(u) \mid u \in D\}$ (it is not the set extension of h on the set D).

If furthermore, $T \subseteq U$ and h is the inclusion mapping, we say that T *is a sub-$\vee$–tree of* U. For trees which are not ordered, the definitions are the same without the conditions on the ordering of directions.

Proposition 4. *1. Every leafy tree T $\vee-$embeds into a unique (up to isomorphism) minimal (for $\vee-$embedding) modular tree denoted by $\widehat{T}$.*
2. T is the sub-$\vee$–tree of $\widehat{T}$ induced by the non-limit nodes.

Proof (Sketch). This construction 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 [5]). We let $\widehat{T}$ consist of the following sets: the set of all directions (i.e., the union of the sets $Dir(x,T)$) and the sets of the form T_u $(= \{w \in T \mid w \leq u\})$ for all nodes u, (a direction can be of the form T_u) ordered by inclusion. The "new" elements in $\widehat{T}$ are the directions which have no greatest element in T.

Proposition 5. *For every graph G, we have $mdec(G) = \widehat{rdec(G)}$, where $rdec(G)$ is the leafy tree of robust modules of G.*

We wish to have a representation of the modular decomposition of a graph G by a relational structure from which G can be defined in a unique way. Hence, it is not enough to know the "abstract" tree $mdec(G)$, we need also represent in a way or another the information attached to each node, that describes which of cases I-IV of Proposition 2 does apply.

The tree $mdec(G)$ can be seen as the syntactic tree of an algebraic expression denoting G, built with substitution operations, possibly of infinite arity. We do not develop here this algebraic aspect (see [11]), but we define a relational structure, somehow equivalent to these algebraic expressions and suitable for expressing properties of modular decomposition in MS logic. Hence we construct from $mdec(G)$ and the five types of nodes (of modules) a binary relational structure $Gdec(G)$, equivalently a vertex- and edge-labelled directed graph, from which G can be defined. We call it the *graph representation of the modular decomposition of G.*

Definition 7. (*Graph representations of modular decompositions.*)

The structure $Gdec(G)$ consists of the tree $mdec(G) = (T, \leq)$, augmented with edges between the sons of the nodes M of T (which are modules of G), in order to represent the edges of G between the submodules corresponding to these sons. It is a straightforward generalization of the similar notion defined in [8]. Formally, we define $Gdec(G)$ from $mdec(G)$ as follows:

For each node M of $mdec(G)$ 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 2):

- if M corresponds to a robust module of type I, we label it by $\oplus$,
- if M corresponds to a robust module of type II, we label it by $\otimes$,
- if M corresponds to a robust module of type III, we label it by $\overrightarrow{\otimes}$, and we define a linear order on the sons of M (which corresponds to the linear order of the strong modules C_i, cf. Proposition 2),
- if M corresponds to a robust module of type IV, we create edges between the sons of M corresponding to the edges of P in an obvious way.

We obtain thus the structure $Gdec(G)$ defined as:

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

where $(T, \leq)$ is the tree $mdec(G)$, $lab_{\oplus}, lab_{\otimes}, lab_{\overrightarrow{\otimes}}$ are unary predicates defining the labels $\oplus$,$\otimes$, $\overrightarrow{\otimes}$ of the nodes of types I,II,III, edg is a binary relation

representing the edges created between sons of nodes of type IV, and *order* is the binary relation such that $order(x, y)$ iff $x \trianglelefteq_{x \vee y} y$ and x, y are sons of $x \vee y$, which implies that $x \vee y$ is labelled by $\overrightarrow{\otimes}$. If G is undirected then *order* and $lab_{\overrightarrow{\otimes}}$ are empty and can be omitted. We can consider $Gdec(G)$ as a graph with three types of edges, corresponding to the binary relations $\leq, edg, order$, and labelled by $\leq, edg, order$. The symbols $\oplus$,$\otimes$, $\overrightarrow{\otimes}$ are thus vertex labels.

The objectives are to prove that $Gdec(G)$ and G can be defined from each other by transformations of relational structures specified by monadic second-order formulas, and thus to obtain that the monadic second-order properties of the modular decomposition of a graph G are monadic second-order expressible in G and vice-versa.

