On the model-checking of monadic second-order formulas with edge set quantifications.

Bruno Courcelle

On the model-checking of monadic second-order formulas with edge set quanti…cations.

Bruno Courcelle
Institut Universitaire de France and
Université Bordeaux-1, LaBRI, CNRS
351, Cours de la Libération, 33405, Talence, France
courcell@labri.fr

September 18, 2010

Abstract

We extend clique-width to graphs with multiple edges. We obtain …xed-parameter tractable model-checking algorithms for certain monadic second-order graph properties that depend on the multiplicities of edges, with respect to this "new" clique-width. We de…ne special tree-width, the variant of tree-width relative to tree-decompositions such that the boxes that contain a vertex are on a path originating from some …xed node. We study its main properties. This de…nition is motivated by the construction of …nite automata associated with monadic second-order formulas using edge set quanti…cations. These automata yield …xed-parameter linear algorithms with respect to tree-width for the model-checking of these formulas. Their construction is much simpler for special tree-width than for tree-width, for reasons that we explain.

1 Introduction

It is well-known that the model-checking problem for graph properties expressed by monadic second-order sentences with edge set quanti…cations is …xed-parameter linear for tree-width as parameter ([5], [12], [14]) and that, for graph properties expressed by the (basic) monadic second-order sentences without edge set quanti…cations, it is …xed-parameter cubic for clique-width ([9], together with the approximate parsing algorithm of [19]; see Chapter 6 of [6]).

Because of the usually considered representation of graphs by relational structures, the graph properties of the second type cannot take into account the multiplicity of edges. In this article, we extend clique-width, de…ned up to

now for simple graphs only, to graphs with multiple edges. We use the same "clique-width graph operations" as for simple graphs, but we let them act on graphs with multiple edges. We also extend the representing logical structures and, accordingly, the vocabulary of monadic second-order formulas, without using edge set quanti…cations. (The idea governing this extension is that, without using edge set quanti…cations, we can nevertheless count multiple edges up to a threshold or modulo a …xed integer). The …xed-parameter cubic algorithm (with clique-width as parameter) extends to this more general situation.

The …xed-parameter tractable monadic second-order model-checking algorithms for tree-width and clique-width as parameters are based on constructions of …nite automata on terms. It appears that these constructions are more complicated for the terms related to tree-width (these terms represent tree-decompositions algebraically) than for those related to clique-width. Analysing this di¢culty lead us to the de…nition of particular tree-decompositions called special tree-decompositions, that yield the notion of special tree-width. This parameter, that is new to our knowledge takes values between path-width and tree-width. Graphs of tree-width 2 have unbounded special tree-width. Special tree-width can be de…ned in terms of the generalized "clique-width operations" that operate on graphs with multiple edges. The corresponding constructions of …nite automata from monadic second-order sentences using edge set quanti…cations are as easy as in the case where clique-width is the intended parameter.

All necessary de…nitions will be given, but we will frequently refer to de…nitions (of secondary importance) and to the constructions developed in detail in Chapters 2 and 6 of [6]. We will use as much as possible the notation and terminology of this book, but this article introduces de…nitions that will not be included in it. Section 2 introduces the clique-width of graphs with multiple edges, Section 3 de…nes the relevant extension of counting monadic second-order logic. The applications to model-checking are in Section 4. Special tree-width is de…ned and studied in Section 5. Its application to model-checking is in Section 6 where we also compare it to tree-width with respect to the construction of automata. Section 7 is a short conclusion.

2 Graphs algebras

All graphs and relational structures will be …nite.

De…nition 1 : Graphs

We will consider …nite graphs that can have loops and multiple (or parallel) edges. We will not consider a undirected graph as a directed graph such that each edge has an opposite edge.

A concrete graph G is a triple $(V_G, E_G, vert_G)$ with vertex set V_G, edge set E_G and incidences de…ned by the mapping $vert_G$ such that $vert_G(e)$ is the

set of end vertices of an edge e if G is undirected (it has a single element if e is loop), and $vert_G(e)$ is the pair (x, y) if G is directed and e links x to y. The notation $e : x \text{ ¡ } _G\ y$ means that e is an undirected edge that links x and y, and $e : x \text{ ¡ ! } _G\ y$ means that e is a directed edge from x (its tail) to y (its head). In both cases, we have $x = y$ if e is a loop. Two edges e and e^0 6= e such that $vert_G(e) = vert_G(e^0)$ are parallel . The edge-thickness of a graph G, denoted by $p(G)$, is the maximum cardinality of a set of pairwise parallel edges. We say that G is simple if it has no pair of parallel edges (i.e., $p(G) \cdot 1$).

A graph is the isomorphism class of a concrete graph. We call it an abstract graph to stress that it is de…ned "up to isomorphism". The reader will …nd the detailed (anyway obvious) de…nitions concerning isomorphisms in Chapter 2 of [6].

If G is a concrete graph, we let $Spl(G)$ be a simple subgraph of it obtained by iteratively removing one edge of any pair of parallel edges. Any two graphs obtained in this way are isomorphic, hence Spl is a (single-valued) mapping on abstract graphs.

De…nition 2 : Operations on graphs and graph algebras

Let A be a countable set of port labels containing the set N of nonnegative integers and the special symbol ?. Unless otherwise speci…ed, the de…nitions are the same for directed and undirected graphs. A concrete graph with ports, or a concrete p-graph in short, is a pair $G = (G^{\pm}, \text{port}_G)$ consisting of a concrete graph $G^{\pm}$ and a mapping $\text{port}_G : V_{G^{\pm}} \text{ ! } A$. A vertex x is an a-port of G if $\text{port}_G(x) = a$. The type $\pi(G)$ of G is the set $\text{port}_G(V_G)$ of port labels of its vertices. (We denote also $V_{G^{\pm}}$ by V_G, and similarly for other items).

If G and H are concrete p-graphs, we say that G is a subgraph of H if $G^{\pm}$ is a subgraph of $H^{\pm}$ and port_G is the restriction of port_H to V_G (so that $\pi(G) \mu \pi(H)$).

An (abstract) graph with ports, or an (abstract) p-graph is the isomorphism class of a concrete p-graph. In many cases, we will omit the distinction between concrete and abstract graphs because it is clear that, for proving properties of (abstract) graphs, we need to use concrete graphs.

Every (concrete or abstract) graph $G^{\pm}$ will be identi…ed with the (concrete or abstract) p-graph $(G^{\pm}, \text{port}_G)$ such that $\text{port}_G(x) = ?$ for every vertex x. Hence, we will use ? as a "default port label".

Our next objective is to de…ne operations on directed and undirected abstract p-graphs, hence, to equip these graphs with algebra structures.

Disjoint union. Two concrete graphs G and H are disjoint if $V_G \setminus V_H = ;$ and $E_G \setminus E_H = ;$, so that one can take their union in an obvious way. For disjoint p-graphs G and H, we let $G \copyright H$ be the union of $G^{\pm}$ and $H^{\pm}$ equipped with the port mapping $\text{port}_{G \copyright H} := \text{port}_G [\text{port}_H$. If G and H are not disjoint,

we replace one of them by an isomorphic copy disjoint from the other. In this way, we obtain a well-de…ned binary operation on abstract p-graphs. Clearly

$$\pi(G \copyright H) = \pi(G) \cup \pi(H).$$

Edge addition. Let $a, b \in A$, with $a \neq b$. For every concrete directed p-graph G, we let $\mathrm{add}_{a,b}(G)$ be a concrete p-graph G' such that $V_{G'} := V_G$, $E_{G'}$ is E_G to which we add one edge from x to y, for every $x, y \in V_G$ such that $\mathrm{port}_G(x) = a$ and $\mathrm{port}_G(y) = b$ (so that $\mathrm{vert}_{G'}(e) := \mathrm{vert}_G(e)$ if $e \in E_G$ and $\mathrm{vert}_{G'}(e) := (x, y)$ if e is such a new edge), and $\mathrm{port}_{G'} := \mathrm{port}_G$. For adding a loop, we use the operation add_a^{loop} that adds a loop at each vertex x such that $\mathrm{port}_G(x) = a$.

For adding undirected edges, we use the operation $\mathrm{add}_{a,b}$ de…ned similarly as $\mathrm{add}_{a,b}$. There is no di¤erence between a directed and an undirected loop, hence, the operation add_a^{loop} will also be used to add loops to undirected graphs. We have:

$$\pi(\mathrm{add}_{a,b}(G)) = \pi(\mathrm{add}_{a,b}(G)) = \pi(\mathrm{add}_a^{loop}(G)) = \pi(G).$$

Note that $\mathrm{add}_{a,b}(G) = G$ if a or b does not belong to $\pi(G)$, and similarly for $\mathrm{add}_{a,b}$, and for add_a^{loop} if $a \notin \pi(G)$.

These operations are well-de…ned on abstract p-graphs.

Port relabelling. Let $h : A \to A$ is a mapping that is the identity outside of a …nite subset of A. We de…ne relab_h as the unary operation such that $\mathrm{relab}_h(G)$ is the concrete p-graph G' such that $V_{G'} := V_G$, $E_{G'} := E_G$, $vert_{G'} := vert_G$ and $\mathrm{port}_{G'} := h \circ \mathrm{port}_G$. We have :

$$\pi(\mathrm{relab}_h(G)) = h(\pi(G)).$$

Clearly, $\mathrm{relab}_h \circ \mathrm{relab}_{h'} = \mathrm{relab}_{h \circ h'}$ for all mappings h and h'. A particular case deserves an easier notation: for $a, b \in A$, $a \neq b$, we let $\mathrm{relab}_{a \to b}$ denote relab_h where $h : A \to A$ is such that $h(a) = b$ and $h(c) = c$ for every $c \in A - \{a\}$. We have $\mathrm{relab}_{a \to b}(G) = G$ if $a \notin \pi(G)$. We can express a composition of relabellings $\mathrm{relab}_{a_1 \to b_1} \circ \mathrm{relab}_{a_2 \to b_2} \circ \cdots \circ \mathrm{relab}_{a_k \to b_k}$ as a single operation relab_h, and vice-versa.

If $C \subseteq A$ and $h : C \to A$ is the identity outside of a …nite subset of C (which holds in particular if C is …nite), we also denote by relab_h the operation $\mathrm{relab}_{h'}$ where h' agrees with h on C and is the identity outside of C. For each set $C \subseteq A$, we denote by $[C \to C]_f$ the set of mappings $h : C \to C$ such that h is the identity outside of a …nite subset of C.

Again, these operations are well-de…ned on abstract p-graphs.

Basic graphs. The constant symbol a will denote the abstract p-graph with a single vertex that is an a-port. The symbol $\emptyset$ will denote the empty graph. We have $\pi(a) = \{a\}$ and $\pi(\emptyset) = \emptyset$.

The two VR algebras of p-graphs. We obtain two countably in…nite sets of graph operations. Those of the …rst set act on directed p-graphs:

$$F^{\mathsf{VRd}} := \{\mathbf{1}, \overset{\rightarrow}{\mathrm{add}}_{a,b}, \mathrm{add}_a^{loop}, \mathrm{relab}_h, \mathbf{a}, ? \mid a, b \in A,\ a \neq b,\ h \in [A \to A]_f\}$$

and those of the second one on undirected p-graphs:

$$F^{\mathsf{VRu}} := \{\mathbf{1}, \mathrm{add}_{a,b}, \mathrm{add}_a^{loop}, \mathrm{relab}_h, \mathbf{a}, ? \mid a, b \in A,\ a \neq b,\ h \in [A \to A]_f\}.$$

We let GP^{d} denote the F^{VRd}-algebra with domain GP^{d} de…ned as the set of all (abstract) directed p-graphs, and we let GP^{u} be the corresponding F^{VRu}-algebra of undirected p-graphs with domain GP^{u}. We call them the VR algebras. (This terminology is motivated by the close relationship with the vertex-replacement graph grammars; see Chapter 4 of [6]).

We denote by $T(F)$ the set of terms over a set F of graph operations. We will identify a term t and its syntactic tree. Hence, we will discuss the occurrences of operation symbols in a term with the terminology of trees: nodes, leaves, root, ancestor etc. The ancestor relation is denoted by $\cdot_t$ ($u \cdot_t v$ if $u = v$ or v is a proper ancestor of u).

Each term t in $T(F^{\mathsf{VRd}})$ (resp. in $T(F^{\mathsf{VRu}})$) evaluates to a directed (resp. an undirected) p-graph that we denote by $\mathrm{val}(t)$. Its vertices are created by the constant symbols $\mathbf{a}$ and its edges by the operations $\overset{\rightarrow}{\mathrm{add}}_{a,b}$, $\mathrm{add}_{a,b}$ and add_a^{loop}. An occurrence in a term t of an edge addition operation is useful if it creates at least one edge. By a previous observation, this is equivalent to the condition that a and b (or a in the case of add_a^{loop}) belong to $\pi(val(t/u_1))$, where t/u_1 is the subterm of t issued from u_1, the son of u. An occurrence of such an operation that is not useful can be deleted: we obtain in this way a term that de…nes the same p-graph. Similarly, the constant symbol ? can be eliminated (except for de…ning the empty graph; this symbol is also useful for certain constructions of automata, see Section 6.3.4 of [6].)

We will also need, for each term t, a uniquely de…ned concrete p-graph $\mathrm{cval}(t)$ of which $\mathrm{val}(t)$ is the isomorphism class. We de…ne it as follows:

> its set of vertices is $Occ_0(t)$, the set of occurrences in t of the constant symbols $\mathbf{a}$ for $a \in A$, and
>
> its edges are the pairs $(u, (x, y))$ such that u is a useful occurrence of an operation $\overset{\rightarrow}{\mathrm{add}}_{a,b}$ that creates an edge from x to y, the pairs $(u, \{x, y\})$ such that u is a useful occurrence of $add_{a,b}$ that creates an undirected edge between x and y and the pairs $(u, \{x\})$ such that u is a useful occurrence of add_a^{loop} that creates a loop incident with x.

This concrete p-graph $\mathrm{cval}(t)$ is built from the occurrences of the symbols in t (i.e., from its nodes since we consider t as a tree). Its formal de…nition, by induction on the structure of t, is clear. There is a natural correspondence between the vertices of a concrete p-graph de…ned by t and the occurrences of

symbols a in this term, but this is not the case for edges: each occurrence u of an edge addition operation may create several edges. Hence, we distinguish these edges by the components (x, y), $\{x, y\}$ and $\{x\}$ in the above pairs $(u, (x, y))$, $(u, \{x, y\})$ and $(u, \{x\})$.

One more technical notion: if u is a node of t, we denote by $cval(t)/u$ the p-graph isomorphic to $cval(t/u)$ with vertex set $\{y \in Occ_0(t) \mid y \cdot_t u\}$, where t/u is the subterm of t issued from u. Note that if $u \not\leq w$ and $t/u = t/w$, then the concrete graphs $cval(t)/u$ and $cval(t)/w$ are isomorphic but not equal; they are actually disjoint because u and w are incomparable with respect to $\cdot_t$. For an example, consider the term

$$t = add_{b,c1}(add_{a,b2}(a_3 \oplus_4 b_5) \oplus_6 relab_{b \to c7}(add_{a,b8}(a_9 \oplus_{10} b_{11})))$$

where the subscripts 1 to 11 number the occurrences of its operation and constant symbols. This concrete p-graph is $3_a \bullet 5_b \bullet 11_c \bullet 9_a$ where the subscripts a, b, c indicate the port labels. Let $u := 2$ and $w := 8$. Then $t/u = t/w = add_{a,b}(a \oplus b)$. However, $cval(t)/u$ is the concrete graph $3_a \bullet 5_b$ and $cval(t)/w$ is $11_b \bullet 9_a$.

