Proc. of ICDM 2006, Pages 205–216

Graph Equivalences and Decompositions Definable in Monadic Second-Order Logic. The Case of Circle Graphs

Bruno Courcelle
Bordeaux 1 University, LaBRI and Institut Universitaire de France
Email: courcell@labri.fr

1 Introduction

Many graph properties and graph transformations can be formalized in *Monadic Second-Order logic*. This language is the extension of *First-Order logic* allowing variables denoting sets of elements. In the case of graphs, these elements can be vertices, and in some cases edges. Monadic second-order graph properties can be checked in linear time on the class of graphs of *tree-width* at most k for any fixed k. These properties are *Fixed Parameter Linear*, for tree-width as a parameter.

Monadic second-order logic as a language for specifying graph properties is interesting from several different points of view: we already mentioned complexity, but another point of view is that of *Graph Grammars*. For logicians, monadic second-order logic is attractive because relatively many classes of structures have a *decidable theory* for this language.

In this communication we will discuss the point of view of *Graph Theory*. Many graph properties concerning colorings, forbidden configurations, connectivity are expressible in Monadic Second-Order logic, but also many graph theoretical constructions like the canonical decompositions of a graph in *2- and 3-connected components*, its *modular* and its *split decompositions*.

We will review a number of cases where a set of graphs or of combinatorial objects is characterized by a common hierarchical decomposition. In the cases we will consider, the decomposition can be formalized in monadic second-order logic and from it, all graphs or objects of the corresponding set can be defined by monadic second-order formulas with the help of auxiliary data like a k-tuple of sets of vertices or a linear order on the vertices.

This general description applies to the following sets of graphs or combinatorial objects :

1. the connected graphs having the same cycle matroid as a given graph,
2. the different planar embeddings of a planar connected graph,
3. the transitive orientations of a comparability graph,
4. the interval models of an interval graph,
5. the pairs of linear orders representing a partial order of dimension 2,
6. the chord representations of a circle graph,
7. the systems of intersecting closed curves in the plane having a same Gauss multiword.

In the first three sections, we will present the main concepts underlying these descriptions. In the fourth one we will apply them to *Circle Graphs*.

For fixed parameter tractability the references are the books [21] and [23]. For the use of monadic second-order logic in the theory of graph grammars the reference is the book chapter [4]. For the links between graph decompositions and the decidability of monadic second-order logic the references are the articles [5, 9, 17]. The results for the seven above cases are explicit or implicit in the articles [6–12].

2 Graph Properties Expressible in Monadic Second-Order Logic

Let $R = \{A, B, C, ...\}$ be a finite set of *relation symbols*; each of them, say A, is given with a nonnegative integer $\rho(A)$ called its *arity*. We denote by $\mathcal{STR}(R)$ the set of *finite* R-structures $S = \langle D_S, (A_S)_{A \in R} \rangle$ where $A_S \subseteq D_S^{\rho(A)}$ for each A. A simple graph G can be defined as the $\{edg\}$-structure $S(G) = \langle V_G, edg_G \rangle$ where V_G is the set of vertices and $edg_G \subseteq V_G \times V_G$ is a binary relation representing the edges. For undirected graphs, the relation edg_G is symmetric. If in addition we need vertex labels, we will represent them by unary relations. Graphs, either simple or not, can also be represented by the richer incidence structure $I(G) = \langle V_G \cup E_G, inc_G \rangle$, where E_G, the set of edges, is now part of the domain and $inc_G(e, u, v)$ holds iff e is an edge from u to v (or between u and v if G is undirected).

Monadic Second-Order logic is the extension of *First-Order logic* by variables denoting subsets of the domains of the considered structures, and new atomic formulas of the form $x \in X$ expressing the membership of x in a set X. (Uppercase letters will denote set variables, lowercase letters will denote first-order variables). In the sequel *MS* will abbreviate Monadic Second-order.

We denote by $MS(R, W)$ the set of *MS formulas* written with the set R of relation symbols and having their free variables in a set W consisting of *first-order as well as of set variables*. As a typical and useful example, we give an MS formula τ 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 given in the structure but is defined by an MS formula, then one replaces $A(u, v)$ by this formula with appropriate substitutions of variables.