Monadic second-order logic and monadic second-order transformations of structures (called *MS transductions*) are presented in many works by the first author ([6], [8], [7]). Lacking of space, we only recall that an MS transduction (also called sometimes an *MS interpretation*, but this term conflicts with its use in semantics, cf. [5]) is a transformation of relational structures that is specified by MS formulas forming its *definition scheme*. It transforms a structure S into a structure T (possibly over a different set of relations) such that the domain D_T of T is a subset of $D_S \times \{1, ..., k\}$. (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 *FO transduction* is a transduction defined by a first-order definition scheme.

Proposition 6. *A graph G can be defined by an FO transduction from $Gdec(G)$ as a graph, the vertices of which are the leaves of $mdec(G)$.*

Theorem 1. *There is a domain extending MS transduction, let γ, constructing $Gdec(G)$ from $(G, \preccurlyeq)$ where $\preccurlyeq$ is an ω-order. There is an FO transduction δ such that $\delta(Gdec(G)) = G$ for every graph G.*

Proof (Sketch). We describe the main steps of the construction of γ.

Step 1: The notions of a module, of a strong module and of a robust module are MS expressible. The types I, ..., IV of robust modules can also be identified by MS formulas. Moreover, 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 G, $\varphi_1(M, M')$ (resp. $\varphi_2(M, M')$) holds in G iff M is a robust module of type I, (resp. of type II), and M' is one of the corresponding modules C_i.There exists an MS formula $\varphi_3(X, Y, Z)$ such that for all sets of vertices $M, M', M"$, $\varphi_3(M, M', M")$ holds iff M is a robust module of type III, M' is a module C_i, $M"$ is a module C_j and $i < j$. Finally, there exists an MS formula $\varphi_4(X, Y, Z)$ such that for all sets of vertices $M, M', M"$, $\varphi_4(M, M', M")$ holds iff M is a robust module of type IV, M' is a module C_i,

M" is a module C_j and $u_i \to u_j$ in the graph P. All these formulas can be constructed by straightforward translations from the definitions.

Step 2: Given a graph G, we construct the leafy tree $rdec(G)$ of its robust modules. The leaves of $rdec(G)$ are the vertices of the graph. We must define the internal nodes of $rdec(G)$ which correspond to the robust modules. The ω-order $\preccurlyeq$ on vertices is here useful. Each robust module M has at least 2 sons (they are also modules). We let $fl(X)$ be the $\preccurlyeq$-smallest vertex in $X \subseteq V_G$. We let N be the son of M containing $fl(M)$, and we take $fl(M - N)$ as the representative of M. Two robust modules are represented by different vertices. (This would not be the case if we decided to represent M by $fl(M)$). We can thus construct $rdec(G)$, by an MS transduction, as a tree with set of nodes $V_G \times \{1\} \cup R_G \times \{2\}$, where R_G is the set of vertices which represent some module.

Step 3: We know from Proposition 5 that $mdec(G)$ is the completion of $rdec(G)$. This completion is a domain extending MS-transduction, using again an auxiliary ω-ordering. The technique is similar to the one used in the previous step. We represent by some leaf, in a well-defined way, each direction to be completed: using $\preccurlyeq$ we define a linear order on directions ; we represent a direction D by $fl(D')$ where D' is the next one in this order (among directions relative to the same node x) ; a maximal direction (in the case where x has finite degree) is represented by x. Hence if N is the set of nodes of $rdec(G)$, the completed tree $mdec(G)$ has set of nodes $N_{mdec(G)} = N \times \{1\} \cup R \times \{2\} \cup S \times \{3\}$, where R (S) is the set of nodes representing non-maximal (maximal) directions to be completed.

Step 4: An MS transduction transforms $(V_G \cup N_{mdec(G)}, edg_G, \leq_{mdec(G)})$ into $Gdec(G)$. Its definition is a straightforward translation from the definition using Step 1.

Since the composition of several domain extending MS transductions is a domain extending MS transduction, we get a domain extending MS transduction γ that maps $(V_G, edg_G, \preccurlyeq)$ into $Gdec(G)$.