Two terms are equivalent if they evaluate into the same (abstract) p-graph.

The subsets F_C^{VRd} of F^{VRd} and F_C^{VRu} of F^{VRu} for $C \subseteq A$ are defined by restricting a, b to belong to C and h to belong to $[C \to C]_f$ in the above definitions. It is easy to show (by induction on t) that $\pi(val(t)) \subseteq C$ for every $t \in T(F_C^{VRd}) \cup T(F_C^{VRu})$.

The two VR algebras of simple p-graphs. We define also two algebras of simple p-graphs, denoted by GP^{sd} and GP^{su} (with the superscript s to distinguish them from GP^d and GP^u). The disjoint union and the relabellings transform simple graphs into simple graphs and the operations that add edges are defined as follows (as operations of GP^{sd} and GP^{su}):

$add^s_{a,b}(G) = Spl(add_{a,b}(G))$,
$add^s_{a,b}(G) = Spl(add_{a,b}(G))$,
$add^{s,loop}_a(G) = Spl(add^{loop}_a(G))$.

To take an example, the term $t = add_{a,b}(add_{a,b}(a \oplus b) \oplus b)$ evaluates in GP^{su} into the simple graph $b \bullet a \bullet b$ (with one a-port, two b-ports and two edges) and, in GP^u, into the graph $b = a \bullet b$ (with 3 edges, two of them being parallel).

Note that the operations of the algebras GP^{sd} and GP^d on the one hand and of GP^{su} and GP^u on the other are the same, but they are evaluated in different ways. We let $sval(t)$ be the simple graph that is the value in GP^{sd} or in GP^{su} of a term $t \in T(F_C^{VRd}) \cup T(F_C^{VRu})$. Clearly, $sval(t) = Spl(val(t))$.

The following facts are clear from the de…nitions:
$Spl(G \oplus H) = Spl(G) \oplus Spl(H)$,
$Spl(\mathrm{add}_{a,b}(G)) = Spl(\mathrm{add}_{a,b}(Spl(G)))$,
$Spl(\mathrm{add}_{a,b}(G)) = Spl(\mathrm{add}_{a,b}(Spl(G)))$,
$Spl(\mathrm{add}_{a}^{loop}(G)) = Spl(\mathrm{add}_{a}^{loop}(Spl(G)))$,
$Spl(\mathrm{relab}_{h}(G)) = \mathrm{relab}_{h}(Spl(G))$,
$Spl(a) = a$ and $Spl(?) = ?$.
They imply that the mapping Spl is a homomorphism of algebras: $GP^{d} \to GP^{sd}$ and: $GP^{u} \to GP^{su}$.

De…nition 3 : Clique-width.

The clique-width of a p-graph G is the minimal cardinality of a set of labels C such that G is the value of a term t in $T(F_C^{VRd}) \cup T(F_C^{VRu})$. This number is denoted by $cwd(G)$. Every p-graph G is the value of some term in $T(F^{VRd}) \cup T(F^{VRu})$ and $cwd(G) \cdot |V_G|$. A simple graph can be de…ned as $val(t)$ for some term $t \in T(F_C^{VRd}) \cup T(F_C^{VRu})$, but it can also be de…ned as $sval(t')$ for such a term t', that might use a smaller set of labels C. However, this is not the case:

Proposition 4 : If $G = sval(t)$ for some term $t \in T(F_C^{VRd}) \cup T(F_C^{VRu})$, then $G = val(t')$ for some term $t' \in T(F_C^{VRd}) \cup T(F_C^{VRu})$. For every graph G, we have $cwd(Spl(G)) \leq cwd(G)$, and the inequality may be strict.

Proof : We let $t \in T(F_C^{VRd})$, $G := sval(t)$ and $H := cval(t)$. If H is simple, we take $t' := t$. Otherwise H has two edges e and e' such that $vert_H(e) = vert_H(e')$. They are created by edge addition operations, $\mathrm{add}_{a,b}$ at an occurrence u and $\mathrm{add}_{c,d}$ at an occurrence v that is a proper ancestor of u in t; the pair (c,d) may di¤er from (a,b) because of possible relabellings on the path between u and v in this tree. We denote this fact by $u @ v$. Since e' is parallel to e, all edges created by $\mathrm{add}_{a,b}$ at u have parallel edges created by $\mathrm{add}_{c,d}$ at v. Hence, if we replace $\mathrm{add}_{a,b}$ at u by the identity (say by the unary operation relab_{Id}), we obtain a term t_1 such that $sval(t_1) = sval(t)$, and such that $val(t_1)$ has less edges than $val(t)$. By repeating …nitely many times this transformation that does not introduce new port labels we obtain a term $t' \in T(F_C^{VRd})$ such that $val(t') = sval(t)$. The same proof can be done for undirected graphs.

Since every term $t \in T(F_C^{VRd}) \cup T(F_C^{VRu})$ that evaluates into a p-graph G can be transformed into $t' \in T(F_C^{VRd}) \cup T(F_C^{VRu})$ that evaluates into $Spl(G)$, we have $cwd(Spl(G)) \leq cwd(G)$. Here is an example such that $cwd(Spl(G)) < cwd(G)$. We let $H := val(t)$ where $t := \mathrm{add}_{a,b}(\mathrm{add}_{a,b}(a \oplus a \oplus b \oplus b))$ and G be H minus one edge. We have $Spl(G) = Spl(H) = val(\mathrm{add}_{a,b}(a \oplus a \oplus b \oplus b))$, hence $cwd(Spl(G)) = 2$. It is clear that $G = val(s)$ where s is the term

$$\mathrm{add}_{a,b}(\mathrm{relab}_{c \to a}(\mathrm{add}_{c,b}(\mathrm{add}_{a,b}(a \oplus b) \oplus b \oplus c))),$$

and it is not hard to check that no term using only 2 labels can de…ne G. Hence, $cwd(G) = 3$. ¤

Clique-width has been de…ned in [10] and [9] for simple graphs only, as the minimal cardinality of a set C such that $G = sval(t)$ for some term $t \in T(F_C^{VRd}) \setminus T(F_C^{VRu})$. The …rst assertion of Proposition 4 shows that the new de…nition agrees for simple graphs with the usual one.

Another technical point is discussed in Section 2.5.6 of [6]: the clique-width of simple graphs can be de…ned by replacing in the sets F_C^{VRd} and F_C^{VRu} the operations $relab_h$ by the particular operations $relab_{a \to b}$ for $a, b \in C$ (as in the original de…nition of [10]). The resulting values of clique-width are the same with the two de…nitions. (This is not completely trivial because if $C = \{a, b\}$ and h exchanges a and b, then $relab_h$ is not a composition of the operations $relab_{a \to b}$ and $relab_{b \to a}$.) The proof given in [6] works as well for terms denoting graphs with multiple edges (as it does not concern the operations that add edges, but only the relabellings).

The notion of clique-width can also be de…ned for simple $(L, ¤)$-labelled graphs, i.e., for graphs such that every edge has a unique label from a …xed …nite set ¤ and every vertex has a possibly empty set of labels from a …xed …nite set L disjoint from A. For specifying labels, one uses the constant symbol a^B to de…ne an isolated a-port with set of labels B and the edge addition operations $add^{\lambda}_{a,b}$, $add^{\lambda}_{a,b}$ and $add^{loop,\lambda}_{a,b}$ that add edges labelled by $\lambda \in ¤$. We refer the reader to [6] for the detailed de…nitions. The extension of the above de…nitions to $(L, ¤)$-labelled graphs with multiple edges is straightforward.

The following proposition shows that adding randomly parallel edges to a given simple graph may increase its clique-width in an unbounded way.

Proposition 5 : There is no function f such that $cwd(G) \leq f(cwd(Spl(G)))$ for every graph G without any triple of parallel edges, hence a fortiori, for every graph G.

Proof : The proof will use the following claim:

Claim: Let K and H be two concrete simple undirected and loop-free graphs such that H is a subgraph of K and $V_H = V_K$; we de…ne $K + H$ as the graph obtained from K by adding a parallel edge to every edge of H. Then we have $cwd(H) \leq cwd(K + H)$. ¤

Let us illustrate the de…nition. Let K be the path: $x - y - z - u$ and let H be $x - y \quad z - u$. Then $K + H$ is the graph $x = y - z = u$ with two pairs of parallel edges between x and y, and z and u.

Proof of the claim: We let $K + H$ be de…ned by a term t in $T(F_C^{VRu})$ such that C has the minimal cardinality, i.e. $|C| = cwd(K + H)$. We can assume

that all occurrences of edge addition operations are useful in t (cf. De…nition 2).

If u is an occurrence of add$_{a,b}$ and v is an occurrence of add$_{c,d}$, we write u @ v (similarily as in the proof of Proposition 4) if and only if v is a proper ancestor of u and the relabellings on the path in t between u and v (composed bottom-up) transform fa,bg into fc,dg. The second condition is equivalent to the fact that the operation add$_{c,d}$ at v creates an edge parallel to some edge that has been created by add$_{a,b}$ at u. It implies that each edge created by add$_{a,b}$ at u gets a parallel edge created by add$_{c,d}$ at v. It is also clear that any two parallel edges of K + H are created by such operations, at some occurrences u and v such that u @ v. It follows from these observations that there are no 3 occurrences u, v, w of edge addition operations such that u @ v @ w, otherwise, we would have a triple of parallel edges. This is not possible by the de…nition of K + H.

We now transform t into t^0 2 $T(F_C^{\mathsf{VRu}})$ as follows:

> if u is an occurrence of add$_{a,b}$ in t such that there is no v with u @ v, then we replace add$_{a,b}$ by the identity operation at u.

We claim that H = $val(t^0)$. Consider an edge e of K + H without parallel edge: it is created by an operation add$_{a,b}$ at some occurrence u such that there is no v with u @ v, hence this operation is replaced in t^0 by the identity and this edge is not in $val(t^0)$; if e has a parallel edge e^0, then these two edges are created by edge addition operations at u and v such that u @ v; the operation at v is replaced by the identity but the operation at u remains by the de…nition of t^0, hence exactly one of the two edges remains in $val(t^0)$. This shows that H = $val(t^0)$. Hence, $cwd(H)$ · $cwd(K + H)$.¤

For proving the proposition, we consider K + H as in the above fact. Then $Spl(K + H)$ = K. Take for K a clique, and for H, any simple undirected and loop-free graph such that V_H = V_K. Hence, $cwd(Spl(K + H))$ = 2. If, for some …xed function f we would have $cwd(G)$ · $f(cwd(Spl(G)))$ for every graph G having no triple of parallel edges, then, by taking G := K + H and by the claim, we would have $cwd(H)$ · $cwd(K + H)$ · $f(2)$. But the simple undirected and loop-free graphs have unbounded clique-width ([10,17]), hence we get a contradiction.

The proof is easily adapted to directed graphs. ¤

Although the following notion is well-known we recall its de…nition at least for making notation precise.

De…nition 6 : Tree-decompositions.

A tree-decomposition of a graph G is a pair (T, f) such that T is a rooted and directed tree with set of nodes N_T and $f : N_T$ ¡ ! P(V_G) is a mapping such that:

1) Every vertex of G belongs to $f(u)$ for some u in N_T,

2) Every edge has its ends in $f(u)$ for some u in N_T,

3) For each vertex x, the set $f^{-1}(x) := \{u \in N_T \mid x \in f(u)\}$ is connected in T.

The width of a tree-decomposition (T, f) is the the maximal cardinality -1 of a box, i.e. of a set $f(u)$. A path-decomposition is defined as a tree-decomposition such that T is a directed path. The tree-width $twd(G)$ (the path-width $pwd(G)$) of a graph G is the minimal width of a tree-decomposition (a path-decomposition) of this graph.

It is known from [4] and [10] that a set of simple graphs, directed or not, that has bounded tree-width has bounded clique-width. This is not true for graphs with multiple edges.

For every graph G, we let $G - \alpha$ be the graph obtained by adding to G a universal vertex, i.e., a vertex α linked to all vertices of G (by undirected edges if G is undirected and by edges directed towards α if G is directed).

Proposition 7 : The set of undirected graphs of tree-width 2 has unbounded clique-width.

Proof : We use an auxiliary construction. Let G be a simple loop-free undirected graph, and let $\widehat{G}$ be obtained from $G - \alpha$ by the addition of parallel edges to all edges of $G - \alpha$, in such a way if $\{x, y\} \neq \{w, z\}$, $x -_{\widehat{G}} y$ and $w -_{\widehat{G}} z$ then, the number of edges between x and y and between w and z are different. Clearly, $twd(\widehat{G}) = twd(G) + 1$, since tree-width does not depend on the multiplicity of edges.

Claim: If $cwd(\widehat{G}) \leq k$, then $pwd(G) \leq k - 1$.

Proof of the claim: Let $t \in T(F_C^{VRu})$ be a term such that $cval(t) = \widehat{G}$ and $|C| = k$. Let x be a vertex of G, hence a leaf of t, and let u be a node above x. We denote by $port_t(x, u)$ the port label of x at u, defined as the port label of the vertex x in the concrete p-graph $cval(t)/u$, cf. Definition 2. (For the term t used as example in this definition, we have $port_t(11, u) = b$ for $u \in \{8, 10, 11\}$ and $port_t(11, u) = c$ for $u \in \{1, 2, 6, 7\}$.)

We denote by $lca(x, y)$ is the least common ancestor of two vertices (hence, two leaves) x and y. It is an occurrence of $\odot$. The vertices x and y are adjacent if and only if there exists an occurrence w of $add_{a,b}$ or $add_{b,a}$ such that $lca(x, y) \cdot_t w$, $port_t(x, w) = a$ and $port_t(y, w) = b$. We say that a vertex x is live at u if $x \cdot_t u$ and there is a vertex y adjacent to x such that $u \cdot_t lca(x, y)$.

We let P be the path in t linking the root to the leaf α. For each u on this path, we let $f(u)$ be the set of vertices of G that are live at u. We claim that (P, f) is a path-decomposition of G of width at most $k - 1$.

(a) Every vertex x is adjacent to α, hence it is live at $lca(x, \alpha)$ and belongs to the box $f(lca(x, \alpha))$.

(b) Let x and y be adjacent in G. If $lca(x,y) <_t lca(x,¤) = lca(y,¤)$, then x and y belong both to $f(lca(x,¤))$ by (a). If $lca(x,¤) <_t lca(x,y) = lca(y,¤)$, then x and y are live at $lca(x,y)$ hence they belong both to the box $f(lca(y,¤))$. If $lca(y,¤) <_t lca(x,y) = lca(x,¤)$ they belong both to the box $f(lca(x,¤))$.