An *MS property* of the structures S of a class $\mathcal{C} \subseteq \mathcal{STR}(R)$ is a property $\mathcal{P}$ such that for all $S \in \mathcal{C}$:

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

for some fixed formula φ in $MS(R, \varnothing)$. We say also that $\mathcal{P}$ is *MS expressible*. (The notation $S \vDash \varphi$ means that the logical formula φ is true in the structure S.)

Example. For each k one can construct a formula φ_k in $MS(\{edg\}, \varnothing)$ such that for every graph G:

$$S(G) \vDash \varphi_k \ \text{ if and only if } G \text{ is } k\text{-colorable.}$$

Here is the formula φ_3:

$\exists X_1, X_2, X_3[\forall x(x \in X_1 \vee x \in X_2 \vee x \in X_3) \wedge$
$\quad \forall x(\neg(x \in X_1 \wedge x \in X_2) \wedge \neg(x \in X_2 \wedge x \in X_3) \wedge \neg(x \in X_1 \wedge x \in X_3))$

$\wedge\forall u, v(edg(u,v) \wedge u \neq v \Longrightarrow \neg(u \in X_1 \wedge v \in X_1) \wedge \neg(u \in X_2 \wedge v \in X_2)$
$\wedge\neg(u \in X_3 \wedge v \in X_3))]$.

Many properties based on the existence of paths like connectivity, strong connectivity, biconnectivity can be expressed in MS logic with the help of the above written formula τ that defines the reflexive and transitive closure of a binary relation.

If H is a simple loop-free undirected graph, then the property that a graph G contains H as a minor is MS expressible. Let H have vertices $1, ..., n$. Then G contains H as a minor iff it has pairwise disjoint sets of vertices $X_1, ..., X_n$ that induce connected subgraphs of G and are such that for every edge $i-j$ in H, there exists an edge of G linking one vertex of X_i and one of X_j. These conditions are easily expressible by an MS formula μ_H. Every minor-closed class of undirected graphs is characterized by finitely many excluded minors, and is thus definable by an MS formula. This is the case of planar graphs which are characterized by the formula $\neg\mu_{K_5} \wedge \neg\mu_{K_{3,3}}$.

All properties of a graph G considered above are expressed via the relational structure $S(G)$. More properties are MS expressible via the relational structure $I(G)$. In particular the existence of a Hamiltonian cycle in a graph G is MS expressible by using $I(G)$ and not by using $S(G)$. See [4, 5].

Every property that is MS expressible via the relational structure $I(G)$ is *fixed parameter linear*, where the parameter is tree-width. Every property that is MS expressible via the relational structure $S(G)$ is *fixed parameter cubic*, where the parameter is clique-width (this follows from [15] and [27], the latter article showing that cubic time is enough for finding an appropriate decomposition of the given graph). However, the paradigm *MS logic + bounded tree-width or bounded clique-width* extends to other algorithmic problems than just the verification of properties. One can compute efficiently optimization functions [16], one can enumerate queries efficiently ([1, 13, 22]), one can label graphs in order to facilitate answers to queries [18].

3 Monadic Second-Order Transductions

The construction of MS formulas for expressing graph properties is not always an easy task. Our articles contribute to building a toolbox, the big hammer of which is the notion of *Monadic Second-Order transduction* (*MS transduction* in short). We first present a very simple case.

The *edge-complement* of a simple loop-free undirected graph G, that we denote by $\overline{G}$, can be defined in logical terms. The edge relation of $\overline{G}$ is defined from that of G by:

$edg_{\overline{G}}(x,y) \Longleftrightarrow\ x \neq y \wedge \neg edg(x,y)$.

Hence we define $\overline{G}$ from G by defining its edge relation by a logical (here first-order) formula to be evaluated in G. We say that the edge-complement transformation is a first-order transduction.

The notion of an *MS transduction* generalizes this example on several respects, and in particular by the use of MS formulas instead of first-order ones. As in *Language Theory*, a binary relation $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$ where $\mathcal{A}$ and $\mathcal{B}$ are sets of relational structures is called a *transduction*: $\mathcal{A} \to \mathcal{B}$. An *MS transduction* 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 $D_S \times [k]$. ($[k] = \{1, ..., k\}$). 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. (The main reference for this section is [4].)

