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 quantifications.

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

May 7, 2010

Abstract

In order to obtain fixed-parameter tractable model-checking algorithms for monadic second-order graph properties that depend on the multiplicities of edges, we extend clique-width to graphs with multiple edges, and we extend counting monadic second-order logic accordingly. For monadic second-order graph properties that need edge set quantifications in their logical expressions, we define a graph complexity measure called *special tree-width*. Its value is between path-width and tree-width. We study its main properties and we explain why it is better that tree-width.

1 Introduction

It is well-known that the model-checking problem for graph properties expressed by monadic second-order sentences with edge set quantifications is fixed-parameter tractable for tree-width as parameter ([DF], [FG]) and that, for graph properties expressed by the (basic) monadic second-order sentences without edge set quantifications, it is fixed-parameter tractable (by [CMR] together with the approximate parsing algorithm of [HliOum]; see [Cou], Chapter 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, defined 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 not using

edge set quantifications. The known fixed-parameter tractable algorithm extends provided the input graph is given with a term (i.e., its decomposition) witnessing that its clique-width is bounded by a given integer k. The idea governing this extension of monadic second-order logic is that, when objects have no identity, we can only *count them*. And if we use a computing device that does not allow arbitrary large integers, we can only *count them up to a threshold* or *modulo some fixed integer*.

The fixed-parameter monadic second-order model-checking algorithms for tree-width or clique-width as parameters are based on constructions of finite 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 difficulty lead us to the definition of particular tree-decompositions called provisionally *special tree-decompositions*, that yield the notion of *special tree-width*. This parameter, that is new to our knowledge (but it may be equivalent to some other one) takes values between path-width and tree-width. Graphs of tree-width 2 have unbounded special tree-width. Special tree-width can be defined in terms of the generalized "clique-width operations" that operate on graphs with multiple edges. The corresponding constructions of finite automata from monadic second-order sentences using edge set quantifications are as easy as in the case where clique-width is the intended parameter.

All necessary definitions will be given, but we will frequently refer to definitions (of secundary importance) and to the constructions developped in detail in the book [Cou]. We will use as much as possible the notation and terminology of this book, but this article introduces definitions that will not be included in it. Section 2 introduces the clique-width of graphs with multiple edges, Section 3 defines the relevant extension of counting monadic second-order logic. The applications to model-checking are in Section 4. Special tree-width is defined and studied in Section 5. Its application to model-checking is in Section 6 where we also explain why special tree-width is better than tree-width in this respect. Section 7 is a short conclusion.

2 Graphs algebras

All graphs and relational structures will be finite.

Definition 1 : Graphs

We will consider *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 graph G is a triple $(V_G, E_G, vert_G)$ with vertex set V_G, edge set E_G and incidences defined 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 $e : x-_G y$ means that e is an undirected edge that links x and y, and $e : x \longrightarrow_G y$ 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. The graph G is *simple* if $vert_G(e) \neq vert_G(e')$ for $e \neq e'$. We let $Spl(G)$ be the simple graph obtained from G by fusing any two *parallel* edges, that is, any two edges e and e' such that $vert_G(e) = vert_G(e')$. We can also describe $Spl(G)$ as obtained by iterating the removal of one element of a pair of parallel edges until one gets a simple graph. Hence, $Spl(G)$ can be seen as a subgraph of G.

An *abstract graph* is a graph up to isomorphism, i.e., formally, the isomorphism class of a (concrete) graph. The reader will find the (obvious) definitions in Chapter 2 of [Cou].

Definition 2 : Operations on graphs.

Let $\mathcal{A}$ be a countable set of *port labels* containing the set $\mathcal{N}$ of nonnegative integers and the special symbol $\perp$. Unless otherwise specified, the definitions are the same for directed and undirected graphs. A *graph with ports*, or a *p-graph* in short, is a pair $G = (G^\circ, port_G)$ consisting of a graph G° and a mapping $port_G : V_{G^\circ} \to \mathcal{A}$. A vertex x is an *a-port* of G if $port_G(x) = a$. The *type* $\pi(G)$ of G is the set $port_G(V_G)$ of port labels of its vertices. (We denote also V_{G° by V_G, and similarly for other items).

If G and H are p-graphs, we say that G is a *subgraph* of H if G° is a subgraph of H° and $port_G$ is the restriction of $port_H$ to V_G (so that $\pi(G) \subseteq \pi(H)$).