(c) The connectivity condition holds because, if x is live at u, it is live at all nodes v on the path in t between x and u.

(d) Let x and y belong to a box $f(u)$. We have $lca(x,y) \cdot_t u$. The vertices x and y have di¤erent port labels at u: there is a vertex z adjacent to x such that $u \cdot_t lca(x,z)$. If x and y had the same port labels at u, both would be adjacent to z with the same numbers of parallel edges, but this is not possible by the construction of $\mathcal{G}$.

Hence, (P,f) is a path-decomposition of G whose boxes have at most k vertices.¤

To complete the proof of the proposition, take G to be a tree. Then $twd(\mathcal{G}) = 2$, but trees have unbounded path-width. Hence, the clique-widths of the graphs $\mathcal{G}$ are unbounded.¤

From this proposition, we obtain another proof of Proposition 5. Since tree-width does not depend on the multiplicity of edges, if we had a function f such that $cwd(G) \cdot f(cwd(Spl(G)))$, the graphs of tree-width 2 (with multiple edges) would have bounded clique-width because simple undirected graphs of tree-width 2 have clique-width at most 6 by [4].

De…nition 8: The parsing problem.

The parsing problem for clique-width consists in …nding an algorithm to do the following:

> Given a graph G and an integer k, to answer that G has clique-width more than k or to output a term witnessing that its clique-width is at most k.

This problem is NP-complete [13] but there exists an approximation algorithm, call it AP_{cwd} (by the results of [19] and [20]) that does the following in time $g(k).n^3$, where n is the number of vertices of the given graph, and f and g are …xed functions:

> Given a simple graph G and an integer k, either it answers (correctly) that G has clique-width more than k, or it outputs a term witnessing that it has clique-width at most $f(k)$.

This result su¢ces to prove that the model-checking problem for every monadic second-order property is …xed-parameter cubic with respect to clique-width as parameter ([9] and Chapter 6 of [6]). It extends to simple $(L,¤)$-labelled graphs because these graphs can be encoded into simple undirected

vertex-labelled graphs, and this encoding preserves the property that a set of graphs has bounded clique-width; the details are in Section 6.2 of [6]. From this result, we obtain as follows a …xed-parameter cubic algorithm for approximating the clique-width of graphs with multiple edges, where the functions f and g depend on k and on the edge-thickness $p(G)$ of the input graph G. We sketch the idea of this extension.

Let G be a graph and $p(G) \cdot p$. For every e 2 E_G, let $P(e)$ be the set consisting of e and the edges parallel to it. We distinguish these edges by labelling them by the integers $1, 2, ..., \mathrm{j}\ P(e)\ \mathrm{j} \cdot p$. In this way, we transform G into a simple $(;, [p])$-labelled graph $\mathbf{b}$. From any term that de…nes $\mathbf{b}$, we get one that de…nes G by deleting the edge labels from its edge addition operations. Hence, $cwd(G) \cdot cwd(\mathbf{b})$. A technical lemma gives the equality. It follows that the algorithm for approximating the clique-width of simple $(;, [p])$-labelled graphs can be used for approximating the clique-width of graphs of edge-thickness at most p. (However, this algorithm is actually not practically usable because it is based on complicated constructions that yield huge constants).

It is an open problem to …nd an approximation algorithm for the clique-width of graphs with multiple edges analogous to AP_{cwd} that would operate in time $g(k).n^c$, or even in time $n^{g(k)}$ where n is the number of vertices and edges of the input graph G and such that the …xed constant c and/or function g do not depend on $p(G)$.

3 Monadic second-order logic

De…nition 9 : CMS_1 and CMS_2 graph properties.

We assume that the reader knows the basics of monadic second-order logic (exposed in, e.g., in [9], [12], [14], [18] and Chapter 5 of [6]). We only review some perhaps not so well-known notions and the relevant notation.

If q , 2 and 0 $\cdot p < q$, the set predicate $Card_{p,q}(X)$ expresses that the cardinality of X is equal to p modulo q. We will use $Card_{p,q}(X)$ as an atomic formula where X is a set variable. Let r be a nonnegative integer: a $\mathrm{C}_r\mathrm{MS}$ formula is a monadic second-order formula that can be written with the set predicates $Card_{p,q}$ for $q \cdot r$. The CMS formulas are the same without any bound on q; the $\mathrm{C}_0\mathrm{MS}$ (that are also the $\mathrm{C}_1\mathrm{MS}$) formulas use no such set predicates and are the MS formulas. Counting monadic second-order logic refers to CMS formulas.

Graph properties can be expressed by monadic second-order formulas (or by formulas of any language) via two (main) representations of graphs by relational structures. The …rst representation associates with every graph G the logical structure $\mathrm{b}G\mathrm{c} := \mathrm{h}V_G, edg_G\mathrm{i}$ where edg_G the binary relation on vertices such that (x, y) 2 edg_G if and only if $vert_G(e) = \mathrm{f}x, y\mathrm{g}$ (possibly with $x = y$) or $vert_G(e) = (x, y)$ for some edge e of G.

A graph property $P(X_1, ..., X_n)$, where $X_1, ..., X_n$ denote sets of vertices, is a C_rMS_1 graph property (a CMS_1 graph property) if there exists a C_rMS formula (a CMS formula) $\varphi(X_1, ..., X_n)$ such that, for every graph G and for all sets of vertices $X_1, ..., X_n$ of this graph, we have:

$$\mathfrak{b}G\mathfrak{c} \models \varphi(X_1, ..., X_n) \text{ if and only if } P(X_1, ..., X_n) \text{ is true in } G.$$

Since for every graph G, we have $\mathfrak{b}G\mathfrak{c} = \mathfrak{b}Spl(G)\mathfrak{c}$, a CMS_1 graph property cannot depend on the multiplicity of edges. This is not due to monadic second-order logic but to the chosen representation of graphs. Incidence graphs can remedy this drawback. The incidence graph of an undirected graph G is the directed bipartite graph $Inc(G) := \mathfrak{h}V_G \mid E_G, in_G\mathfrak{i}$ where in_G is the set of pairs (e, x) such that $e \in E_G$ and x is an end vertex of e. (We use the simpler notation in_G instead of $edg_{Inc(G)}$). If G is directed, we let $Inc(G) := \mathfrak{h}V_G \mid E_G, in_{1G}, in_{2G}\mathfrak{i}$ where in_{1G} (resp. in_{2G}) is the set of pairs (e, x) such that $e \in E_G$ and x is the tail vertex of e (resp. its head vertex). Hence, $Inc(G)$ is directed and bipartite with two types of edges. We will also denote by $\mathfrak{d}G\mathfrak{e}$ the graph $Inc(G)$ considered as a relational structure.

A graph property $P(X_1, ..., X_n, Y_1, ..., Y_m)$, where $X_1, ..., X_n$ denote sets of vertices and $Y_1, ..., Y_m$ denote sets of edges, is a C_rMS_2 graph property (a CMS_2 graph property) if there exists a C_rMS formula (a CMS formula) $\varphi(X_1, ..., X_n, Y_1, ..., Y_m)$, such that, for every graph G, for all sets of vertices $X_1, ..., X_n$ and for all sets of edges $Y_1, ..., Y_m$ of this graph, we have:

$$\mathfrak{d}G\mathfrak{e} \models \varphi(X_1, ..., X_n, Y_1, ..., Y_m) \text{ if and only if}$$
$$P(X_1, ..., X_n, Y_1, ..., Y_m) \text{ is true in } G.$$

For example, the property Ham that an undirected graph has at least 3 vertices and a Hamiltonian cycle is an MS_2-property that is not CMS_1 (see [6], Chapter 5). Note that an undirected graph G satisfies Ham if and only if $Spl(G)$ satisfies Ham, so this fact has nothing to do with the representation of multiple edges.

We will introduce graph properties that can depend on the multiplicity of edges without being written with edge set quantifications. They will be intermediate between CMS_1 and CMS_2 properties, but they will not include Ham. The constructions of finite automata that yield fixed-parameter linear model-checking algorithms for input graphs given with the corresponding terms extend to them.

Definition 10 : $CMS_\sharp$ graph properties.

For every graph G, we denote by $\sharp edg_G$ the mapping that associates with every pair of vertices (x, y), the number of edges e of G such that $vert_G(e) = \{x, y\}$ (possibly with $x = y$) or $vert_G(e) = (x, y)$ (if G is directed). We define:

$\flat G\sharp := \langle V_G, \sharp edg_G \rangle$. This pair is not a relational structure because $\sharp edg_G$ is a function with values in the in…nite set of integers and not a relation, but we will use it as if it was. Two graphs G and H are isomorphic if and only if $\flat G\sharp$ and $\flat H\sharp$ are isomorphic (in the obvious sense).

The CMS formulas that specify CMS_1 graph properties are written with the binary relation symbol edg. We de…ne the $C_rMS_\sharp$-formulas as the monadic second-order formulas that can be written with the set predicates $Card_{p,q}$ for $p < q \leq r$ and the (new) binary relation symbols edg_q for $0 \leq q \leq r$ and $edg_{p,q}$ for $0 \leq p < q \leq r$ and $2 \leq q$. The new symbols will be interpreted in $\flat G\sharp$ as follows :

$(x, y) \in edg_{p,qG}$ if and only if $(x, y) \in edg_G$ and $\sharp edg_G(x, y) \equiv p \pmod q$, and

$(x, y) \in edg_{qG}$ if and only if $\sharp edg_G(x, y) = q$.

The notation $\flat G\sharp \models \varphi(X_1, ..., X_n)$ is thus meaningful if φ is a $C_rMS_\sharp$-formula and $X_1, ..., X_n$ denote sets of vertices. Note that $(x, y) \in edg_G$ if and only if $(x, y) \notin edg_{0G}$. Hence, every C_rMS-formula can be identi…ed with the $C_rMS_\sharp$-formula obtained from it by replacing every atomic formula $edg(x, y)$ by $\neg edg_0(x, y)$.

The purpose of the following proposition is to illustrate the expressive power of $CMS_\sharp$-formulas. For every graph G and sets of vertices X and Y of this graph, we let $Edg_G(X, Y)$ denote the set of edges from a vertex of X to a vertex of Y if G is directed and that link a vertex of X and a vertex of Y if G is undirected. This set includes the loops incident with a vertex in $X \setminus Y$. We denote by $\sharp Edg_G(X, Y)$ the cardinality of $Edg_G(X, Y)$.

Proposition 11 : For every p and q in $\mathbb{N}$, there exist $C_qMS_\sharp$-formulas expressing that $\sharp Edg_G(X, Y) = q$ and $\sharp Edg_G(X, Y) \equiv p \pmod q$ (where $q \neq 2$ and $0 \leq p < q$) for all sets of vertices X and Y of a directed graph G. Similar formulas exist for undirected graphs. For simple graphs, these constructions yield respectively MS-formulas and C_qMS-formula. □

Proof: (1) The formulas expressing that $\sharp Edg_G(X, Y) = q$ are easy but lengthy to write. For directed graphs, consider for example the property that $\sharp Edg_G(X, Y) = 2$. It is equivalent to the following:

either there is only one pair in $(X \times Y) \setminus edg_G$ and this pair is in edg_{2G},

or there are exactly two pairs in $(X \times Y) \setminus edg_G$ and each of them is in edg_{1G}.

These conditions can be expressed by a $C_0MS_\sharp$-formula. The construction for the general case is similar and need not use the set predicates $Card_{p,q}$.

(2) We now consider, for directed graphs G, the property $\sharp Edg_G(X,Y) \equiv p \pmod q$. Clearly, $\sharp Edg_G(X,Y) = \sum_{x \in X} \sharp Edg_G(\{x\},Y)$. We also have:

$$\sharp Edg_G(\{x\},Y) = \sum_{i \in \mathbb{N}} i.\,|\{y \in Y \mid (x,y) \in edg_{iG}\}|.$$

Let us compute this modulo q:

$$\sharp Edg_G(\{x\},Y) \equiv \sum_{0 \leq i < q} i.\,\mathrm{mod}_q(|\{y \in Y \mid (x,y) \in edg_{i,qG}\}|) \pmod q,$$

where for each integer s, $\mathrm{mod}_q(s)$ is the unique integer in $[0, q-1]$ that is equivalent to s modulo q. Hence, $\sharp Edg_G(\{x\},Y) \equiv s \pmod q$ if and only if the following formula $\theta_{s,q}(x,Y)$ with free variables x and Y is satisfied in $\mathbb{G}_\sharp$:

$$\bigvee_{(p_0,\ldots,p_{q-1}) \in A(s,q)} \bigwedge_{0 \leq i < q} \exists U.(Card_{p_i,q}(U) \wedge \forall u(u \in U \Leftrightarrow u \in Y \wedge edg_{i,q}(x,u))),$$

where $A(s,q)$ denotes the set of q-tuples $(p_0,\ldots,p_{q-1}) \in [0,q-1]^q$ such that $0.p_0 + 1.p_1 + \ldots + (q-1).p_{q-1} \equiv s \pmod q$.

By similar observations, we get that $\sharp Edg_G(X,Y) \equiv p \pmod q$ if and only if there exists a q-tuple $(p_0,\ldots,p_{q-1}) \in A(p,q)$ such that, for each $i = 0,\ldots,q-1$, we have $p_i = \mathrm{mod}_q(|\{x \in X \mid \sharp Edg_G(\{x\},Y) \equiv i \pmod q\}|)$. It follows that $\sharp Edg_G(X,Y) \equiv p \pmod q$ if and only if the following formula $\mu_{p,q}(X,Y)$ with free variables X and Y is satisfied in $\mathbb{G}_\sharp$:

$$\bigvee_{(p_0,\ldots,p_{q-1}) \in A(p,q)} \bigwedge_{0 \leq i < q} \exists U.(Card_{p_i,q}(U) \wedge \forall u(u \in U \Leftrightarrow u \in X \wedge \theta_{i,q}(u,Y))).$$

(3) The construction is the same for undirected graphs.

(4) For the particular case of simple directed graphs, we use in these constructions edg instead of $edg_{1,q}$ and of edg_1, and $\neq edg$ instead of $edg_{i,q}$ and edg_i for every $i \neq 1$. For undirected graphs, there is a slight difference. If G is simple and undirected, then

$$\sharp Edg_G(X,X) = |\{x \in X \mid (x,x) \in edg_G\}| + |\{(x,y) \in (X \times X) \setminus edg_G \mid x \neq y\}|/2,$$

whereas, if G is simple and directed, we have:

$$\sharp Edg_G(X,X) = |\{x \in X \mid (x,x) \in edg_G\}| + |\{(x,y) \in (X \times X) \setminus edg_G \mid x \neq y\}|.$$

If X and Y are disjoint, then $\sharp Edg_G(X,Y) = |(X \times Y) \setminus edg_G|$ for simple undirected graphs, as for simple directed ones. In all cases, $Edg_G(X,Y)$ is the disjoint union of $Edg_G(X,Y-X)$, $Edg_G(X-Y,X \setminus Y)$ and $Edg_G(X \setminus Y, X \setminus Y)$. The construction of the formula for simple undirected graphs is then routine from these observations and the technique used in the first part of the proof. □

.