We let R and Q be two finite sets of relation symbols. Let W be a finite set of set variables, called the *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 $k > 0, Q^*k := \{(q, \vec{j}) \mid q \in Q, \vec{j} \in [k]^{\rho(q)}\}$,
$\varphi \in MS(R, W)$, $\psi_i \in MS(R, W \cup \{x_1\})$ 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) $D_T = \{(d, i) \mid d \in D_S, i \in [k], (S, \gamma, d) \models \psi_i\}$

(iii) for each q in Q : $q_T = \{((d_1, i_1), \cdots, (d_t, i_t)) \in D_T^t \mid (S, \gamma, d_1, \cdots, d_t) \models \theta_{(q,\vec{j})}\}$, where $\vec{j} = (i_1, \cdots, i_t)$ and $t = \rho(q)$.

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

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

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

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

An MS-transduction is defined as a binary relation. Hence it can be seen as a "nondeterministic" partial function associating with an R-structure one or more Q-structures. However, it is not really nondeterministic because the different outputs come from different choices of parameters. In the case where $W = \emptyset$ it defines a partial function. It may also happen that different choices of parameters yield isomorphic output structures. This is the case in the example of edge contraction detailed below. We will refer to the integer k by saying that Δ and $\mathcal{D}_\Delta$ are k-*copying*; if $k = 1$ we will say that they are *noncopying*. A noncopying definition scheme can be written more simply: $\Delta = (\varphi, \psi, (\theta_q)_{q \in Q})$.

Example. *Edge contraction.*

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

It is formally defined as $\langle V_H, edg_H \rangle$ where $V_H = V_G \,/\sim$, $\sim$ is the equivalence relation such that $x \sim y$ if and only if x and y are linked by an undirected path made

of $\varepsilon-$edges, and $edg_H([u],[v])$ holds if and only if $x \in [u]$, $y \in [v]$ for some (x,y) in edg_G ($[u]$ denotes the equivalence class of u). The MS formula $\xi(x,y)$ defined as:

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

expresses $x \sim y$. For defining V_H we must select a set containing one and only one vertex of each equivalence class. This can be done with a set variable Y that will be a parameter of the MS transduction, satisfying the formula $\varphi(Y)$ saying that for every x there is a unique y such that $y \in Y \wedge \xi(x,y)$ holds. Edge contraction can thus be defined by the transduction with noncopying definition scheme $\Delta = (\varphi, \psi, \theta_{edg})$ where $\psi(Y,x)$ is $x \in Y$ and:

$\theta_{edg}(Y,x,y)$ is the formula $x \in Y \wedge y \in Y \wedge \exists u,v.[edg(u,v) \wedge \xi(x,u) \wedge \xi(y,v)]$.

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

The Fundamental Property of MS Transductions

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

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

Proposition 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 $\gamma : W \longrightarrow \mathcal{P}(D)$, for every assignment $\eta : V' \longrightarrow \mathcal{P}(D)$ 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 $(def_\Delta(S,\gamma), \eta^k) \models \beta$.

We call $\beta^\#$ the *backwards translation* of β relative to the transduction def_Δ. The composition of two transductions is defined as the composition of the corresponding binary relations (equivalently, multivalued functions). From Proposition 1 we get:

Proposition 2. 1) *The composition of two MS transductions is an MS transduction.*

2) *The inverse image of an MS-definable class of structures under an MS transduction is MS-definable.*

The mapping from $I(G)$ to the set of spanning forests of a graph G or to the set of its subgraphs, or of its minors (represented by their incidence structures) are MS transductions. The mapping from $S(G)$ to the set of its *induced* subgraphs for a simple graph G is an MS transduction (a subgraph H of G is represented by $S(H)$).

4 Graph Decompositions Defined by MS Transductions

Of particular interest are *graph decompositions*. They are useful for the construction of efficient algorithms, and also because they give structural descriptions of the considered graphs. The general results of [19, 20] define canonical decompositions which include as particular instances the *modular decomposition* and the *Tutte decomposition of a graph in 3-connected components*. (Tree-decompositions, rank decompositions [27], optimal decompositions for clique-width are not canonical.)