The inverse of γ: We define δ. The vertices of G are the leaves of the tree underlying $Gdec(G)$, hence can be identified by FO formulas. Given two vertices x and y of G, whether there is in G an edge $x \to y$ can be determined from the label of $x \vee y$ in $Gdec(G)$ and, when $x \vee y$ satisfies case III, by the ordering the directions of x and y relative to $x \vee y$ (by using the condition "there exist sons u, v of $x \vee y$ such that $order(u, v)$, $x \leq u$, and $y \leq v$"), and when $x \vee y$ satisfies case IV, the existence of an edge in P between the submodules containing x and y (by the condition "there exist sons u, v of $x \vee y$ such that $edg(u, v)$, $x \leq u$ and $y \leq v$").

3 Universal $\vee-$Trees

It is well-known that the linearly ordered set $\mathbf{Q}$ 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 $\vee$-embeddings.

Definition 8. (*Ordered trees constructed from linear orders.*) We let S and D be two nonempty linearly ordered sets. We let $Aseq(S, D)$ denote the set of

alternating sequences of the form: $s_1 d_1 s_2 d_2 ... d_n s_{n+1}$, for $n \geq 0$; they have at least one occurrence of an element in S. We order them by $\leq_{S,D}$ defined by: $w \leq_{S,D} u$ iff $u = u's, w = u's'w'$ for some u' in $(SD)^*$, some w' in $(DS)^*$, some s, s' in S with $s' \leq_S s$. In particular, $w \leq_{S,D} u$ if $u \leq_{pref} w$ (where $\leq_{pref}$ denotes the prefix order on sequences).

Lemma 1.

1. *The ordered set* $(Aseq(S,D), \leq_{S,D})$ *is a* $\vee$*-tree, denoted by* $T(S,D)$.
2. *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:*

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

$D(0,us)$ is called the *main direction* relative to us. We let $\mathbf{Q}_-$ be the set of negative rational numbers and $\mathbf{Q}_+$ be the set of positive ones. Of course they are both order-isomorphic to $\mathbf{Q}$, but it is more convenient to distinguish them. We let $D = \mathbf{Q}_- + \mathbf{Q}_+$ (i.e., we concatenate as ordered sets $\mathbf{Q}_-$ and $\mathbf{Q}_+$), we let $S = \mathbf{Q}$ and we make the $\vee$-tree $T(S,D)$ into an ordered tree by ordering directions as follows:

$$D(d,us) \trianglelefteq_{us} D(0,us) \trianglelefteq_{us} D(d',us) \text{ for } d \in \mathbf{Q}_-,\ d' \in \mathbf{Q}_+, \text{ and}$$
$$D(d,us) \trianglelefteq_{us} D(d',us) \text{ for } d, d' \in D, d < d'.$$

We denote this tree by $UT(\mathbf{Q}, \mathbf{Q}_-, \mathbf{Q}_+)$ (read *Universal Tree*).

Proposition 7. *Every ordered tree* $\vee$*-embeds into* $UT(\mathbf{Q}, \mathbf{Q}_-, \mathbf{Q}_+)$.

Proof (Sketch). We consider an ordered tree $(T, \leq, \trianglelefteq)$, 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 4.

Step 1: We associate with every $x \in T$ a unique subset $U(x)$ characterized as follows:

1. $U(x)$ is a maximal chain containing 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 $(U(x) - W) \cup (W - U(x))$ is in $U(x)$.

We note for later use that this set is MS definable. Some facts: if $y < x$ and $y \in U(x)$, then $U(y) = U(x)$. If $U(x) \neq U(y)$, then $U(x) \cap U(y) = T^z$ for some z, and if x and y are incomparable, then $z = x \vee y$.

Step 2 : We define a sequence of chains $W_0, ..., W_n, ...$ by:
$W_0 = U(t_0)$, w_1 is the $\preccurlyeq$ -smallest node not in W_0,
$W_1 = U(w_1) - W_0$, ...
w_n is the $\preccurlyeq$ -smallest node not in $W_0 \cup W_1 \cup ... \cup W_{n-1}$,

$W_n = U(w_n) - (W_0 \cup W_1 \cup ... \cup W_{n-1}), ...$

Every node has a finite *depth*, $d(x) = 0$ if $x \in W_0$, $d(x) = 1 + d(p(x))$ if $x \in W_{n+1}$ and $p(x)$ is the $\leq$ −smallest node strictly above all nodes in W_{n+1}. (Note that $W_{n+1} = U(w_{n+1}) - T^{p(x)}$).