4 Finite automata from monadic second-order formulas

De…nitions 12 : Assignments encoded in the terms that de…ne graphs

Let F be a …xed …nite subset of F^{VRd} or of F^{VRu}. For every graph property P, we let L_P be the set of terms in $T(F)$ that evaluate to a graph satisfying P. If P is a CMS_1-property, then L_P is regular, i.e., is de…nable by a …nite F-automaton. We will extend the proof given in Section 6.3.4 of [6] to the language $\mathsf{CMS}_\sharp$. This proof uses an induction on the structure of the sentences that de…ne the properties P. Hence, we need automata associated with formulas having free variables to handle inductively the case of sentences of the form $9X_1, ..., X_n.\varphi$. Hence, we generalize the previous de…nition.

Let $P(X_1, ..., X_n)$ be a property of sets of vertices $X_1, ..., X_n$ of the graphs denoted by terms in $T(F)$. We recall from De…nition 2 that a term t 2 $T(F)$ evaluates to a concrete p-graph $cval(t)$ whose vertex set is $Occ_0(t)$, the set of occurrences in t of the constant symbols di¤erent from ?. If G is another concrete graph de…ned by t, then it is isomorphic to $cval(t)$, and the veri…cations of monadic second-order properties that we can do on $cval(t)$ apply to G via this isomorphism.

For example, consider the term t = $\mathsf{add}_{a,b1}(\mathsf{add}_{a,b2}(\mathsf{a}_3 \copyright_4 \mathsf{b}_5) \copyright_6 \mathsf{b}_7)$ where the indices from 1 to 7 designate the occurrences in t of the operation and constant symbols. We have $Occ_0(t)$ = f3, 5, 7g and the graph $cval(t)$ is:

$$5_b = 3_a \text{ ¡ } 7_b.$$

(The port labels a and b are indicated here as subscripts and there are two edges between vertices 5 and 3).

Let us go back to the general case. We let $F^{(n)}$ be obtained from F by replacing each constant symbol a by the constant symbols (a, w) where w 2 f0, 1g^n. For 1 · m · n, we let $pr_m : F^{(n)}$ ¡ ! $F^{(n \text{ i } m)}$ be the mapping, usually called a projection, that transforms (a, w) into (a, w') where w' is obtained from w by removing the last m Booleans. A term t in $T(F^{(n)})$ de…nes two things: …rst, the graph $cval(pr_n(t))$, (hence, $pr_n(t)$ is obtained from t by removing all Boolean components of the constant symbols), and second, the n-tuple $(V_1, ..., V_n)$ such that V_i is the set of vertices of $cval(pr_n(t))$ that are occurrences of constant symbols (a, w) where the i-th component of w is 1. The tuple $(V_1, ..., V_n)$ is an assignment of sets of vertices of $cval(pr_n(t))$ to the set variables $X_1, ..., X_n$. We will write t as $pr_n(t)$ ¤ $(V_1, ..., V_n)$. Every term in $T(F^{(n)})$ is of this form.

Then, we de…ne $L_{P(X_1,...,X_n)}$ as the set of terms s ¤ $(V_1, ..., V_n)$ 2 $T(F^{(n)})$ (with s 2 $T(F)$) such that $P(V_1, ..., V_n)$ is true in $cval(s)$. If P is de…ned by a formula φ with free variables in f$X_1, ..., X_n$g, then we denote $L_{P(X_1,...,X_n)}$ by $L_{\varphi,(X_1,...,X_n)}$. (A formula φ does not determine the variables $X_1, ..., X_n$ in a unique may; furthermore, if, for example, φ is X_3 µ X_1, we may have to

consider $L_{\varphi,(X_1,X_2,X_3,X_4)}$ as well as $L_{\varphi,(X_1,X_2,X_3)}$; hence we specify $(X_1,...,X_n)$ as argument of $L_{\varphi,(X_1,...,X_n)}$). The relevant set F is …xed by the context.

Theorem 13: Let F be a …nite subset of F^{VRd} or of F^{VRu}. For every $CMS_\sharp$ graph property $P(X_1,...,X_n)$, the language $L_{P(X_1,...,X_n)}$ is regular and an F-automaton de…ning it can be constructed from a $CMS_\sharp$ formula that de…nes P.

Proof: The proof is a small extension of that given in Section 6.3 of [6] for CMS_1 graph properties and the evaluation mapping $sval$ from terms to simple graphs. Here, we consider $CMS_\sharp$ graph properties and the evaluation mapping val from terms to graphs that can have multiple edges.

We review the main steps of the proof.

First, monadic second-order formulas can be written without …rst-order variables and without universal quanti…cations. Furthermore, one can always assume that formulas are written with the "standard" set variables $X_1,...,X_n,...$ and that the variables X_i are used in such a way that, for any subformula of the form $\exists X_n.\theta$, the formula θ has its free variables in $\{X_1,...,X_n\}$.

The atomic formulas are of the forms $X_i \subseteq X_j$, $X_i = \emptyset$, $Sgl(X_i)$, $Card_{p,q}(X_i)$ and $edg(X_i,X_j)$, (as in [6]), and here, we also use $edg_q(X_i,X_j)$, $edg_{p,q}(X_i,X_j)$. Their meanings, if not already de…ned or not clear from the notation, are as follows for a graph G:

$Sgl(X_i)$ means that X_i is singleton,

$edg(X_i,X_j)$ means that X_i and X_j are singletons, respectively $\{x\}$ and $\{y\}$, and that $(x,y) \in edg_G$,

$edg_q(X_i,X_j)$ means the same with $(x,y) \in edg_{qG}$ and

$edg_{p,q}(X_i,X_j)$ means the same with $(x,y) \in edg_{p,qG}$.

Then, the main part of the proof is the construction of a F-automaton $A_{\varphi,(X_1,...,X_n)}$ that recognizes the language $L_{\varphi,(X_1,...,X_n)}$. (All automata will be …nite, complete and bottom-up deterministic unless otherwise speci…ed). The construction is by induction on the structure of φ:

1) If φ is $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ or $\neg\varphi_1$, then one constructs $A_{\varphi,(X_1,...,X_n)}$ from $A_{\varphi_1,(X_1,...,X_n)}$ and $A_{\varphi_2,(X_1,...,X_n)}$ by the classical constructions of automata for intersection, union and complementation with respect to $T(F^{(n)})$ of the associated languages (cf. [3]).

2) If φ is $\exists X_n.\theta$, then we have $L_{\varphi,(X_1,...,X_{n-1})} = pr_1(L_{\theta,(X_1,...,X_n)})$, (the mapping pr_1 replaces every symbol (a,wi), where i is 0 or 1, by (a,w), so that $pr_1(T(F^{(n)})) = T(F^{(n-1)})$). It is straightforward to obtain from the deterministic $F^{(n)}$-automaton $A_{\theta,(X_1,...,X_n)}$ that recognizes $L_{\theta,(X_1,...,X_n)}$, a nondeterministic $F^{(n-1)}$-automaton A that recognizes $pr_1(L_{\theta,(X_1,...,X_n)}) = L_{\varphi,(X_1,...,X_{n-1})}$. Since we have decided to construct deterministic automata (this is necessary for complementations), we determinize A, which gives $A_{\varphi,(X_1,...,X_{n-1})}$. This determinization step increases (from N to at most 2^N) the number of states of A.

3) It remains to construct automata for the atomic formulas. The constructions are in most cases straightforward from the definitions. For example, if φ is $Sgl(X_3)$, then the automaton $A_{\varphi,(X_1,\ldots,X_6)}$ has to accept the terms that contain one and only one occurrence of a constant symbol of the form (a, w) where the third component of w is 1 (here $w \in \{0,1\}^6$).

> Convention: Every state called $Error$ is a "sink state": it is not accepting and the recognized "error" "propagates", that is, every transition with $Error$ among the input states yields $Error$ as output state.

We will only construct the automata for the atomic formulas $edg_q(X_1, X_2)$ and $edg_{p,q}(X_1, X_2)$. We first construct the automaton $A := A_{edg_q(X_1,X_2)}$ for the set $F := F_C^{VRd}$ and $q \geq 1$. Its set of states is :

$$S := \{0, Error\} \cup \{1(a), 2(a), a(i), ab(i) \mid a, b \in C, i \in [0, q]\}$$

(Another notation of states must be used if $C = [k]$). The meanings of these states are described in Table 1. Each state s is characterized by a property P_s in the following sense: for every term $t \in T(F^{(2)})$ and (V_1, V_2) in $T(F^{(2)})$, we have:

$t \in (V_1, V_2) \in L(A, s)$ if and only if the graph $cval(t)$ satisfies $P_s(V_1, V_2)$.

The notation $t' \in L(A, s)$ means that the unique run of A on a term $t' \in T(F^{(2)})$ terminates with state s (at the root of the syntactic tree of t').

The number of states is thus $(k + 1)(kq + 2)$ where $k = |C|$. The transition rules are in Table 2. The missing transitions yield $Error$. Here is an example: $\odot[ab(0), a(2)] \to Error$. The accepting states are those of the form $a(q)$ or $ab(q)$ (possibly with $a = b$). The table specifies $O(k^4)$ transitions.

State s	Property P_s
0	$V_1 = V_2 = \emptyset$
$1(a)$	$V_1 = \{v\}, V_2 = \emptyset, \text{port}_{cval(t)}(v) = a$
$2(a)$	$V_1 = \emptyset, V_2 = \{v\}, \text{port}_{cval(t)}(v) = a$
$a(i)$	$V_1 = V_2 = \{v\}$, $\text{port}_{cval(t)}(v) = a$, $edg_{icval(t)}(v, v)$
$ab(i)$	$V_1 = \{v_1\}, V_2 = \{v_2\}$, $v_1 \neq v_2$, $port_{cval(t)}(v_1) = a$, $port_{cval(t)}(v_2) = b$ and $edg_{icval(t)}(v_1, v_2)$
Error	All other cases

Table 1: Meanings of the states of A.

The states $aa(0)$ could be identified with $Error$ because no run including such a state can reach an accepting state. (The state $aa(0)$ at a node u indicates that the two vertices of V_1 and V_2 have been found in the p-graph $H := cval(t)/u$, that they are not linked by an edge and that they are both a-ports in H. Hence, no sequence of operations applied to H can create an edge

Transition rules	Conditions
$? \to 0$ $(a, 00) \to 0$ $(a, 10) \to 1(a)$ $(a, 01) \to 2(a)$ $(a, 11) \to a(0)$	
$\text{relab}_h[0] \to 0$ $\text{relab}_h[i(a)] \to i(c)$ $\text{relab}_h[a(j)] \to c(j)$ $\text{relab}_h[ab(j)] \to cd(j)$	$c = h(a), d = h(b),$ $i \in [2], j \in [0, q]$
$\text{add}_a^{loop}[s] \to s$	$s \notin \{a(0), \dots, a(q)\}$
$\text{add}_a^{loop}[a(i)] \to a(i+1)$ $\text{add}_a^{loop}[a(q)] \to \text{Error}$	$i < q$
$\text{add}_{a,b}[s] \to s$	$s \notin \{ab(0), \dots, ab(q)\}$
$\text{add}_{a,b}[ab(i)] \to ab(i+1)$ $\text{add}_{a,b}[ab(q)] \to \text{Error}$	$i < q$
$\copyright[1(a), 2(b)] \to ab(0)$ $\copyright[2(b), 1(a)] \to ab(0)$ $\copyright[s, 0] \to s$ $\copyright[0, s] \to s$	(possibly $a = b$) $s \in S$

Table 2: The transition rules of A.

between them). This shows that the automaton A is not minimal. However, keeping states like $aa(0)$ yields a more uniform description of the transitions.

If, in this automaton, we replace the transitions $\text{add}_a^{loop}[a(q)] \to \text{Error}$ and $\text{add}_{a,b}[ab(q)] \to \text{Error}$ by $\text{add}_a^{loop}[a(q)] \to a(q)$ and $\text{add}_{a,b}[ab(q)] \to ab(q)$ respectively, then we obtain an automaton A_1 that recognizes the language $L_{P(X_1,X_2)}$ where $P(X_1, X_2)$ means that X_1 and X_2 are singletons $\{x\}$ and $\{y\}$ such that $(x, y) \in edg_{rG}$ for some $r \neq q$. Furthermore, if $0 < p < q$, the automata A and A_1 recognize the language $L_{edg_p(X_1,X_2)}$ if they are given $ab(p)$ and $a(p)$ as accepting states (for both of them).

We will not detail the construction for $edg_{p,q}(X_1, X_2)$ because it is fully similar. The set of states is in this case

$$S^0 := \{0, Error\} \cup \{1(a), 2(a), a(i), ab(i) \mid a, b \in C, i \in [0, q-1]\}.$$

The state $ab(i)$ is characterized by the property:

$$V_1 = \{v_1\}, V_2 = \{v_2\}, v_1 \neq v_2, \text{port}_{\text{cval}(t)}(v_1) = a,$$
$$port_{cval(t)}(v_2) = b, (v_1, v_2) \in edg_{i,q\text{cval}(t)},$$

(it implies $(v_1, v_2) \in edg_{\mathrm{cval}(t)}$ even if $i = 0$). The state $a(i)$ is characterized by the property:

$$V_1 = V_2 = \{v\},\ \mathrm{port}_{\mathrm{cval}(t)}(v) = a,\ (v,v) \in edg_{i,q\mathrm{cval}(t)},$$

(similarly, it implies $(v,v) \in edg_{\mathrm{cval}(t)}$ even if $i = 0$). The only transitions that differ from the previous case are

$$\mathrm{add}_{a,b}[ab(i)] \to ab(\mathrm{mod}_q(i+1)) \quad \text{and} \quad \mathrm{add}_a^{loop}[a(i)] \to a(\mathrm{mod}_q(i+1))$$

for all $i \in [0, q-1]$. This set of transitions guarantees that, if the state $ab(0)$ is reached, there is at least one edge from v_1 to v_2 (cf. the above description of the meaning of a state $ab(i)$), and similarly for $a(0)$.

It is straightforward to transform these automata into automata for the atomic formulas $edg_q(X_i, X_j)$ and $edg_{p,q}(X_i, X_j)$, and to adapt these constructions to undirected graphs.

Remark 14 : Automata for $\sharp Edg_G(X,Y) \equiv p \pmod q$: In Proposition 11, we have constructed a formula $\mu_{p,q}(X,Y)$ to express that sets of vertices X and Y of a graph G satisfy $\sharp Edg_G(X,Y) \equiv p \pmod q$. This formula uses the relations $edg_{i,q}$, hence, since we have automata for the atomic formulas $edg_{i,q}(X_j, X_k)$, we can combine them to build automata for the formulas $\mu_{p,q}(X,Y)$. However, there is a direct construction that we present as an example of what one can do by "avoiding logic", i.e., by not using the general construction. (This technique is used in other cases in [7, 8] and in Section 6.3.4 of [6]. For example the basic property $Path(X,Y)$ saying that X is a set of two vertices linked by a path whose vertices are all in Y is monadic second-order expressible by a formula of quantifier-height 5. A direct construction yields an $F_{[k]}^{\mathrm{VRu}}$-automaton with less than 2^{k^2+2} states. An $F_{[k]}^{\mathrm{VRu}}$-automaton of similar size can be constructed for the property $Clique(X)$ expressing that there is an edge between any two distinct vertices of X. These constructions make easier the construction of automata for more complex formulas in which $Path(X,Y)$ and $Clique(X)$ occur as subformulas.)