We show that these canonical decompositions can be defined by monadic second-order formulas "inside" the considered graphs. In our language the mapping from a (linearly ordered) graph to its decomposition of this type is an MS transduction. We first review informally the *modular decomposition*. (See [25, 26] for thorough studies of this notion).

Modular decomposition

The modular decomposition of a graph is a canonical expression of this graph in terms of (nested) substitutions. Let G and H be two simple loop-free undirected graphs with disjoint sets of vertices. The *substitution* of H for a vertex u of G is the graph $G[H/u]$ defined as the union of G and H, *minus* the vertex u and the incident edges, *plus* edges between w and all vertices of H for every edge $w - u$ in G. Substitutions can be done *in parallel* (disjoint graphs for distinct vertices) giving $G[H_1/u_1, ..., H_n/u_n]$ and *nested* in expressions like $G[H[L/u]/v]$.

Let K be a graph and M a set of vertices. The graph K can be expressed as $G[H/u]$ with $M = V_H$ iff M is a *module* of K, that is, a set of vertices such that every vertex not in M is linked either to no vertex of M or to all of them.

A module is *strong* if does not overlap any module. (Two sets *overlap* if they have a nonempty intersection and none is a subset of the other.) The strong modules of a graph G form a tree for inclusion, called its *modular decomposition*. This tree is the syntactic tree of a canonical expression of G in terms of substitutions. (The modular decomposition is called *substitution decomposition* in some works). This tree can be enrich by labels and additional edges, and this gives the *graph representation of the modular decomposition.* From it the graph G can be reconstructed.

The mapping from a graph G given with a linear order $\preccurlyeq$ of its vertices is an MS transduction. We sketch the construction which is given in [6, 10, 14]. There exists an MS formula $\varphi(X)$ that is valid in the given graph iff X is a strong modules. To build the tree of strong modules, we specify in each strong module that is not a singleton a *representing* vertex x that will be the corresponding non-leaf node of the modular decomposition. This specification is done by an MS formula $\psi(x, X)$. Furthermore we need that distinct strong modules are represented by distinct vertices. The order $\preccurlyeq$ on vertices is here useful. For a set $Y \subseteq V_G$, we let $fl(Y)$ be the $\preccurlyeq$-smallest vertex of Y. Each strong module M contains a unique maximal proper strong module N such that $fl(N)$ is minimal. (Maximal is understood for set inclusion). We take $fl(M - N)$ as the representative vertex of M. Two non-singleton strong modules are represented by different vertices. (This would not be the case if we would represent M by $fl(M)$). We can thus construct the tree of strong modules 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 non-singleton strong module. The remaining parts of the construction are routine. Note that this construction uses a linear order on

vertices, but for any two linear orders, the corresponding outputs are isomorphic relational structures.

Split decomposition:
This decomposition due to Cunningham ([20]) generalizes the modular decomposition. We review it briefly. We only consider simple loop-free directed graphs. An undirected edge can be considered as a pair of opposite directed edges. Hence, this definition also applies to undirected graphs.

A *split* of a strongly connected graph G is a bipartition $\{A, B\}$ of V_G such that A and B have at least 2 elements and $E_G = E_{G[A]} \cup E_{G[B]} \cup (A_1 \times B_1) \cup (B_2 \times A_2)$ for some $A_i \subseteq A$, $B_i \subseteq B$. If $\{A, B\}$ is a split, then G can be expressed as the union of $G[A]$ and $G[B]$ linked by two directed, complete bipartite graphs. (Since G is strongly connected the set $(A_1 \times B_1) \cup (B_2 \times A_2)$ is not empty). If G is undirected, then strong connectivity is just connectivity.

The inverse of splitting is the *join operation*, defined as follows.