Every graph will be considered as a p-graph, all vertices of which are $\perp$-ports (hence, $\perp$ is a default port label).$\square$

Our next objective is to define *signatures* of graph operations, more precisely, of operations that act on directed and undirected p-graphs.

Disjoint union. Two graphs G and H are disjoint if $V_G \cap V_H = \emptyset$ and $E_G \cap E_H = \emptyset$, so that one can take their union in an obvious way. For disjoint p-graphs G and H, we let $G \oplus H$ be the union G° and H° equipped with the port mapping $port_{G\oplus H} := port_G \cup port_H$. 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-defined binary operation on abstract p-graphs. Clearly

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

Edge addition. Let $a, b \in \mathcal{A}$, with $a \neq b$. For every directed p-graph G, we let $\overrightarrow{add}_{a,b}(G)$ be the p-graph G' such that $V_{G'} := V_G$, $E_{G'}$ is E_G to which we add one edge e from x to y for every $x, y \in V_G$ such that $port_G(x) = a$ and $port_G(y) = b$ (so that $vert_{G'}(e) := vert_G(e)$ if $e \in E_G$ and $vert_{G'}(e) := (x, y)$ if $e \in E_{G'} - E_G$ is as above), and $port_{G'} := port_G$.

For adding a loop, we use the operation add_a^{loop} that adds a loop at each vertex x such that $port_G(x) = a$.

For adding undirected edges, we use the operation $add_{a,b}$ defined similarly as $\overrightarrow{add}_{a,b}$. There is no difference 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:

$$\pi(\overrightarrow{add}_{a,b}(G)) = \pi(add_{a,b}(G)) = \pi(add_a^{loop}(G)) = \pi(G).$$

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

Port relabelling. Let $h : \mathcal{A} \to \mathcal{A}$ is a mapping that is the identity outside of a finite subset of $\mathcal{A}$. We define $relab_h$ as the unary operation such that $relab_h(G)$ is the p-graph G' such that $V_{G'} := V_G$, $E_{G'} := E_G$, $vert_{G'} := vert_G$ and $port_{G'} := h \circ port_G$. We have :

$$\pi(relab_h(G)) = h(\pi(G)).$$

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

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

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

The two VR algebras of p-graphs. We obtain two countably infinite signatures, the first one acts on directed p-graphs:

$$F^{\mathrm{VRd}} := \{\oplus, \overrightarrow{add}_{a,b}, add_a^{loop}, relab_h, \mathbf{a}, \varnothing \mid a, b \in \mathcal{A},\ a \neq b,\ h \in [\mathcal{A} \to \mathcal{A}]_f\}$$

and the second one on undirected p-graphs:

$$F^{\mathrm{VRu}} := \{\oplus, add_{a,b}, add_a^{loop}, relab_h, \mathbf{a}, \varnothing \mid a, b \in \mathcal{A},\ a \neq b,\ h \in [\mathcal{A} \to \mathcal{A}]_f\}.$$

We let $\mathbb{GP}^{\mathrm{d}}$ denote the F^{VRd}-algebra with domain $\mathcal{GP}^{\mathrm{d}}$ defined as the set of all (abstract) directed p-graphs, and we let $\mathbb{GP}^{\mathrm{u}}$ be the corresponding F^{VRu}-algebra of undirected p-graphs with domain $\mathcal{GP}^{\mathrm{u}}$. We call them the *VR algebras*. (There are "historical reasons" for this terminology: see [Cou], Chapter 4).

Each term t over F^{VRd} (resp. over F^{VRu}) evaluates into a directed (resp. an undirected) concrete p-graph denoted by $cval(t)$. Its set of vertices is $Occ_0(t)$,

the set of occurrences in t of the constant symbols $\mathbf{a}$ for $a \in \mathcal{A}$, and its edges are the pairs $(u,(x,y))$ such that u is an occurrence of an operation $\overrightarrow{add}_{a,b}$ that creates an edge from x to y, the pairs $(u,\{x,y\})$ such that u is an occurrence of $add_{a,b}$ that creates an undirected edge between x and y and the pairs $(u,\{x\})$ such that u is an occurrence of add_a^{loop} that creates a loop incident with x. The formal definition of $cval(t)$, using an induction on the structure of t, is clear from the definitions of the operations. We denote by $val(t)$ the corresponding abstract p-graph (the isomorphism class of $cval(t)$). Two terms are *equivalent* if they evaluate into the same abstract p-graph.