We fix q and the set of port labels C. The states of the automaton B_p equivalent to $A_{\mu_{p,q}(X_1,X_2)}$ are the 7-tuples $(A_1, f_1, A_2, f_2, A_3, f_3, j)$ such that $j \in [0, q-1]$, $A_1, A_2, A_3 \subseteq C$ and f_i is a mapping : $C \to [0, q-1]$ such that $f_i(a) = 0$ if $a \notin A_i$ for each i. We describe with the same notation as above the characteristic property of such a state:

$A_1 = port_{cval(t)}(V_1 - V_2)$, i.e., A_1 is the set of port labels of the vertices of $V_1 - V_2$,

$f_1(a) = \mathrm{mod}_q(|\, port_G^{-1}(a) \setminus (V_1 - V_2)\,|)$ for every $a \in C$,

$A_2 = port_{cval(t)}(V_2 - V_1)$ and $f_2(a) = \mathrm{mod}_q(|\, port_G^{-1}(a) \setminus (V_2 - V_1)\,|)$ for every $a \in C$,

$A_3 = port_{cval(t)}(V_1 \setminus V_2)$ and $f_3(a) = \mathrm{mod}_q(|\, port_G^{-1}(a) \setminus V_2 \setminus V_1 \,|)$ for every $a \in C$,

and finally, $j = \mathrm{mod}_q(\sharp Edg_{cval(t)}(V_1, V_2))$.

The accepting states will be those such that $j = p$. The number of states is $q.k^{3(q+1)}$ where $k = |\, C \,|$. The transitions are easy to define from the above specifications. Let us show some examples:

i) For a constant symbol, we have $(a, 11) \to (\emptyset, \emptyset, \{(a, 1)\}, 0)$ (we write a pair (A_i, f_i) as the set of pairs $(a, f_i(a))$ for a in A_i).

ii) For the disjoint union, we have the following general description:

$\oplus[(A_1, f_1, A_2, f_2, A_3, f_3, , j), (A_1^0, f_1^0, A_2^0, f_2^0, A_3^0, f_3^0, , j^0)] \to$
$(A_1 \cup A_1^0, f_1 + f_1^0, A_2 \cup A_2^0, f_2 + f_2^0, A_3 \cup A_3^0, f_3 + f_3^0, , j + j^0)$

where additions are modulo q, hence

$$(f_1 + f_1^0)(a) := \mathrm{mod}_q(f_1(a) + f_1^0(a)).$$

iii) For $relab_h$ the transition replaces everywhere a by $h(a)$ and updates the counts of vertices. For example, f_1 is replaced by f_1^0 such that $f_1^0(a)$ is the sum modulo q of the numbers $f_1(b)$ such that $h(b) = a$. The integer j that counts edges is not modified.

iv) We now consider the operation add_a^{loop} that adds loops to the a-ports. We must count the loops incident with vertices in $V_1 \setminus V_2$. Hence, the only component of a state $(A_1, f_1, A_2, f_2, A_3, f_3, j)$ that is modified is j that becomes $\mathrm{mod}_q(j + f_3(a))$.

v) Finally, we consider the operations $add_{a,b}$. The transition must updates the number of edges from V_1 to V_2 that are added by this operation. These edges are from $V_1 \cap V_2$ to $V_2 \setminus V_1$, from $V_1 \cap V_2$ to $V_1 \setminus V_2$ and from $V_1 \setminus V_2$ to $V_2 \setminus V_1$, and these cases are mutually exclusive. It follows that j becomes

$$\mathrm{mod}_q(j + f_1(a).f_2(b) + f_1(a).f_3(b) + f_3(a).f_2(b)).\square$$

All these constructions of automata are done for generic sets C. That is, if we replace C by another set in bijection with it by f, then the corresponding automata are obtained by replacing $a \in C$ by $f(a)$ everywhere in the states, in the transitions and the accepting states of the original ones. In particular, the numbers of states and transitions depend only the cardinality of the considered set C.

From these constructions and the remarks at the end of Definition 8, we obtain that the model-checking problem for $CMS_\sharp$ sentences is fixed-parameter

cubic with respect to the parameter $(cwd(G), p(G))$ where G is the input graph (and $p(G)$ is its edge-thickness, cf. De…nition 1). It would be desirable to eliminate the dependency on $p(G)$.

5 Special tree-width

We de…ne special tree-width by means of terms over the sets F^{VRd} and F^{VRu}. An equivalent de…nition in terms of tree-decompositions will be given later.

De…nition 15: Special VR-terms.

We recall that $\pi(G)$ denotes the set of port labels of a p-graph G; we also denote by $\pi_1(G)$ the subset of those that label a single vertex of G. If t 2 $T(F^{\mathsf{VRd}})$ [$T(F^{\mathsf{VRu}})$, then $\pi(t)$ denotes $\pi(val(t))$ and $\pi_1(t)$ denotes $\pi_1(val(t))$. A term t in $T(F^{\mathsf{VRd}})$ [$T(F^{\mathsf{VRu}})$ is a special VR-term if it satis…es the following conditions:

1) $\pi(t')$ ¡ $\pi_1(t')$ µ f?g for every subterm t' of t (we consider t as one of its subterms),

2) if t_1 © t_2 is a subterm of t, then $\pi(t_1)$ \ $\pi(t_2)$ µ f?g,

3) for every relabelling relab_h occurring in t, we have $h(?) = ?$,

4) for every operation $\mathrm{add}_{a,b}, \mathrm{add}_{a,b}, \mathrm{add}_a^{loop}$ that occurs in t, we have a 6= ? and b 6= ?,

5) the constant symbol ? has no occurrence in t.

If C is a …nite set of port labels, we denote by $SpT(F_C^{\mathsf{VRd}})$ and $SpT(F_C^{\mathsf{VRu}})$ the sets of special VR-terms in $T(F_C^{\mathsf{VRd}})$ and in $T(F_C^{\mathsf{VRu}})$ respectively. The special tree-width of a graph G, denoted by $sptwd(G)$, is the least integer k such that such that $G = val(t)$ for some term t in $SpT(F_C^{\mathsf{VRd}})$ [$SpT(F_C^{\mathsf{VRu}})$ such that j C ¡ f?g j= k + 1. Since we identify a graph with a p-graph whose vertices are labelled by ?, the set C must always contain ?, except if G is the empty graph. The comparison with tree-width will justify the "+1" in the de…nition. The special tree-width of an empty graph is ¡1, that of a graph consisting of loops and isolated vertices is 0. Since the sets $\pi(t)$ and $\pi_1(t)$ are computable inductively on the structure of a term t, the sets $SpT(F_C^{\mathsf{VRd}})$ and $SpT(F_C^{\mathsf{VRu}})$ are regular.

Example 16 : Trees

Trees have special tree-width 1. To prove this, we let C := f?, 1, 2g. An undirected tree with one distinguished node called its root, is made into a p-graph as follows: the root is labelled by 1, all other nodes by ?. Let T_1, T_2 be two such trees, de…ned by terms t_1, t_2 2 $SpT(F_C^{\mathsf{VRu}})$. Then, we let T := T_1 n T_2 be de…ned by the term

$t := \mathrm{relab}_{2\,i\,!\,?}\ (\mathrm{add}_{1,2}(t_1 \,©\, \mathrm{relab}_{1\,i\,!\,2}(t_2)))\ 2\ SpT(F_C^{\mathsf{VRu}})$.

This tree is built as the disjoint union of the trees T_1 and T_2 augmented with an undirected edge between their roots, and the root of T is de…ned as that of T_1. Every rooted and undirected tree is generated by n from the trees reduced to isolated roots, that are de…ned (up to isomorphism) by the constant symbol 1. Hence, every rooted and undirected tree is de…ned by a term in $SpT(F_C^{\mathsf{VRu}})$. One can forget the root by applying the operation $\mathrm{relab}_{1\,i\,!\,?}$.¤

We now consider tree-decompositions. A rooted and directed tree T is always directed from the root towards the leaves. For two nodes x and y, we let $x \cdot_T y$ if and only if y is on the directed path from the root to x.

De…nition 17 : Special tree-decompositions.
A tree-decomposition (T, f) of a graph G is special if it satis…es the following condition, in addition to the three conditions of De…nition 6:

4) For each vertex x, the set $f^{i1}(x)$ is a directed path in T.

Proposition 18 : The special tree-width of a graph is the minimal width of a special tree-decomposition of this graph. There are linear-time algorithms for converting a term t in $SpT(F_C^{\mathsf{VRd}})$ [$SpT(F_C^{\mathsf{VRu}})$ into a special tree-decomposition of width j C ¡ f?g j ¡1 of the graph $val(t)$ and vice-versa.

Proof: From terms to decompositions. We will de…ne for every term t in $SpT(F_C^{\mathsf{VRd}})$ [$SpT(F_C^{\mathsf{VRu}})$ a special tree-decomposition $S(t)$ of the graph $G := cval(t)$, the boxes of which have at most j C ¡ f?g j vertices. The proof is by induction on the structure of t.

For every t, we will de…ne $S(t)$ so that its root box consists of the vertices of G that are not ?-ports. By the de…nition of special terms, each element of C ¡ f?g labels at most one vertex, hence the root box has at most j C ¡ f?g j vertices.

If t = a, then $S(t)$ has a single (root) box consisting of the unique vertex of G.

If $t = f(t_1)$ where f is an operation that adds edges, then, we take $S(t) := S(t_1)$.

If $t = \mathrm{relab}_h(t_1)$ and $(T_1, f_1) := S(t_1)$, we add to T_1 a new node r, we link it to the root r_1 of T_1 and we let r be the root of the new tree T. We de…ne f as the extension of f_1 such that $f(r)$ is the set of vertices of $G := val(t)$ that are not ?-ports. By the de…nition of a special VR-term, we have $f(r)$ µ $f_1(r_1)$. We obtain a special tree-decomposition $S(t) := (T, f)$ of G. (If $h(a)$ 6= ? for every a 6= ?, then $f(r) = f_1(r_1)$ and we can take $S(t) := S(t_1)$.)

If $t = t_1 \odot t_2$, then we use $(T_1, f_1) := S(t_1)$ and $(T_2, f_2) := S(t_2)$ as follows. We take the union of T_1 and T_2 that we can assume disjoint, we add a new node r, we link it to the roots r_1 and r_2 of T_1 and T_2, we let r be the root of the new tree T. We de…ne f as the extension of f_1 and f_2 such that $f(r) := f_1(r_1) \cup f_2(r_2)$. Hence, $f(r)$ is the set of vertices of $G := val(t)$ that are not ?-ports and $S(t) := (T, f)$ is a special tree-decomposition of G with the required property.

Since each box of $S(t)$ is the root box of $S(t')$ for some subterm t' of t, we have a special tree-decomposition of $G := cval(t)$ of width at most $|C - \{?\}| - 1$.

From decompositions to terms. We now construct special VR-terms from special tree-decompositions. We need some notation and a claim. Let C be a …nite set of port labels that contains ?. Let (T, f) be a tree-decomposition of a graph G and $\gamma : V_G \to C - \{?\}$ be a mapping that is injective on each box. We call such a mapping a proper coloring of (T, f). It is also a proper vertex-coloring of G since every edge has its ends in a same box. For every node u of T, we let T/u be the rooted and directed subtree of T issued from u, with $N_{T/u} = \{w \in N_T \mid w \cdot_T u\}$. Its root is u.

We denote by $G(u)$ the p-graph $(G(u)^{\pm}, port_{G(u)})$ where $G(u)^{\pm}$ is the induced subgraph of G with vertex set $\bigcup\{f(w) \mid w \in N_{T/u}\}$ and $port_{G(u)}(x) := \gamma(x)$ if $x \in f(u)$ and $port_{G(u)}(x) := ?$ if $x \in V_{G(u)} - f(u)$. Hence, $G(u)$ is a p-graph such that $\pi(G(u)) - \pi_1(G(u)) \subseteq \{?\}$. We have $G = G(root_T)^{\pm}$.

Claim: Let (T, f) be a tree-decomposition of width at most $k - 1$ of a graph G and let $C \supseteq A$ be a set of cardinality $k + 1$ that contains ?. There exists a proper coloring $\gamma : V_G \to C - \{?\}$ of (T, f). Such a coloring can be determined in time $O(|N_T|)$. ¤

Proof of the claim: Let G, C, T, f be as in the statement and $\delta_0 : f(root_T) \to C - \{?\}$ be any injective mapping. We will prove that the following holds for every $u \in N_T$:

> Every injective mapping $\delta : f(u) \to C - \{?\}$ can be extended into a mapping $\gamma : V_{G(u)} \to C - \{?\}$ that is injective on $f(w)$ for each w in $N_{T/u}$.

The proof is by bottom-up induction on u. If u is a leaf of T there is nothing to prove. Otherwise, let $u_1, \ldots, u_p$ be the sons of u. For each of them one can …nd an injective mapping $\delta_i : f(u_i) \to C - \{?\}$ that coincides with δ on $f(u_i) \cap f(u)$. By the induction hypothesis, it can be extended into γ_i de…ned on $V_{G(u_i)}$.

Then, the common extension γ of these mappings γ_i and of the mapping δ is the desired coloring. This extension exists because if $x \in N_{T/u_i} \cap N_{T/u_j}$, $i \neq j$, then $x \in f(u_i) \cap f(u) \cap f(u_j)$ by the connectivity condition (Condition 3) of De…nition 6), and so $\gamma_i(x) = \gamma_j(x) = \delta(x)$.

It is routine work to construct a linear algorithm computing γ.¤

Let (T, f) be a special tree-decomposition of a graph G of width at most k ¡ 1 and γ be as in the claim. (We need not distinguish the cases of directed and undirected graphs). We will construct terms $t(u)$ that de…ne the p-graphs $G(u)$ (their port labels depend on γ) so that: G = relab$_{C\ ¡\ !\ ?}(G(\text{root}_T))$ (where for every subset B of C, we let relab$_{B\ ¡\ !\ ?}$ denote the composition, in any order, of the operations relab$_{b\ ¡\ !\ ?}$ for all b 2 B ¡ f?g).