Let H and K be two disjoint graphs with distinguished vertices h in H and k in K. We define $H \boxtimes_{(h,k)} K$ as the graph with set of vertices $V_H \cup V_K - \{h, k\}$ and edges $x \longrightarrow y$ such that, either $x \longrightarrow y$ is an edge of H, or an edge of K, or we have $x \longrightarrow h$ in H and $k \longrightarrow y$ in K, or we have $h \longrightarrow y$ in H and $x \longrightarrow k$ in K.

If $\{A, B\}$ is a split, then $G = H \boxtimes_{(h,k)} K$ where H is $G[A]$ augmented with a new *marker* vertex h and edges $x \longrightarrow h$ whenever there are in G edges from x to some u in B, and edges $h \longrightarrow x$ whenever there are edges from some u in B to x. The graph K is defined similarly from $G[B]$, with a new vertex k. The graphs H and K have at least 3 vertices and strictly less vertices than G.

A *decomposition* of a strongly connected graph G is defined as follows: $\{G\}$ is the only decomposition of size 1; if $\{G_1, ..., G_n\}$ is a decomposition of size n, and $G_n = H \boxtimes_{(h,k)} K$, then $\{G_1, ..., G_{n-1}, H, K\}$ is a decomposition of G of size $n + 1$. The graphs G_i are called the *components* of the decomposition. The graph G can be reconstructed without ambiguity provided the marker vertices and their matchings are specified. The components of a decomposition form an unrooted tree for the "matching" relation. By decomposing iteratively a strongly connected graph and by using only *good splits* defined as those that do not overlap any other, one gets *the Unique (Canonical) Split Decomposition.* The restriction to good splits generalizes the restriction to strong modules. These restrictions are important for having canonical decompositions. Figure 4.1 shows an undirected graph G and Figure 4.2 its split decomposition.

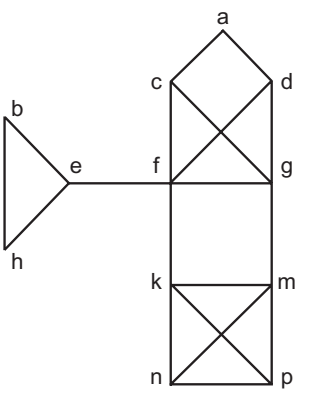

Figure 4.1: Graph G

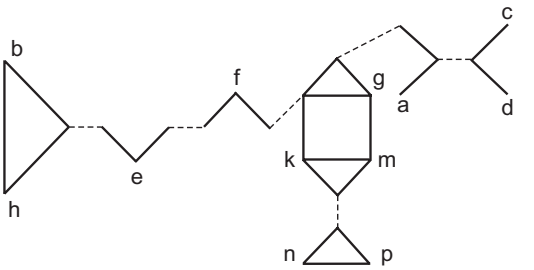

Figure 4.2: The split decomposition of G.

Theorem 1 ([6, 10]). *There are MS transductions that associate with a linearly ordered graph the graph representations of its modular and split decompositions. Up to isomorphism, the constructed relational structures do not depend on the chosen ordering of the vertices.*

5 Circle Graphs

A *circle graph* is the intersection graph of a set of chords of a circle. An equivalent combinatorial characterization can be given in terms of words *where each letter has two occurrences.* If a letter represents a chord, a set of chords is a word corresponding to the sequences of extremities of chords read around the circle, and the chords represented by a and b intersect iff the word can be written $aubvawbx$ for some words u, v, w, x.

Circle graphs which are *prime*, i.e., indecomposable for the *split decomposition* have *unique representations* as words (or as sets of chords), where *unique* is meant up to some obvious transformations. The main results are the following ones:

1. If two words define the same connected circle graph and have the same *subword of first occurrences of letters*, then they are equal.

2. If a circle graph is prime, the unique word representing it can be constructed by formulas of MS logic.

3. If a circle graph is not prime, then all its representations by double occurrence words can be defined from it and the linear orders of its set of vertices, by a fixed MS formula.

Result 2 uses MS formulas written with set predicates of the form $Even(X)$ expressing that a set X has even cardinality. These formulas will be called C_2MS formulas. They are needed because the proof uses a result by Courcelle and Oum [17] showing that the characterization of circle graphs by the three *excluded vertex-minors* given by Bouchet [3] is expressible by C_2MS formulas. *Even* helps to express computations in GF(2), and these computations occur because the notion of a vertex-minor is handled through that an *isotropic system*, which is a vector space over GF(2).