The signatures F_C^{VRd} and F_C^{VRu} for $C \subseteq \mathcal{A}$ 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 $t \in T(F_C^{\mathrm{VRd}}) \cup T(F_C^{\mathrm{VRu}})$. Every term in $T(F_C^{\mathrm{VRd}}) \cup T(F_C^{\mathrm{VRu}})$ that denotes a nonempty p-graph is equivalent to a term in $T(F_C^{\mathrm{VRd}} - \{\varnothing\}) \cup T(F_C^{\mathrm{VRu}} - \{\varnothing\})$.

The two VR algebras of simple p-graphs. The following facts are clear from the definitions:

$Spl(G \oplus H) = Spl(G) \oplus Spl(H)$,
$Spl(\overrightarrow{add}_{a,b}(G)) = Spl(\overrightarrow{add}_{a,b}(Spl(G)))$,
$Spl(add_{a,b}(G)) = Spl(add_{a,b}(Spl(G)))$,
$Spl(add_a^{loop}(G)) = Spl(add_a^{loop}(Spl(G)))$,
$Spl(relab_h(G)) = relab_h(Spl(G))$,
$Spl(\mathbf{a}) = \mathbf{a}$ and $Spl(\varnothing) = \varnothing$.

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

$\overrightarrow{add}^{s}_{a,b}(G) = Spl(\overrightarrow{add}_{a,b}(G))$,
$add^{s}_{a,b}(G) = Spl(add_{a,b}(G))$,
$add^{s,loop}_{a}(G) = Spl(add^{loop}_{a}(G))$.

To take an example, the term $t = add_{a,b}(add_{a,b}(\mathbf{a} \oplus \mathbf{b}) \oplus \mathbf{b})$ evaluates in $\mathbb{GP}^{\mathrm{su}}$ into the simple graph $b - a - b$ (with 2 edges) and, in $\mathbb{GP}^{\mathrm{u}}$, into the graph $b = a - b$ (with 3 edges).

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

Definition 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 $T(F_C^{\mathrm{VRd}}) \cup T(F_C^{\mathrm{VRu}})$. This number is denoted by $cwd(G)$. It is easy to prove that every p-graph G is the value of some term in $T(F^{\mathrm{VRd}}) \cup T(F^{\mathrm{VRu}})$ and that $cwd(G) \leq \mid V_G \mid$. A simple graph can be defined as $val(t)$ for some term $t \in T(F_C^{\mathrm{VRd}}) \cup T(F_C^{\mathrm{VRu}})$, but it can also be defined 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 $t \in T(F_C^{\mathrm{VRd}}) \cup T(F_C^{\mathrm{VRu}})$, then $G = val(t')$ for some term $t' \in T(F_C^{\mathrm{VRd}}) \cup T(F_C^{\mathrm{VRu}})$. For every graph G, we have $cwd(Spl(G)) \leq cwd(G)$, and the inequality may be strict.

Proof : Let $G = sval(t)$ and $H := val(t)$. If H is simple, we take $t' := t$. Otherwise H has at least two edges e and e' such that $vert_H(e) = vert_H(e')$. They are specified by edge addition operations, $\overrightarrow{add}_{a,b}$ at an occurrence u and $\overrightarrow{add}_{c,d}$ at an occurrence v that is an ancestor of u in the syntactic tree of t. The pair (c,d) may differ from (a,b) because of possible relabellings on the path between u and v in this tree. Since e' is parallel to e, all edges created by $\overrightarrow{add}_{a,b}$ at u have parallel edges created by $\overrightarrow{add}_{c,d}$ at v. Hence, if we replace $\overrightarrow{add}_{a,b}$ at u by the identity (say by $relab_{Id}$), 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 this transformation step (that does not introduce new port labels) finitely many times, we obtain a term $t' \in T(F_C^{\mathrm{VRd}}) \cup T(F_C^{\mathrm{VRu}})$ such that $val(t') = sval(t)$.

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

$$\overrightarrow{add}_{a,b}(relab_{c \to a}(\overrightarrow{add}_{c,b}(\overrightarrow{add}_{a,b}(\mathbf{a} \oplus \mathbf{b}) \oplus \mathbf{b} \oplus \mathbf{c}))),$$

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

Clique-width has been defined in [CouOla], [Cou], [CMR] for *simple graphs* only, as the minimal cardinality of a set C such that $G = sval(t)$ for some term $t \in T(F_C^{\mathrm{VRd}}) \cup T(F_C^{\mathrm{VRu}})$. The first assertion of Proposition 4 shows that the new definition agrees for simple graphs with the usual one.