Hence x of depth n has a sequence of ancestors $p(x), p^2(x), ..., p^n(x)$ of depths $n-1, n-2, ..., 0$.

Step 3: For each m we fix an (order preserving) embedding $h_m : W_m \to \mathbf{Q}$, and we let $h(x) = h_m(x)$ if $x \in W_m$.

Step 4: We associate with x as in Step 2 the sequence of rational numbers $h(p^n(x))...h(p(x))h(x)$. We have in this way the elements $s_1, s_2, ..., s_{n+1}$ of a sequence $s_1 d_1 s_2 d_2 ... d_n s_{n+1}$ in $ASeq(\mathbf{Q}, \mathbf{Q}_- \cup \mathbf{Q}_+)$, but the d_i's which encode directions are still missing.

Step 5: The main direction relative to x (cf. the lemma) is the one that meets the maximal chain $U(x)$. The set of directions $\trianglelefteq$-smaller than the main direction is linearly ordered by $\trianglelefteq$. We embed it into $\mathbf{Q}_-$ and we do the same into $\mathbf{Q}_+$ for the directions which are $\trianglelefteq$-larger than the main direction. Hence x is finally represented by: $h(p^n(x))f_{n-1}...h(p^2(x))f_1 h(p(x))f_0 h(x)$, where f_i represents the direction of $p^i(x)$ relative to $p^{i+1}(x)$.

We have defined in this way a $\vee$-embedding of $(T, \leq, \trianglelefteq)$ into the tree $UT(\mathbf{Q}, \mathbf{Q}_-, \mathbf{Q}_+)$.

Remarks. 1. Using an obvious extension of the notation, $UT(\mathbf{1}, \varnothing, \mathbf{N})$ is a tree where each infinite tree in the sense of [4] $\vee$-embeds.

2. Fraïssé defines in [14] (Theorem 6.2 of Chapter 10) a (countable) tree $\mathbf{W}$, which is actually the unordered tree underlying $UT(\mathbf{Q}, \varnothing, \mathbf{1})$. All finite or countable trees embed in $\mathbf{W}$. His theorem concerns trees and embeddings, and not ordered trees and $\vee$-embeddings as does our Proposition 7. The tree $\mathbf{W}$ is a binary $\vee$-tree and $\vee$-embeds all binary $\vee$-trees, but only binary $\vee$-trees since it is binary and least upper bounds of pairs of nodes are preserved.

4 Representing Modular Decompositions by Low Degree Relational Structures

Our objective is to represent ordered 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 $(\mathbf{Q}, \leq)$ is isomorphic to the set of nodes of the complete infinite binary tree, a graph of degree 3, ordered appropriately. 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 P can be represented by its Hasse diagram, which may contain $O(m^{1/2})$ edges whereas the directed graph of P has m edges. The same ratio holds for certain dense cographs represented by their modular decompositions. 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.

This motivation does not apply to infinite graphs, but bounds on degrees of infinite structures are nevertheless interesting because they yield structural or logical properties. For examples, every *equational graph* of bounded degree is *prefix recognizable*, see Caucal [3], or Barthelmann [1] for a similar result. MS logic with edge set quantifications is as powerful as MS logic without them for expressing properties of sparse graphs, see [9].

We achieve this goal and we define mutual transformations of relational structures that are MS transductions.

Definition 9. (*Standard binary trees.*) By a *standard binary tree*, we mean a simple directed edge-labelled 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 just a right son or a left son. The *root* is the unique node of indegree 0 and every node is reachable from it by a unique directed path. For a standard binary tree T, and $x, y \in N_T$, we write $x \rightarrow_l y$ if y is the left son of x, $x \rightarrow_r y$ if y is the right son of x, and $x \rightarrow y$ if y is the left or the right son of x.

A linear order, the *in-order*, on N_T is defined by: $x \leq_{in,T} y$ iff $x = y$ or $x \rightarrow_r z \rightarrow^* y$ or $y \rightarrow_l z \rightarrow^* x$ for some z, or $t \rightarrow_l z \rightarrow^* x$ and $t \rightarrow_r z' \rightarrow^* y$ for some t, z, z'.