Result 3 makes a crucial use of Theorem 1 for *split decompositions*. Every circle graph has a canonical split decomposition the components of which are prime circle graphs, cliques and stars. For a circle graph, the prime components have constructible word representations by Result 2. The other components have word representations constructible by means of the linear order. By varying the linear

order one gets all word representations (the orders corresponding to the first occurrence words of the word representations of the given graph actually suffice). The results of this section are proved in [11].

Concerning the unique representability of prime circle graphs, a quite similar result holds for *comparability graphs* (see [25]). Those which are undecomposable for the *modular decomposition* have *unique transitive orientations* (actually they have two, one and its reversal) and it is possible to construct these orientations by MS formulas by a proof similar to the one used for circle graphs. This proof uses a characterization of comparability graphs by forbidden induced subgraphs. This characterization is expressible in MS logic although there are infinitely many forbidden subgraphs. See [9].

Circle graphs.
Let A be a countable set called the *set of letters*. We let W be the set *double occurrence words i.e.,* of finite nonempty words over A having two occurrences or no occurrence of each letter. We let $V(w)$ be the set of letters occuring in w and $G(w)$ be the graph with set of vertices $V(w)$ and an undirected edge between a and b iff $w = u_1au_2bu_3au_4bu_5$ or $w = u_1bu_2au_3bu_4au_5$ for some $u_1, ..., u_5$ in A^*. The graphs $G(w)$ are also called *circle graphs*. It is clear that $G(w) = G(w')$ if $w' = \widetilde{w}$ (the *mirror image* of w) or if w and w' are *conjugate*, denoted by $w \sim w'$, which means $w = uv$ and $w' = vu$ for some u, v in A^*; w and w' are *equivalent*, denoted by $w \equiv w'$, iff either $w \sim w'$ or $\widetilde{w} \sim w'$. Two equivalent words represent the same circle graph. Figure 5.1 shows a circle graph G and its chord representation defined by the word : $axbcuyvbycauxv$.

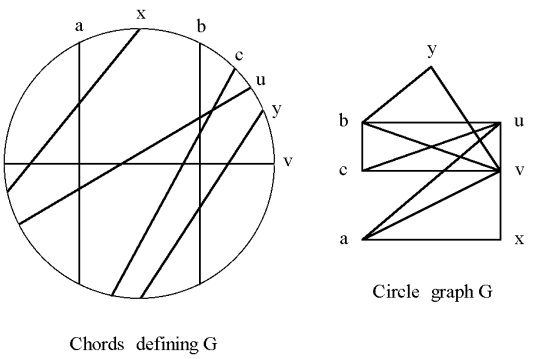

Figure 5.1: A circle graph and a chord representation of it.

A circle graph G is *uniquely representable* if $G = G(w) = G(w')$ implies $w \equiv w'$. The circle graphs with at most 3 vertices are uniquely representable, so are C_4, P_4, the graph K_4 minus one edge. The graphs K_4, S_3 are not. To take an example the star S_3 with center a is represented by the two inequivalent words *abcdadcb* and *acbdadbc*.

Theorem 2 ([11]). *A circle graph with at least 5 vertices is uniquely representable iff it is prime for the split decomposition.*

Proof. "if" is proved in [2, 24]. "Only if" is claimed in [24] but not actually proved. □

Theorem 3 ([11]). *There exist C_2MS formulas that associate with every prime circle graph G a word $w \in W$ such that $G(w) = G$.*

A double occurrence word can be defined for logical treatment as a directed circuit with a binary relation representing pairs of positions bearing the same letter. Whether this letter is a or b does not matter.

Eulerian trails of 4-regular graphs.

We need a lemma concerning the Eulerian trails of 4-regular graphs without loops and multiple edges. We prove that these trails can be encoded by tuples of sets, hence defined and used by MS formulas.