Let u have sons $u_1, \ldots, u_p$, p ¸ 0. We can assume that we have already constructed the terms $t(u_1), \ldots, t(u_p)$. We have:

$G(u)$ = ADD(relab$_{B_1\ ¡\ !\ ?}(G(u_1))$ © ¢¢¢© relab$_{B_p\ ¡\ !\ ?}(G(u_p))$ © a$_1$ © ... © a$_s$)

where f$a_1, ..., a_s$g := $\pi(G(u))$ ¡ (f?g [$\pi(G(u_1))$ [... [$\pi(G(u_p))$, B_i := f$\gamma(x)$ j x 2 $f(u_i)$ ¡ $f(u)$g for each i = 1,...p, and ADD is the composition of the edge addition operations that create the edges (and loops) of $G(u)$ that are not in the graphs $G(u_1), ..., G(u_p)$. Note that, since (T, f) is a special tree-decomposition, the sets π_1(relab$_{B_i\ ¡\ !\ ?}(G(u_i))$) are pairwise disjoint. Hence, we can de…ne:

$t(u)$:= ADD(relab$_{B_1\ ¡\ !\ ?}(t(u_1))$ © ¢¢¢© relab$_{B_p\ ¡\ !\ ?}(t(u_p))$ © a$_1$ © ... © a$_s$),

$t(u)$ belongs to $SpT(F_C^{\mathrm{VRd}})$ [$SpT(F_C^{\mathrm{VRu}})$ and de…nes $G(u)$.

The term relab$_{C\ ¡\ !\ ?}(t(\text{root}_T))$ de…nes G.

This construction can be done by a linear time algorithm, where the size of the input is jV_G [E_G [N_T j.¤

Proposition 19 : For every graph G we have:

(1) $twd(G)$ · $sptwd(G)$ · $pwd(G)$.

(2) $cwd(G)$ · $sptwd(G)$ + 2.

Proof: The …rst assertion follows from Proposition 18, and the second one from De…nition 15.¤

We will denote by $STWD(\cdot\ k)$ the class of directed and undirected graphs of special tree-width at most k. Smoothing a vertex of degree 2 means contracting any one of its two incident edges. (This de…nition excludes the case of a vertex incident with a loop and with no other edge).

Proposition 20 : For each k, the class $STWD(\cdot\ k)$ is closed under the following transformations:

1) Removal of vertices and edges,

2) Reversals of edge directions,

3) Addition and removal of loops incident with existing vertices,

4) Addition of edges parallel to existing edges,

5) Smoothing vertices of degree 2.

Proof: The closure is clear for the transformations of types 1)-4) because every special tree-decomposition of a graph is also a special tree-decomposition of any graph transformed in these ways.

We now consider the smoothing of a vertex y of degree 2 with neighbour vertices x and z in a graph G. Let (T, f) be a special tree-decomposition of G. The intersection of the directed paths $f^{-1}(x)$ and $f^{-1}(y)$ is not empty and is a directed path, and we let u be its minimal element with respect to $\cdot_T$. We let similarly v be the minimal element of $f^{-1}(z) \setminus f^{-1}(y)$. Since u and v are on the directed path $f^{-1}(y)$, they are comparable, say, $u \cdot_T v$. Let us now contract the edge between x and y, that is, we delete y and make x adjacent to z. This gives a graph G^0 such that $V_{G^0} = V_G - \{y\}$. Then, we define f^0 by:

$f^0(w) := (f(w) - \{y\}) \cup \{x\}$ for every w on the path in T from v to u (including u and v),

$f^0(w) := f(w) - \{y\}$ otherwise.

It is easy to see that (T, f^0) is a special tree-decomposition of G. In particular, x and z belong both to $f^0(v)$. The set $f^{0-1}(x)$ is the union of the directed path $f^{-1}(x)$ and of the path from v to u, hence it is a directed path since u belongs to $f^{-1}(x)$. Hence, (T, f^0) is a special tree-decomposition of G and its width that is no larger than that of (T, f), which shows that $sptwd(G^0) \cdot sptwd(G)$.¤

It follows from items 1) and 5) of this proposition that the class $STWD(\cdot k)$ is closed under taking topological minors ([11]). It is not closed under taking minors as we will see in Proposition 25 below. Topological minor inclusion is not a well-quasi order on $STWD(\cdot k)$ as one might hope. It is not on the simple undirected graphs in $STWD(\cdot 3)$: for each $n \geq 3$, take the undirected cycle C_n (with n vertices) and replace each of its edges by a Wheatstone bridge. One obtains an infinite set of graphs of special tree-width (and path-width) 3 that are pairwise incomparable for topological minor inclusion.

In the following proposition, $pwd(L)$ denotes the least upper bound of the path-widths of the graphs in L and similarly for the other notions of width.

Proposition 21 : The class of graphs of tree-width 2 has unbounded special tree-width. For every set of graphs L:

$$pwd(L) < 1 \Rightarrow sptwd(L) < 1 \Rightarrow twd(L) < 1 \text{ and}$$

$$sptwd(L) < 1 \Rightarrow cwd(L) < 1,$$

whereas the converse implications do not hold.¤

Proof : We will use the following claim.

Claim : For every graph G, the special tree-width of G - ¤ is equal to its path-width.

Proof of the claim: Let (T, f) be a special tree-decomposition of G - ¤ of width k. We let P be the directed path $f^{-1}(¤)$. We claim that $(P, f^{-1} P)$ is a path-decomposition of G - ¤.

For each vertex x of G, the directed paths $P = f^{-1}(¤)$ and $f^{-1}(x)$ have a nonempty intersection (because ¤ and x are adjacent), hence $x \in f(u)$ for some u in P.

If y is another vertex of G that is adjacent to x, then $P \setminus f^{-1}(y)$ is not empty and contains some node v. Let u and v be the $\cdot_T$-minimal nodes in $P \setminus f^{-1}(x)$ and in $P \setminus f^{-1}(y)$ respectively. If $u = v$, then the edges between x and y have their ends in $f(u)$. Otherwise, let us assume that $u <_T v$. Since $f^{-1}(x) \setminus f^{-1}(y)$ is not empty, it must contain v, and the edges between x and y have their ends in $f(v)$. The pair $(P, f^{-1} P)$ satis…es also the connectivity condition, hence it is a path-decomposition of G - ¤ of no larger width than (T, f). Since we have $sptwd(G$ - ¤$) \cdot pwd(G$ - ¤$)$ by Proposition 19, we have an equality.¤

For proving the proposition by contradiction, we assume that every graph of tree-width 2 has special tree-width at most k. If T is any tree, then T - ¤ has tree-width at most 2, hence special tree-width at most k, and path-width at most k by the claim. It follows that T, since it is a subgraph of T - ¤, has path-width at most k, but trees have unbounded path-width (see [11]), which gives a contradiction.

The implications follow from Proposition 19. Trees have special tree-width at most 1 (Example 16) and unbounded path-width. Graphs of tree-width 2 have unbounded special tree-width, hence the opposite implications are false. The converse of $sptwd(L) < 1 \Rightarrow cwd(L) < 1$ is false if L the set of cliques, of maximal clique-width 2 and of unbounded tree-width and special tree-width. ¤

De…nition 22: Tree-partitions.

A tree-partition of a graph G is a pair (T, f) such that T is a rooted tree with set of nodes N_T and $f : N_T \to \mathcal{P}(V_G)$ is a mapping such that:

1) Every vertex of G belongs to $f(u)$ for a unique node u of T,

2) Every edge has its two ends in $f(u) \cup f(v)$ for some nodes u, v of T such that v is the father of u.

The width of (T, f) is de…ned as the maximal cardinality of a box, (no -1 here !), and the tree-partition-width of a graph G (called strong tree-width in [21]) is the minimal width of its tree-partitions. We denote it by $tpwd(G)$. The wheels, i.e., the graphs C_n - ¤ where C_n is the undirected cycle with n vertices

have path-width (and special tree-width) 3 but unbounded tree-partition width (see [1], [24]). $MaxDeg(G)$ denotes the maximum degree of a graph G.

Proposition 23: For every graph G :

1) $sptwd(G) \cdot 2.tpwd(G)$ ¡ 1,

2) $sptwd(G) \cdot 20(twd(G) + 1)MaxDeg(G)$.

A set of graphs of bounded degree has bounded special-tree-width if and only if it has bounded tree-width.¤

By Proposition 20, we have even:

$$sptwd(G) \cdot 20(twd(G) + 1)MaxDeg(Core(G))$$

where $Core(G)$ is the simple, loop-free and undirected graph obtained from G by forgetting edge directions, removing loops and fusing parallel edges (independently of their original directions).

Proof: 1) Let (T, f) be a tree-partition of G of width k. We will transform it into a special tree-decomposition (T^0, f^0) of G such that $N_T = N_{T^0}$ and $f(u)$ µ $f^0(u)$ for every u 2 N_T. We choose an arbitrary linear order $\cdot$ on N_T and we let T^0 be the binary tree associated with T in the following classical way:

> if u is a node with sons $u_1, u_2, ..., u_p$, such that $u_1 < u_2 < ... < u_p$, then we let u_1 be the left son of u in T^0 and, for each i = 1, ..., p ¡ 1, we let u_{i+1} be the right son of u_i.

There are no other edges, hence T^0 is a tree with root $root_T$. The root has no right son. For every u 2 $N_T = N_{T^0}$, we de…ne:

$f^0(u) := f(u)$ if $u = root_T$,

$f^0(u) := f(u)$ [$f(w)$ if w is the father of u in T.

It is straightforward to verify that (T^0, f^0) is a special tree-decomposition of G. Its boxes have at most $2k$ vertices, hence G has special tree-width at most $2k$ ¡ 1. Figure 1 shows a tree-partition (to the left, the letters A, B, C, ... represent pairwise disjoint sets of vertices), and, to the right, the corresponding special tree-decomposition. (The box of the node XY is X [Y). Unless T has $root_T$ as single node, it can be delete from T^0. Hence our construction does not add new nodes to the given tree T.

2) For every graph G of tree-width and of maximal degree at least 1, we have $tpwd(G) \cdot 5(twd(G) + 1)(7.MaxDeg(G)/2 ¡ 1)/2$ by [24]. For these graphs, we get $sptwd(G) \cdot 20(twd(G) + 1)MaxDeg(G)$ by the …rst assertion. This inequality is actually valid if G is empty or has only loops and isolated vertices. ¤

This result suggests a question:

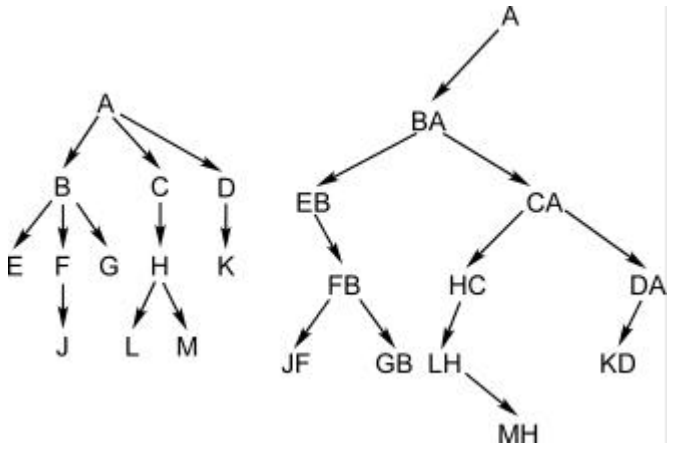

Figure 1: A tree-partition and the associated tree-decomposition

Which conditions on a set of graphs, other than bounded degree, imply that it has bounded tree-width if and only if it has bounded special tree-width?

Planarity does not since the graphs of tree-width at most 2 are planar but of unbounded special tree-width. From this case, we can see that conditions like excluding a …xed graph as minor or being uniformly k-sparse for some k do not either. All these conditions however, imply that, for simple graphs, bounded tree-width is equivalent to bounded clique-width (see [6], Chapter 9). Since bounded degree for a class of graphs is equivalent to excluding a star as a subgraph, one might also try to …nd such conditions expressed in terms of excluded subgraphs.

Proposition 24 : Every graph of tree-width k is obtained by edge contractions from a graph of special tree-width at most $2k + 1$. The class of graphs of special tree-width at most k is not closed under taking minors for any $k \geq 5$.

Proof: Every graph of tree-width k is obtained by edge contractions from a graph of tree-partition-width at most $k + 1$ (easy to check). The …rst assertion follows then from Proposition 23. The graphs of tree-width 2 are thus minors of graphs of special tree-width at most 5. If for some $k \geq 5$ the class $SPTWD(\cdot k)$ would be closed under taking minors, then all graphs of tree-width 2 would have special tree-width at most k. We know that this is not the case. ¤

Connected and biconnected components.

Proposition 25 : The special tree-width of a graph is the maximal special tree-width of its connected components. It is at most one plus the maximal special tree-width of its biconnected components. This upper bound is tight.

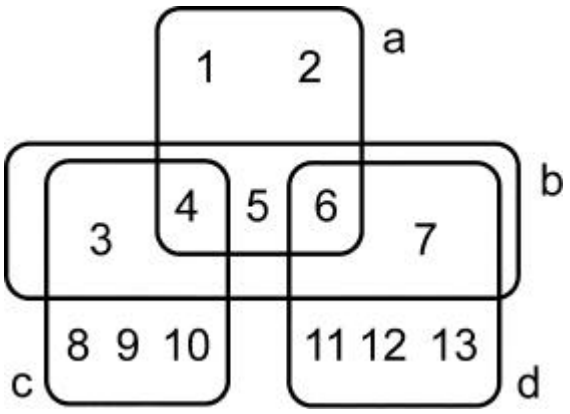

Figure 2: The special tree-decomposition (T, f) of H.

Proof: The …rst two assertions are easily proved by routine constructions that we omit. In order to prove the last one, we de…ne a graph G whose special tree-width is strictly larger than those of its biconnected components.

We let T be the rooted and directed tree with set of nodes fa, b, c, dg and edges a ¡ ! b, b ¡ ! c and b ¡ ! d. We let V_H := [13] and f be the mapping: N_T ¡ ! P(V_H) such that $f(a)$:= f1, 2, 4, 5, 6g, $f(b)$:= f3, 4, 5, 6, 7g, $f(c)$:= f3, 4, 8, 9, 10g and $f(d)$:= f6, 7, 11, 12, 13g.We de…ne H as the simple undirected graph that is the union of the cliques with vertex sets $f(a)$, $f(b)$, $f(c)$ and $f(d)$. It is clear that H is a chordal graph with 4 maximal cliques of size 5. It has tree-width 4 and also special tree-width 4: the pair (T, f) is a special tree-decomposition of H (see Figure 2).

Every clique in a graph is contained in some box of any tree-decomposition of this graph. It follows that any special tree-decomposition (T_1, f_1) of H of minimal width must have four nodes a_1, b_1, c_1, d_1 such that $f_1(a_1) = f(a)$, $f_1(b_1) = f(b), f_1(c_1) = f(c)$ and $f_1(d_1) = f(d)$. The tree T_1 cannot have a directed path containing b_1, d_1, a_1 in this order because this would imply that vertex 3 belongs to $f_1(d_1)$ by the connectivity condition. By similar arguments, we can see that T_1 must have directed paths containing a_1, b_1, c_1 and a_1, b_1, d_1 in this order and no directed path containing b_1, c_1 and d_1 (in any order). Roughly speaking, (T, f) is the only special tree-decomposition of H of width 4. This fact is a key point for our construction.

We let H^0 be the isomorphic copy of H where each vertex i is made into i^0 and (T^0, f^0) be the corresponding "isomorphic" special tree-decomposition of H^0. We construct G from the union of H and H^0 by fusing vertices 7 and 7^0 (that is, we delete 7^0 and we connect 7 with the neighbours of 7^0 in H^0). The biconnected components of G are H and H^0 hence, G has tree-width 4. It has special tree-width at most 5: Figure 3 shows a special tree-decomposition (T^{00}, f^{00}) of G. Its box $f^{00}(a^0)$ contains vertex 7 hence has 6 elements.