Another technical point is discussed in [Cou], Chapter 2: the clique-width of simple graphs can be defined by replacing in the signatures 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 definition of [CouOla]). The resulting values of clique-width are the same with the two definitions. (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 [Cou] 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 defined for simple (L, Λ)-*labelled graphs*, i.e., for graphs such that every edge has a unique label from a fixed finite set Λ and every vertex has a possibly empty set of labels from a fixed finite set L disjoint from $\mathcal{A}$. We refer the reader to [Cou] for the detailed definitions. We only recall that clique-width is independent of the labelling of vertices. The extension of the above definitions to (L, Λ)-labelled graphs with multiple edges is straightforward and does not offer any particularly interesting question.

The following proposition shows that adding 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 simple undirected and loop-free graphs such that H is a subgraph of K and $V_H = V_K$; let $K + H$ be 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 definition. Let K be the graph (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 parallel edges between x and y, and z and u.

Proof of the claim: We let $K+H$ be defined by a term t in $T(F_C^{\mathrm{VRu}})$ such that C has the minimal cardinality, i.e. $\mid C \mid = cwd(K + H)$. We can assume that each occurrence u of an operation $add_{a,b}$ is *useful in* t, i.e., that it creates at least one edge. This is equivalent to the condition that a and b belong to $\pi(val(t/u_1))$, where t/u_1 is the subterm of t issued from u_1, the son of u. We can assume this condition because if an occurrence u of an operation $add_{a,b}$ is not useful, we can replace $add_{a,b}$ at u by $relab_{Id}$, the identity operation (or we can delete it), and we get an equivalent term written with no other port labels than those of C.

If u is an occurrence of $add_{a,b}$ and v is an occurrence of $add_{c,d}$, we write $u \sqsubset v$ if and only if u is below v in t (hence $u \neq v$) and the relabellings on the path in t between u and v (composed bottom-up) transform $\{a, b\}$ into $\{c, d\}$. 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 is clear that *each edge* created by $add_{a,b}$ at u gets by $add_{c,d}$ at v a parallel edge.

It is also clear that two parallel edges of $K+H$ are created by such operations, at some u and v such that $u \sqsubset v$. It follows from these observations that there are no 3 occurrences u, v, w of edge addition operations such that $u \sqsubset v \sqsubset w$, otherwise, we would have a triple of parallel edges. This is not possible by the definition of $K+H$.

We now transform t into $t' \in T(F_C^{\mathrm{VRu}})$ as follows: if u is an occurrence of $add_{a,b}$ in t such that there is no v with $u \sqsubset v$, then we replace $add_{a,b}$ by the identity operation at u. We claim that $H = val(t')$. Consider an edge e of $K+H$ without parallel edge: it is created by an operation $add_{a,b}$ at some u such that there is no v with $u \sqsubset v$, hence this operation is replaced in t' by the identity and this edge is not in $val(t')$; if e has a parallel edge, these two edges are created by edge additions at u and v such that $u \sqsubset v$. The operation at v is replaced by the identity (but not the operation at u), hence only one of the two edges remains in $val(t')$. This shows that $H = val(t')$. Hence, $cwd(H) \leq cwd(K+H)$.□

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 fixed function f we would have $cwd(G) \leq f(cwd(Spl(G)))$ for every graph G having no triple of parallel edges, then by taking $G := K+H$, we would have $cwd(H) \leq cwd(K+H) \leq f(2)$. But the simple undirected and loop-free graphs have unbounded clique-width ([CouOla], [Cou], [GolRot]), hence we get a contradiction.

The proof is easily adapted for directed graphs. □

Although the following notion is well-known we recall the definition at least for notation.

Definition 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 $f : N_T \longrightarrow \mathcal{P}(V_G)$ 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 isknown from [CouOla], [Cou] and [CorRot] 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 $G \otimes *$ be the graph obtained by adding to G a *universal vertex*, i.e., a vertex $*$ linked to all vertices of G (by undirected edges if G is undirected and by edges directed towards $*$ 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 $G \otimes *$ by the addition of parallel edges to all edges of $G \otimes *$, 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 $t \in T(F_C^{\mathrm{VRu}})$ be a term that defines $\widehat{G}$ by using k labels. Without loss of generality, we will identify the vertices of $\widehat{G}$ with the elements $Occ_0(t)$ (the occurrences in t of the constant symbols different from $\varnothing$; cf. Definition 2), and we will consider them as the leaves of t (we will identify a term and its the syntactic tree).