Lemma. *There exist MS formulas that associate with every connected 4-regular undirected graph H the structures $< V_H, edg_H, edg_{G(E)} >$ for all Eulerian trails E of H, where $edg_{G(E)}$ is the edge relation of the corresponding circle graph $G(E)$.*

Proof sketch of Theorem 3. Let $w \in W$, $a, b \in V(w)$, $a \neq b$. We say that a and b are *neighbours in* w if $w \equiv abw'$ for some w' in A^*. This means that on the representation of $G(w)$ by intersecting chords, chords a and b have two consecutive ends on the circle. If G is prime with at least 5 vertices, this notion depends only on G, and not on the word w representing it. Each letter occurring in w has *four different neighbours*. We let $N(G)$ be the *graph of neighbourhood*, with set of vertices $V(w)$ and an edge $a-b$ iff a and b are neighbours in G. This graph is 4-regular. We can prove that its adjacency relation is definable by a C_2MS formula over the given prime circle graph G, and that w can be constructed from $N(G)$. Figure 5.2 shows with solid lines the graph $N(G)$ for the graph G associated with the word : *axbcuyvbycauxv*. The dotted lines around the vertices show the Eulerian trail which corresponds to the chord representation of G, i.e., to the circular sequence of the ends of the chords corresponding to letters.

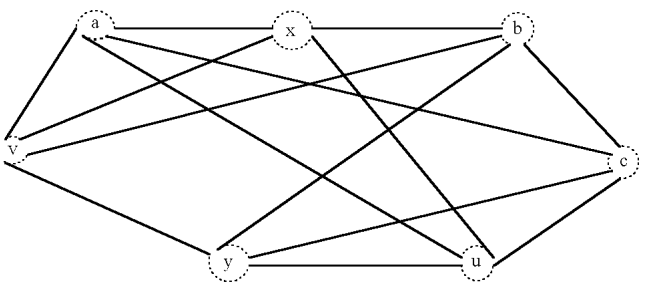

Figure 5.2: The neighbourhood graph *N(G)*.

For $a, b \in V_G (\subset A), a \neq b$, we let $G(a, b; u, v)$ be the graph G augmented with the path $a - u - v - b$ where $u, v \in A - V_G$.

Claim 1. $G(a, b; u, v)$ is a circle graph iff a, b are neighbours in G.

Claim 2. That a and b are neighbours in G is expressible by a C_2MS formula.

Hence the mapping associating $N(G)$ with a prime circle graph G is a C_2MS transduction. This claim uses the C_2MS characterization of circle graphs given in [17]. That a given graph G is a prime circle graph with at least 5 vertices is a C_2MS property. Assuming this satisfied, one can build from G and by C_2MS formulas (Claim 2) the 4-regular graph $N(G)$. This graph is connected and has an Eulerian trail F such that $G(F) = G$. The Eulerian trails of $N(G)$ are defined by tuples of sets of vertices (by the Lemma) and an MS formula can select the one defining F. □

First occurrence words:

For every word w in A^* we denote by $F(w)$ the subword of w consisting of the first occurrence of each letter. For an example, $F(abbdacdcefef) = abdcef$.

Theorem 4 ([11]). *If w, w' are double occurrence words such that $G(w) = G(w')$ is connected and $F(w) = F(w')$, then $w = w'$.*

Proof. The proof is by induction on the length of w. □

Theorem 5 ([11]). *There exist MS formulas that associate with $(G, \preccurlyeq)$ where G is a circle graph and $\preccurlyeq$ a linear order on V_G, the unique double occurrence word w representing it such that $F(w) = (V_G, \prec)$, provided such a word does exist.*

Proof sketch. On a structure given with a linear order, the set predicate $Even(X)$ can be expressed by an MS formula (see [4]). Hence on these structures, every C_2MS formula can be translated into an equivalent MS formula. In particular, an MS formula can check that the given graph is a circle graph. By Theorem 1 (Theorem 4.21 of [10]), one can construct from $(G, \preccurlyeq)$ its split decomposition $Split(G)$ by an MS transduction. For those components of $Split(G)$ which are prime, an MS formula can build double occurrence words representing them by Theorem 3. For the other components, which are isomorphic to stars and to cliques, the linear order $\preccurlyeq$ makes it possible to define by MS formulas representing double occurrence words. One obtains a tree of relational structures $S_1, ..., S_k$ representing double occurrence words for the k components of $Split(G)$. From this tree one can produce a double occurrence word for G. It remains to check that it matches the first occurrence word defined by $F(w) = (V_G, \prec)$. □

References

[1] G. Bagan, MSO queries on tree decomposable structures are computable with linear delay, *Computer Science Logic 2006, Lec. Notes Comput. Sci.*, **4207** (2006). pp. 167–181.