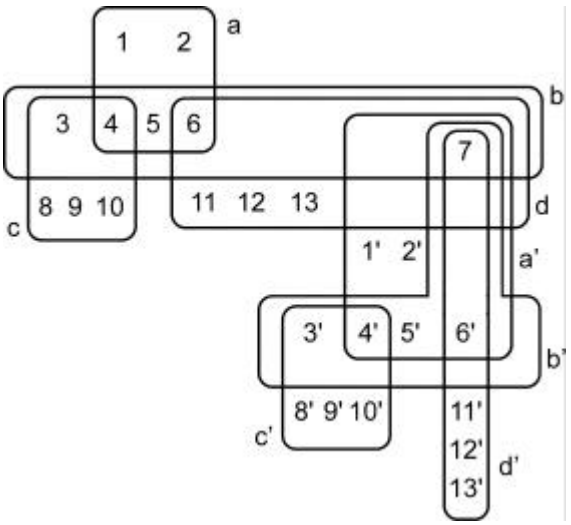

Figure 3: The special tree-decomposition (T^{00}, f^{00}) of G.

Assume that G has a special tree-decomposition (T_2, f_2) of width 4. It must have nodes $b_2, d_2, a_2^0, b_2^0, d_2^0$ such that $f_2(b_2) = f(b), f_2(d_2) = f(d)$, $f_2(a_2^0) = f^0(a^0)$ = f1$^0, 2^0, 4^0, 5^0, 6^0$g, $f_2(b_2^0)$ = f3$^0, 4^0, 5^0, 6^0, 7$g and $f_2(d_2)$:= f6$^0, 7, 11^0, 12^0$, 13^{0}g. Since (T_2, f_2) is a special tree-decomposition and by the connectivity condition, T_2 must have a directed path containing b_2, d_2, b_2^0 and d_2^0. By the observation made above for H (which applies also to H^0), we can see that we must have b_2 before d_2 and b_2^0 before d_2^0. But then we must also have a_2^0 on this path. We cannot do that without having 7 2 $f_2(a_2^0)$. Hence, (T_2, f_2) cannot exist and G has special tree-width 5.¤

Open question: The parsing problem.

Does there exist …xed functions f and g and an approximation algorithm to do the following in time $O(n^{g(k)})$, where n is the number of vertices of the given graph :

> Given a simple graph G and an integer k, either it answers (correctly) that G has special tree-width more than k, or it outputs special VR-term witnessing that its special tree-width is at most $f(k)$?

Stronger requirements would be that $f(k) = k$, giving an exact algorithm and/or the computation time $O(g(k).n^c)$ for some …xed c instead of $O(n^{g(k)})$. Since by a result by Bodlaender (presented in detail in [12]) such an algorithm

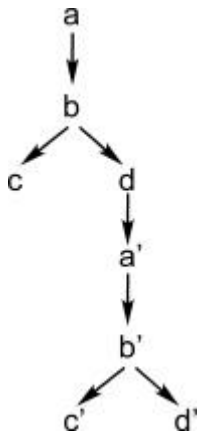

Figure 4: The tree T''

exists for tree-width, with $f(k) = k$ and $c = 1$, one can think that this algorithm can be adapted to make it construct special tree-decompositions.

A directed path graph ([16], also called a rooted path graph in [2]) is the intersection graph of a set of directed paths in a rooted and directed tree. It follows from Proposition 18 that a simple, loop-free and undirected graph has special tree-width at most k if and only if it is a subgraph of a directed path graph with maximum clique size k. This characterization is similar to a classical characterization of graphs of tree-width at most k. Directed path graphs can be recognized in polynomial time ([16]) and [2] investigates their possible characterization by forbidden induced subgraphs. However, none of these works gives a parsing algorithm or a characterization of $SPTWD(\cdot\ k)$ by forbidden con…gurations.

6 Finite automata for monadic second-order formulas with edge set quanti…cations

Our objective is to adapt the constructions of Section 4 to the model-checking of CMS_2 graph properties for graphs de…ned by special VR-terms. We will obtain …xed-parameter linear algorithms for graphs of bounded special tree-width given by the relevant terms or decompositions.

De…nition 26 : CMS_2 formulas and the encoding of assignments

In order to use CMS_2-formulas, i.e., monadic second-order formulas with edge set quanti…cations (and set cardinality predicates), we will represent a

graph G by the relational structure $\mathsf{d}G\mathsf{e} := Inc(G)$ of De…nition 9. If G is undirected, then $\mathsf{d}G\mathsf{e} := \mathsf{h}V_G$ [$E_G, in_G\mathsf{i}$ where in_G is the set of pairs (e, x) such that e 2 E_G and x is an end vertex of e. If G is directed, $\mathsf{d}G\mathsf{e} := \mathsf{h}V_G$ [$E_G, in_{1G}, in_{2G}\mathsf{i}$ where in_{1G} (resp. in_{2G}) is the set of pairs (e, x) such that e 2 E_G and x is the tail vertex of e (resp. its head vertex).

As in the proof of Theorem 13, we will use formulas with a particular "normalized" syntax. They will be written without …rst-order variables and universal quanti…cations, with the "standard" set variables $X_1, ..., X_n, ...$ for denoting sets of vertices and $Y_1, ..., Y_m, ...$ for denoting sets of edges. In any subformula $9X_n.\theta$, the formula θ has no free variables in $\mathsf{f}X_{n+1}, ...\mathsf{g}$, and similarly for $9Y_m.\theta$. The atomic formulas are of the forms $edg(X_i, X_j)$, $in(Y_i, X_j)$ (for undirected graphs), $in_1(Y_i, X_j)$ and $in_2(Y_i, X_j)$ (for directed graphs), and of course, X_i µ X_j, Y_i µ Y_j, Z = ;, $Sgl(Z)$, $Card_{p,q}(Z)$ where Z is X_i or Y_j. Their meanings, if not already de…ned are as follows for a graph G:

> $in(Y_i, X_j)$ means that Y_i and X_j are singletons, respectively $\mathsf{f}y\mathsf{g}$ and $\mathsf{f}x\mathsf{g}$, and that (y, x) 2 in_G,
>
> and similarly for $in_1(Y_i, X_j)$ and $in_2(Y_i, X_j)$.

We now discuss the encoding of assignments in terms. Let t be a special VR-term and G be the concrete graph $cval(t)$. Its vertices are the elements of $Occ_0(t)$ (they are leaves of t). Its edges are pairs $(u, (x, y))$, $(u, \mathsf{f}x, y\mathsf{g})$ or $(u, \mathsf{f}x\mathsf{g})$ where u is a useful occurrence of an edge addition operation f (cf. De…nition 2). Each such occurrence u creates a unique edge or loop because t is a special VR-term. Hence, the useful occurrences of edge addition operations can be used to represent edges. They form the set $Occ_1(t)$. A reduced term is a special VR-term such that all occurrences of edge addition operations are useful. We will denote by $RT(F_C^{\mathsf{VRd}})$ and $RT(F_C^{\mathsf{VRd}})$ the sets of reduced terms in $T(F_C^{\mathsf{VRd}})$ and $T(F_C^{\mathsf{VRd}})$ respectively. If a special term is not reduced, it can be transformed into a smaller equivalent reduced term by deleting the edge addition operations that are not useful.

In order to encode $\mathsf{f}X_1, ..., X_n, Y_1, ..., Y_m\mathsf{g}$-assignments, we will use, the sets of operations $F_C^{\mathsf{VRd}(n,m)}$ and $F_C^{\mathsf{VRu}(n,m)}$ instead of $F_C^{\mathsf{VRd}(n)}$ and $F_C^{\mathsf{VRu}(n)}$: the set $F_C^{\mathsf{VRd}(n,m)}$ is obtained from $F_C^{\mathsf{VRd}(n)}$ by replacing every edge addition operation f by the unary operations (f, w), for all w in $\mathsf{f}0, 1\mathsf{g}^m$ and similarly for $F_C^{\mathsf{VRu}(n,m)}$.

We will use the projections pr_s as in Theorem 13 and the projections pr_s^{0}, that delete the last s Booleans in the unary operations (f, w). It is clear that a term $t\,\mathsf{¤}\,\gamma$ 2 $T(F_C^{\mathsf{VRd}(n,m)})$ such that t is a special VR-term and the occurrences of edge addition operations in t are all useful, de…nes a concrete graph $cval(t)$ and an $\mathsf{f}X_1, ..., X_n, Y_1, ..., Y_m\mathsf{g}$-assignment γ such that $\gamma(X_i)$ is a set of vertices (for i 2 $[n]$) and $\gamma(Y_j)$ is a set of edges (for j 2 $[m]$).

We let $RT(F_C^{\mathsf{VRd}(n,m)})$:= $pr_m^{\mathsf{0}}\mathsf{i}^{\,1}(pr_n^{\,\mathsf{i}\,1}(RT(F_C^{\mathsf{VRd}})))$ µ $T(F_C^{\mathsf{VRd}(n,m)})$ and similarily for $RT(F_C^{\mathsf{VRu}(n,m)})$.Whether a term t in $T(F_C^{\mathsf{VRd}(n,m)})$ is reduced or

not, i.e., belongs or not to $RT(F_C^{\mathsf{VRd}(n,m)})$ does not depend on the Boolean components of its constant symbols and unary edge addition operations. We now sketch the construction of an F_C^{VRd}-automaton R that recognizes $RT(F_C^{\mathsf{VRd}})$. Its set of states is fA j A µ C ¡ f?gg [f$Error$g. The meanings of these states are described as follows for a term t in $T(F_C^{\mathsf{VRd}})$ (in a similar way as in as in Table 1):

P_A , t 2 $RT(F_C^{\mathsf{VRd}})$ and A = π_1 (t),
P_{Error} , t 2/ $RT(F_C^{\mathsf{VRd}})$.

The transition rules are in Table 3. All states except $Error$ are accepting. This automaton checks simultaneously that the given term is a special term and that it is reduced.

Transition rules	Conditions
? ! ; a ! fag	
relab$_h$[A] ! $h(A)$ ¡ f?g	if a, b 2 A, a 6= b and $h(a)$ = $h(b)$, then $h(a)$ = ?
add$_a^{loop}$[A] ! A	a 2 A
add$_{a,b}$[A] ! A	a, b 2 A
©[A, B] ! A [B	A \ B = ;

Table 3: The transition rules of R.

By replacing in this table every edge addition operation f by the unary operations (f, w) for w 2 f0, 1g^m and every constant symbol a by (a, w) for w 2 f0, 1g^n , we obtain an automaton with the same set of states that recognizes $RT(F_C^{\mathsf{VRd}(n,m)})$. Similar constructions can be done for $RT(F_C^{\mathsf{VRu}(n,m)})$.

If F is a …nite subset of F^{VRd} or of F^{VRu}, then it is contained in F_C^{VRd} or in F_C^{VRu} for some …nite set C, and $F^{(n,m)}$ denotes the corresponding subset of $F_C^{\mathsf{VRd}(n,m)}$ or of $F_C^{\mathsf{VRu}(n,m)}$. Let us …x such F (to simplify notation). For every CMS$_2$ formula φ with free variables in f$X_1, ..., X_n, Y_1, ..., Y_m$g written with in if it concerns undirected graphs, or in_1 and in_2 if it concerns directed graphs, we de…ne $L_{\varphi,(X_\mathbf{1},...,X_n,Y_\mathbf{1},...,Y_m)}$ as the set ft¤γ 2 $RT(F^{(n,m)})$ j (d$cval$(t)e, γ) j= φg. The language $L_{P(X_\mathbf{1},...,X_n,Y_\mathbf{1},...,Y_m)}$ can be de…ned similarly for a graph property P independently of its logical expression.

Theorem 27: Let F be a …nite subset of F^{VRd} or of F^{VRu}. For every CMS$_2$ graph property $P(X_1, ..., X_n, Y_1, ..., Y_m)$, the language $L_{P(X_\mathbf{1},...,X_n,Y_\mathbf{1},...,Y_m)}$ is regular and an F-automaton recognizing it can be constructed from a CMS$_2$ formula that de…nes P.

Proof: As for proving Theorem 13, we will construct by induction on the structure of φ an F-automaton A$_{\varphi,(X_\mathbf{1},...,X_n,Y_\mathbf{1},...,Y_m)}$ that recognizes the language $L_{\varphi,(X_\mathbf{1},...,X_n,Y_\mathbf{1},...,Y_m)}$.

1) If φ is $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$, then one uses the classical constructions of (product) automata for intersection and union since we have

$$L_{\varphi_1 \wedge \varphi_2,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)} = L_{\varphi_1,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)} \setminus L_{\varphi_2,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$$

and similarly for $\vee$ with $\cup$. If φ is $\neg\varphi_1$, we construct an automaton that recognizes $L_{\neg\varphi_1,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)} = RT(F^{(n,m)}) \setminus L_{\varphi_1,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$.

2) If φ is $\exists X_n.\theta$, then we have:

$$L_{\varphi,(X_1,\ldots,X_{n-1},Y_1,\ldots,Y_m)} = pr_1(L_{\theta,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}),$$

and if φ is $\exists Y_m.\theta$, we have:

$$L_{\varphi,(X_1,\ldots,X_n,Y_1,\ldots,Y_{m-1})} = pr_1^1(L_{\theta,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}).$$

It is straightforward to obtain from the deterministic $F^{(n,m)}$-automaton that recognizes $L_{\theta,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$ a nondeterministic automaton for $L_{\varphi,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$, that we determinize to get the desired one.

3) It remains to construct automata for the atomic formulas. Most of the constructions are straightforward from the definitions, as in Theorem 13. We only consider the atomic formulas $edg(X_1, X_2)$ and $in(Y_1, X_1)$.

The automaton A' for $edg(X_1, X_2)$ is derived from the automaton A of Theorem 13. Its set of states is (we name its states as those of A):

$$S' := \{0, Error, Ok\} \cup \{1(a), 2(a), a(0), ab(0) \mid a, b \in C \setminus \{?\}\}.$$

The meanings of these states are as in Table 1 (Theorem 13) where Ok replaces all the states $a(i)$ and $ab(i)$ for $i \geq 1$ because here, we do not count edges, we only want to check the existence of at least one edge from the vertex in V_1 to the one in V_2. The number of states is $k^2 + 3(k+1)$ where $k = |C \setminus \{?\}|$. The transition rules are in Table 4. The missing transitions yield $Error$. Here is an example: $relab_{a \to ?}[ab(0)] \to$ Error. The unique accepting state is Ok.

However, the automaton A' has been constructed so as to work correctly on reduced terms, not on all terms. The automaton $A_{edg(X_1,X_2)}$ is then obtained by a product with the one that recognizes reduced terms, so that it recognizes $L(A') \cap RT(F_C^{VRd(2,0)})$. Its number of states is thus $2^k.(k^2 + 3k + 3)$ instead of $k^2 + 3k + 3$. In the following remark, we will discuss this point.