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

Proposition 8. *1. There exist 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, binary relations lson and rson such that $(N, lson, rson)$ is a standard binary tree T such that $\Omega(T) = (N, \sqsubseteq)$. This tree T is defined from $(N, \sqsubseteq, \preccurlyeq)$ by an FO transduction.*

2. 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 $(N_T, \leq_{in,T})$.

By combining the two constructions, one can represent, using an FO-transduction, the ordered set N as the in-ordered set of *leaves* of a standard binary tree whereas the first one represents it as the in-ordered set of *nodes*.

Proof (Sketch).

1. See [10].
2. This is a classical transformation: for an example, using the notation of trees by terms, $a(b, c)$ is replaced by $*(b, *(a, c))$.

Thus we can represent the universal tree $UT(\mathbf{Q}, \mathbf{Q}_-, \mathbf{Q}_+)$ and whence to all trees via Proposition 7, by standard binary trees with appropriate node labels.

Proposition 9. *There exists a domain extending MS transduction α that associates with every ordered tree T that is also ω-ordered, a node-labelled standard binary ω-ordered tree $W = (N_W, node_T, lson_W, rson_W)$ and an MS-transduction β that defines T from W.*

Intuitively, α encodes T into a binary tree and β is its inverse, the decoding transduction.

Proof (Sketch). We first describe the idea for a tree that is embedded into $UT(\mathbf{Q}, \mathbf{Q}_-, \mathbf{Q}_+)$, by Proposition 7. A node x is described by a sequence of rational numbers $s_1 d_1 s_2 d_2 ... d_n s_{n+1}$ such that s_1 is a node on the chain of nodes of depth 0, d_1 is a direction relative to s_1, saying "in which direction to go next below s_1". This direction indicates a chain of nodes of depth 1, in which s_2 is selected. Then d_2 indicates where to go next, etc... until one reaches s_{n+1}.

By Proposition 8, every rational number can be represented by a path in a standard binary tree, i.e. a word in $\{left, right\}^*$. We concatenate the words representing $s_1, d_1, s_2, d_2, ..., d_n, s_{n+1}$ in this order, and we obtain a path in a standard binary tree. The edges of this path are colored, say in *blue* for those encoding the positions $s_1, s_2, ..., s_{n+1}$ and in *red* for those encoding the directions $d_1, d_2, ..., d_n$. So we can distinguish in a path the portions encoding positions and those encoding directions. It follows that all trees, and in particular $UT(\mathbf{Q}, \mathbf{Q}_-, \mathbf{Q}_+)$ can be represented as subtrees of the complete standard binary tree with colored edges. Actually, coloring an edge is equivalent to coloring its target. So node labels are sufficient and we can use a single unary relation $node_T$.

Proposition 8 says also that for a linear order given with an auxiliary ω-order, the encoding of its elements by paths of the binary tree is definable by an MS transduction. By combining the transductions associated at each depth with the chains W_i and with the sets of directions (cf. the proof of Proposition 7), one obtains the desired one.

We now apply this result to the representation of modular decompositions.

Definition 10. (*Sparse representations of modular decompositions.*) Assuming that, by Proposition 9, $(T, \leq)$ is represented by a node-labelled standard binary tree $(W, node_T, lson_W, rson_W)$, then we define a *sparse representation of the modular decomposition of G* as a structure:

$$Sdec(G) = (W, node_T, lson_W, rson_W, lab_{\oplus}, lab_{\otimes}, lab_{\overrightarrow{\otimes}}, edg).$$

The relation *order* 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 2. *There exists a domain extending MS transduction that associates with an ω-ordered graph G a sparse representation $Sdec(G)$ of its modular decomposition. The structure $Sdec(G)$ is a vertex- and edge-labelled graph of degre $m+3$ where m is the maximum degree of a vertex in a prime factor of G (cf case IV of Proposition 2). There exists an MS-transduction that defines G from $Sdec(G)$.*

Proof. It suffices to combine the MS transductions of Proposition 1 and Proposition 9. The tree $(T, \leq)$ underlying $Gdec(G)$ is not ordered: only the sons of the nodes of type III are linearly ordered, whereas Proposition 9 uses ordered trees. But since an ω-order is available in 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 $Sdec(G)$ follows from the definitions.