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

[3] A. Bouchet, Circle graph obstructions, *J. Combinatorial Theory B*, **60** (1994), pp. 107–144.

[4] 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*, pages 313–400. World Scientific (1997).

[5] B. Courcelle, The Monadic Second-Order Logic of Graphs VIII: Orientations, *Ann. Pure Appl. Logic*, **72** (1995), pp. 103–143.

[6] B. Courcelle, The Monadic Second-Order Logic of Graphs X: Linear Orderings, *Theor. Comput. Sci.*, **160** (1996), pp. 87–143.

[7] B. Courcelle, The Monadic Second-Order Logic of Graphs XI: Hierarchical Decompositions of Connected Graphs, *Theor. Comput. Sci.*, **224** (1999), pp. 35–58.

[8] B. Courcelle, The monadic second-order logic of graphs XII: Planar Graphs and Planar Maps, *Theor. Comput. Sci.*, **237** (2000), pp. 1–32.

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

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

[11] B. Courcelle, Circle graphs and Monadic Second-order logic, *J. of Applied Logic*, in press.

[12] B. Courcelle, Diagonal walks in plane graphs and local duality, March 2005, http://www.labri.fr/perso/courcell/ActSci.html

[13] B. Courcelle, Linear delay enumeration and monadic second-order logic, March 2006, http://www.labri.fr/perso/courcell/ActSci.html, to appear in Discrete Applied Mathematics

[14] B. Courcelle and C. Delhommé, The modular decomposition of countable graphs: Constructions in Monadic Second-Order Logic, *Theoretical Computer Science*, **394** (2008), pp. 1–38.

[15] B. Courcelle, J. Makowsky and U. Rotics, On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic, *Discrete Applied Mathematics*, **108** (2001), pp. 23–52.

[16] B. Courcelle and M. Mosbah, Monadic Second-Order Evaluations on Tree-Decomposable Graphs, *Theor. Comput. Sci.*, **109** (1993), pp. 49–82.

[17] B. Courcelle and S. Oum, Vertex-minors, monadic second-order logic and a conjecture by Seese, *J. Comb. Theor. B*, **97** (2007), pp. 91–126.

[18] B. Courcelle and R. Vanicat, Query efficient implementation of graphs of bounded clique-width, *Discrete Applied Mathematics*, **131** (2003), pp. 129–150.

[19] W. Cunnigham and J. Edmonds, A combinatorial decomposition theory, *Can. J. of Math.*, **XXXII** (1980), pp. 734–765.

[20] W. Cunnigham, Decomposition of directed graphs, *SIAM J. Algebraic Discrete Methods*, **3** (1982), pp. 214–228.

[21] R. Downey and M. Fellows, Parameterized Complexity, Springer-Verlag (1999).

[22] J. Flum, M. Frick and M. Grohe, Query evaluation via tree-decompositions. *J. ACM*, **49** (2002), pp. 716–752.

[23] J. Flum and M. Grohe, Parameterized Complexity Theory, Springer (2006).

[24] C. Gabor, W. Hsu and K. Supowit, Recognizing circle graphs in polynomial time, *J. of ACM*, **36** (1989), pp. 435–473.

[25] D. Kelly, Comparability graphs, in *Graphs and order*, I. Rival ed., D. Reidel Pub. Co., 1985, pp. 3–40.

[26] R. Möhring and F. Radermacher, Substitution decomposition for discrete structures and connections with combinatorial optimization, *Annals of Discrete Maths*, **19** (1984), pp. 257–356.

[27] S. Oum, Approximating Rank-Width and Clique-Width Quickly, *Proc. WG 2005, Lec. Notes Comput. Sci.*, **3787** (2005), pp. 49–58.