We now construct an automaton B for $in_1(Y_1, X_1)$, intended to work on reduced terms. Its set of states is :

$$S'' := \{0, Error, Ok\} \cup \{1(a) \mid a \in C \setminus \{?\}\} \cup S'.$$

Their meanings are described in Table 5, where W_1 denotes the value of Y_1. The unique accepting state is Ok. As examples of transitions to $Error$ we give:

$\odot[Ok, 1(a)] \to$ Error,

$(add_a^{loop}, 1)[1(b)] \to Error$ if $b \neq a$, and

$(add_{a,b}, 1)[Ok] \to Error$.

Transition rules	Conditions
? ! 0	
(a, 00) ! 0	
(a, 10) ! 1(a)	
(a, 01) ! 2(a)	
(a, 11) ! a(0)	
relab$_h$[0] ! 0	
relab$_h$[i(a)] ! i(c)	i 2 [2]
relab$_h$[a(0)] ! c(0)	c = h(a) 6= ?, d = h(b) 6= ?,
relab$_h$[ab(0)] ! cd(0)	
add$_a^{loop}$[s] ! s	s 6= a(0)
add$_a^{loop}$[a(0)] ! Ok	
add$_{a,b}$[s] ! s	s 6= ab(0)
add$_{a,b}$[ab(0)] ! Ok	
©[1(a), 2(b)] ! ab(0)	
©[2(b), 1(a)] ! ab(0)	(possibly a = b)
©[s, 0] ! s	all s
©[0, s] ! s	

Table 4: The transition rules of A$^{\emptyset}$.

State s	Property P_s
0	V_1 = ;, W_1 = ;
1(a)	V_1 = fvg, W_1 = ;, port$_{\mathsf{cval}(t)}$(v) = a
Ok	V_1 = fvg, W_1 = feg, $in_{1\,cval(t)}(e, v)$
Error	All other cases

Table 5: Meanings of the states of B.

Transition rules	Conditions
? ! 0	
(a, 0) ! 0	
(a, 1) ! 1(a)	
relab$_h$[0] ! 0	
$relab_h$[Ok] ! Ok	
$relab_h$[1(a)] ! 1(b)	b = h(a) 6= ?
(add$_a^{loop}$, 0)[s] ! s	all s
(add$_a^{loop}$, 1)[1(a)] ! Ok	
(add$_{a,b}$, 0)[s] ! s	all s
(add$_{a,b}$, 1)[1(a)] ! Ok	
©[s, 0] ! s	all s
©[0, s] ! s	

Table 6: The transition rules of B.

Remark 28 : The above construction associates with each subformula $\theta(X_1, ..., X_n, Y_1, ..., Y_m)$ of the considered formula φ an automaton $A_{\theta,(X_1,...,X_n,Y_1,...,Y_m)}$ that recognizes only reduced terms. This means that each of these automata repeats the veri…cation that the input term is reduced. One can actually postpone this veri…cation to the very end.

Assume that for each atomic formula $\alpha(X_1, ..., X_n, Y_1, ..., Y_m)$, we have an automaton $B_{\alpha,(X_1,...,X_n,Y_1,...,Y_m)}$ such that

$$L_{\alpha,(X_1,...,X_n,Y_1,...,Y_m)} = L(B_{\alpha,(X_1,...,X_n,Y_1,...,Y_m)}) \setminus RT(F^{(n,m)}).$$

This means that $B_{\alpha,(X_1,...,X_n,Y_1,...,Y_m)}$ is constructed so as to works correctly on reduced terms, and this is what we did above for A^0 and B.

Let us build $B_{\varphi,(X_1,...,X_n,Y_1,...,Y_m)}$ for every all formulas φ by applying the general inductive construction described above with, for the negation:

$$L(B_{:\varphi_1,(X_1,...,X_n,Y_1,...,Y_m)}) = T(F^{(n,m)}) \text{ ¡ } L(B_{\varphi_1,(X_1,...,X_n,Y_1,...,Y_m)}).$$

At the end, for the input formula $\varphi(X_1, ..., X_n, Y_1, ..., Y_m)$, we make the restriction to reduced terms by de…ning $A_{\varphi,(X_1,...,X_n,Y_1,...,Y_m)}$ in such a way that:

$$L(A_{\varphi,(X_1,...,X_n,Y_1,...,Y_m)}) = L(B_{\varphi,(X_1,...,X_n,Y_1,...,Y_m)}) \setminus RT(F^{(n,m)}).$$

Hence, we use only once and at the end, the restriction to reduced terms. We claim that $L(A_{\varphi,(X_1,...,X_n,Y_1,...,Y_m)}) = L_{\varphi,(X_1,...,X_n,Y_1,...,Y_m)}$. This is true by the hypotheses on the automata B_α associated with the atomic formulas and by the following observations:

$$(L \setminus R) \setminus (M \setminus R) = ((L \setminus M) \setminus R),$$
$$(L \setminus R) \cup (M \setminus R) = ((L \cup M) \setminus R),$$
$$R \text{ ¡ } (L \setminus R) = (T \text{ ¡ } L) \setminus R,$$
$$pr(L^0 \setminus R^0) \setminus R = pr(L^0) \setminus R,$$

where $L, M, R, ...$ are sets such that $L, M, R \subseteq T$ and $L^0, R^0 \subseteq T^0$, and pr is a mapping from T^0 to T such that $R^0 = pr^{-1}(R)$.

However, the automaton $L(B_\varphi)$ associated with a sentence φ is su¢cient if all its inputs are reduced terms. This may be guaranteed by a preprocessing algorithm and the advantage is that B_φ is smaller than A_φ. A crucial di¢culty in the implementation of this method comes from the sizes of the constructed automata. ¤

Tree-width versus special tree-width We now explain why the constructions of automata are easier for graphs of bounded special tree-width than for those of bounded tree-width.

De…nition 29 : Special HR-terms.
We let F^{HRd} be the set obtained from F^{VRd} by replacing the operation © by //. This operation symbol will be interpreted as follows: for directed p-graphs G and H such that, as in De…nition 15, $\pi(G)$ ¡ $\pi_1(G)$ µ f?g and $\pi(H)$ ¡ $\pi_1(H)$ µ f?g, we let $G//H$ be obtained from G © H by the fusion of any two vertices having the same port label a 6= ?. An HR-term is a term t in $T(F^{\mathsf{HRd}})$ such that:

1) $\pi(t^0)$ ¡ $\pi_1(t^0)$ µ f?g for every subterm t^0 of t,

2) for every relabelling relab_h occurring in t, we have $h(?) = ?$,

3) for every operation $\mathrm{add}_{a,b}$, add_a^{loop} that occurs in t, we have $a \neq ?$ and $b \neq ?$,

4) the constant symbol ? has no occurrence in t.

We denote by $HT(F^{\mathsf{HRd}})$ the set of HR-terms. (The accronym HR refers to hyperedge-replacement graph grammars; see Chapter 4 of [6]). The notations F_C^{VRd} and $F_C^{\mathsf{VRd}(n,m)}$ extend in the obvious way, yielding sets like $HT(F_C^{\mathsf{HRd}(n,m)})$, that are, clearly, regular languages. These de…nitions also extend to undirected graphs, giving F^{HRu} , F_C^{HRu}, $F_C^{\mathsf{HRu}(n,m)}$ etc... Every graph is the value $val(t)$ of some term HR-term t, using a large enough set of labels.

Proposition 30 : The tree-width of a graph is the least integer j C ¡ f?g j ¡ 1 such that this graph is the value of a term in $HT(F_C^{\mathsf{HRd}})$ [$HT(F_C^{\mathsf{HRu}})$. There are linear-time algorithms for converting a term t in $HT(F_C^{\mathsf{HRd}})$ [$HT(F_C^{\mathsf{HRu}})$ into a tree-decomposition of width j C ¡ f?g j ¡1 of the graph $val(t)$ and vice-versa.

Proof: The proof is an easy variant of the proof of Proposition 18. It is done in detail in [6], Chapter 2 (with slightly di¤erent de…nitions).¤

Let us go back to De…nition 26, where we discuss the encoding of assignments in terms. Let t be an HR-term and G be a concrete graph isomorphic to $val(t)$. Its edges are in bijection with the set $Occ_1(t)$ de…ned as for special VR-terms. However, its vertex set is isomorphic to the quotient of $Occ_0(t)$ by the equivalence relation ¼ expressing that two leaves x and y in $Occ_0(t)$ have a least common ancestor u that is an occurrence of //, and that $port_t(x,u) = port_t(y,u) \neq ?$. This implies that they are fused at some stage and yield the same vertex of G. Hence, we have no nice bijection between the vertices of G and particular occurrences of symbols in t. A set X µ $Occ_0(t)$ represents correctly a set of vertices of G if and only it is saturated for ¼ (is a union of classes

of this equivalence). The automata analogous to $B_{\varphi,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$ would have to check this saturation property, which would increase substancially their numbers of states. (Taking the leftmost occurrence of each equivalence class of ¼ as representation of the corresponding vertex would not make things easier).

There is actually another possibility for representing vertices in terms. Let us assume that G is a concrete graph (and not a concrete p-graph), hence that its vertices are all ?-ports. This implies that each vertex of G corresponds to a unique occurrence of an operation $relab_{a\,!\,?}$. Such occurrences, let us denote their set by $Occ_1^{vert}(t)$, can be chosen to represent the vertices. In this case, an edge will be represented by a node in the term that is below the nodes representing its ends. This is not a di¢culty for constructing automata for the atomic formulas $in(Y_1, X_1), in_1(Y_1, X_1)$ and $in_2(Y_1, X_1)$ like B in the proof of Theorem 27. These automata have also k + 3 states (where k =j C ¡ f?g j), but the construction of automata for $edg(X_1, X_2)$ is more complicated. Since $edg(X_1, X_2)$ is equivalent (for directed graphs) to $9Y_1(in_1(Y_1, X_1)\wedge in_2(Y_1, X_2))$, the general construction can be used, and it produces an automaton with $2^{O(k^2)}$ states. (The term k^2 is due to the use for ^ of a product of two automata, and the exponentiation is due to the determinization that is needed because of $9Y_1$). However, every deterministic automaton for $edg(X_1, X_2)$ must have at least $2^{k(k\,i\,1)}$ states (Section 6.3.5 of [6]). Hence, with this representation, an atomic formula like $edg(X_1, X_2)$ needs already fairly "large" automata.

> Question: Does there exist alternative encodings of tree-decompositions of width k by terms (or labelled trees) for which the automata associated with $edg(X_1, X_2)$, $in(Y_1, X_1), in_1(Y_1, X_1)$ and $in_2(Y_1, X_1)$ have, say, $O(k^2)$ states?

7 Conclusion

We have extended the notion of clique-width to graphs with multiple edges. Without introducing edge set quanti…cations, we have extended monadic second-order logic so as to take into account the multiplicity of edges. We have generalized the usual constructions of automata and obtained a …xed-parameter tractable model-checking algorithm with respect to the parameter consisting of clique-width and edge-thickness. The problem remains open of eliminating edge-thickness from the parameter, which amounts to …nding a …xed-parameter tractable parsing algorithm for clique-width where clique-width is the only parameter.

The constructions of automata used for establishing …xed-parameter tractability results for monadic second-order sentences are di¢cult if not impossible in

practice, because of the sizes of the de…ned automata. This di¢culty is not avoidable for general monadic second-order formulas as proved by [15, 22] and [23]. But even for a basic property like connectedness, the minimal $F_{[k]}^{VRu}$-automaton has more than $2^{2^{k/2}}$ states (Chapter 6 of [6]). An attractive possibility is to replace the "compilation" of automata by the computation of the needed transitions for each input term. Such ‡y-automata are introduced and used in [7] and studied in [8]. The notion of special tree-width has been introduced in order to facilitate the speci…cation of such automata. The corresponding parsing problem, presented at the end of Section 5, is open.

Acknowledgements: I thank H. Bodlaender, M. Kanté, M. Rao and the referees for their useful comments.

8 References

[1] H. Bodlaender and J. Engelfriet, Domino tree-width, J. Algorithms 24 (1997) 94-123.

[2] K. Cameron, C. Hoàng and B. Lévêque, Asteroids in rooted and directed path graphs, Electronic Notes in Discrete Mathematics 32 (2009) 67-74. ...

[3] H. Comon et al., Tree automata techniques and applications, On line for free at: http://tata.gforge.inria.fr/

[4] D. Corneil and U. Rotics, On the relationship between clique-width and treewidth. SIAM J. Comput. 34 (2005) 825-847

[5] B. Courcelle, The monadic second-order logic of graphs I: Recognizable sets of …nite graphs, Information and Computation, 85 (1990) 12-75.

[6] B. Courcelle, Graph structure and monadic second-order logic, book to be published by Cambridge University Press. Readable on:

http://www.labri.fr/perso/courcell/Book/CourGGBook.pdf

[7] B. Courcelle and I. Durand, Verifying monadic second-order graph properties with tree automata, 3rd European Lisp Symposium, May 2010, Lisbon, Informal proceedings edited by C. Rhodes, pp.7-21. See:

http://www.labri.fr/perso/courcell/ArticlesEnCours/BCDurandLISP.pdf

[8] B. Courcelle and I. Durand, Automata for the veri…cation of monadic second-order graph properties, In preparation, 2010.

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

[10] B. Courcelle and S. Olariu, Upper bounds to the clique-width of graphs, Discrete Applied Mathematics 101 (2000) 77-114.

[11] R. Diestel, Graph Theory, 3rd edition, Springer, 2005.

[12] R. Downey and M. Fellows, Parameterized complexity, Springer, 1999.

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

[14] J. Flum and M. Grohe, Parametrized complexity theory, Springer, 2006.

[15] M. Frick and M. Grohe: The complexity of …rst-order and monadic second-order logic revisited. Ann. Pure Appl. Logic 130 (2004) 3-31

[16] F. Gavril, A recognition algorithm for the intersection graphs of directed paths in directed trees. Discrete Mathematics 13 (1975) 237-249.

[17] M. Golumbic and U. Rotics, On the clique-width of certain perfect graph classes, International Journal of Foundations of Computer Science, 11 (2000) 423-443.

[18] L. Libkin, Elements of …nite model theory, Springer, 2004.

[19] P. Hliněný and S. Oum, Finding branch-decompositions and rank-decompositions. SIAM J. Comput. 38 (2008) 1012-1032.

[20] S. Oum and P. Seymour, Approximating clique-width and branch-width. J. Comb. Theory, Ser. B 96 (2006) 514-528.

[21] D. Seese, Tree-partite graphs and the complexity of algorithms (extended abstract), in Foundations of Computation Theory, L. Budach ed., Lec. Notes Comput. Sci. 199 (1985) 412-421.

[22] L. Stockmeyer and A. Meyer, Cosmological lower bound on the circuit complexity of a small problem in logic. J. ACM 49 (2002) 753-784.

[23] M. Weyer, Decidability of S1S and S2S, in Automata, Logics, and In…nite Games: A guide to current research, Lecture Notes in Computer Science 2500 (2002) 207-230.

[24] D. Wood, On tree-partition-width, European Journal of Combinatorics 30 (2009) 1245-1253