5 Concluding Remarks and Questions

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

Finite presentations of countable graphs of several types are studied by Blumensath and Graedel in [2]. One can thus ask whether a finite presentation of G yields one of same type of $Sdec(G)$. A graph G is *VR-equational* (i.e. is the canonical solution of a finite system of equations over so-called *VR operations*) iff it is the image of the standard binary tree $\mathbf{B} = (\{0,1\}^*, lson_{\mathbf{B}}, rson_{\mathbf{B}})$ under an MS transduction (Proposition 2.2 of [2]). If G is VR-equational, and if an ω-order of V_G is MS definable, then by Proposition 2, $Sdec(G)$ 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 $Sdec(G)$ is VR-equational, so is G.

Question 1. Is the former assertion true without the hypothesis that an ω-order of V_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 Theorems 1 and 2 to hold.

The article [2] studies in detail *automatic structures* (also considered in [16] ; they contain 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 $UT(\mathbf{Q}, \mathbf{Q}_-, \mathbf{Q}_+)$ with domain defined as $(L_{\mathbf{Q}}.L_{\mathbf{Q}^*})^*L_{\mathbf{Q}}$ where $L_{\mathbf{Q}} = (0 \cup 11)^*10$ represents $\mathbf{B}$ and $L_{\mathbf{Q}^*} = (0 \cup 1)(0 \cup 11)^*10$ represents the linear order $\mathbf{Q}_- + \mathbf{Q}_+$.

If in the structure $Sdec(G)$ 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 $Fdec(G)$ (that is no longer sparse) from which G can be constructed by an FO transduction. It follows that G is automatic if $Fdec(G)$ is, because the image of an automatic structure under an FO transduction is automatic ([2] Proposition 4.3).

Question 2. For which graphs G is it true that the binary structure $Fdec(G)$ is automatic?

References

1. K. Barthelmann, When can an equational simple graph be generated by hyperedge replacement? *Mathematical Foundations of Computer Science*, Lec. Notes Comput. Sci. 960 (1998) 543-552.
2. A. Blumensath, E. Grädel, Finite presentations of infinite structures: Automata and interpretations, *Theory of Computing Systems* 37 (2004) 641-674.
3. D. Caucal, On the regular structure of prefix rewriting. *Theoretical Computer Science* 106 (1992) 61 - 86.
4. B. Courcelle, Fundamental properties of infinite trees, *Theoretical Computer Science* 25 (1983) 95-169.
5. B. Courcelle, Recursive applicative program schemes, in *Handbook of Theoretical Computer Science vol. B*, J. Van Leeuwen ed., Elsevier, 1990, pp.459-492.
6. B. Courcelle, Monadic second-order graph transductions: A survey. *Theoretical Computer Science*, 126 (1994)53–75.
7. 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.
8. B. Courcelle, The monadic second-order logic of graphs X: Linear orderings, *Theoretical Computer Science* 160 (1996) 87-143.
9. B. Courcelle, The monadic second-order logic of graphs XIV: Uniformly sparse graphs and edge set quantifications. *Theoretical Computer Science* 299 (2003) 1-36.
10. B. Courcelle, The monadic second-order logic of graphs XV: On a Conjecture by D. Seese, to appear in *Journal of Applied Logic*, see: http://www.labri.fr/~courcell/ActSci.html
11. B. Courcelle, C. Delhommé, The modular decomposition of countable graphs, 2005, see: http://www.labri.fr/ ~courcell/ActSci.html.
12. A. Ehrenfeucht, T. Harju, G. Rozenberg, Decomposition of infinite labeled 2-structures, Lec. Notes Comput. Sci. 812 (1994) 145-158.
13. 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.
14. R. Fraïssé, Theory of relations, *Studies in logic* vol. 118, North-Holland, 1986 (Second edition, Elsevier, 2000).
15. D. Kelly, Comparability graphs, in *Graphs and order*, I. Rival ed., D. Reidel Pub. Co., 1985, pp. 3-40.
16. B. Khoussainov, A. Nerode, Automatic presentations of structures, in *Logic and Computational Complexity*, Lec. Notes Comput. Sci. 960 (1995) 367-392.
17. R. Möhring, R. Radermacher, Substitution decomposition of discrete structures and connections with combinatorial optimization, *Annals Discrete Maths* 19 (1984) 257-356.