Let x be a vertex (a leaf of t) and 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 x in the graph $cval(t)/u$ with set of vertices $\{y \in Occ_0(t) \mid y \leq_t u\}$ that is the value of t/u (the subterm of t issued from node u). 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 $\oplus$. 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 $lca(x, y) \leq_t w$, $port_t(x, w) = a$ and $port_t(y, w) = b$. We say that *x is live at u* if there is a vertex y adjacent to x such that $u \leq_t lca(x, y)$.

We let P be the path in t linking the root to the leaf $*$. 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 $*$, hence it is live at $lca(x, *)$ and belongs to the box $f(lca(x, *))$.

(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 $lca(x, y) \leq_t u$. The vertices x and y have different port labels at u: there is a vertex z adjacent to x such that $u \leq_t lca(x, z)$. If x and y had the same port labels at u, they would be both adjacent to z with the same numbers of parallel edges, but this is not possible by the construction of $\widehat{G}$.

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 $twd(\widehat{G}) = 2$, but trees have unbounded path-width. Hence, the clique-widths of the graphs $\widehat{G}$ 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 $cwd(G) \leq f(cwd(Spl(G))$, the graphs of tree-width 2 (with multiple edges) would have bounded clique-width because simple undirected (resp. directed) graphs of tree-width 2 have clique-width at most 6 by [CorRot] (resp. at most 65 by [CouOla]).

Definition 8: The parsing problem.
The *parsing problem for clique-width* consists in finding 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 [FRSS] but there exists an *approximation* algorithm, call it $\mathcal{AP}_{cwd}$ (by the results of [HliOum] and [OumSey]) that does the following, for some fixed functions f and g, and in time $O(g(k).n^3)$, 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 clique-width more than k, or it outputs a term witnessing that its clique-width is at most $f(k)$.

This result suffices to prove that the model-checking problem for every monadic second-order property is fixed-parameter cubic with respect to clique-width as parameter ([CMR] and [Cou] Chapter 6). It extends actually to simple (L, Λ)-labelled graphs because these graphs can be encoded into simple undirected $(L', \emptyset)$-labelled graphs for some fixed set L', i.e., into vertex-labelled graphs, and this encoding preserves for a set of graphs the property of having bounded clique-width; the details are in [Cou], Section 6.2. However, we do not see how to extend this type of encoding so as to handle multiple edges. Hence, it is an open problem to find an algorithm analogous to $\mathcal{AP}_{cwd}$ that would operate in time $O(g(k).n^c)$, or even in time $O(n^{c(k)})$ for some fixed constant or function c and for input graphs that are not simple. Proposition 5 shows that is not a trivial question.

The above definitions of the parsing problem and of approximation algorithms can of course be used for other width notions, like special tree-width to be defined below.

3 Monadic second-order logic

Definition 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., [CMR], [Cou], [DF], [FG], [Lib]). We only review some perhaps not so well-known notions and the relevant notation.

If $q \geq 2$ and $0 \leq p < q$, 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 *C_rMS formula* is a monadic second-order formula that can be written with the set predicates $Card_{p,q}$ for $q \leq r$. The *CMS formulas* are the same without any bound on q; the *C_0MS* (that are also the *C_1MS*) 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 first representation associates with every graph G the logical structure $\lfloor G \rfloor := \langle V_G, edg_G \rangle$ where edg_G the binary relation on vertices such that $(x, y) \in edg_G$ if and only if $vert_G(e) = \{x, y\}$ (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:

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

Since for every graph G, we have $\lfloor G \rfloor = \lfloor Spl(G) \rfloor$, 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 $Inc(G) := \langle V_G \cup E_G, in_G \rangle$ 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 $Inc(G) := \langle V_G \cup E_G, in_{1G}, in_{2G} \rangle$ 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 $\lceil G \rceil$ 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:

$$\lceil G \rceil \models \varphi(X_1, ..., X_n, Y_1, ..., Y_m)$$
if and only if $P(X_1, ..., X_n, Y_1, ..., Y_m)$ is true in G.

For example, the property Ham that an undirected graph has at least 3 vertices and an Hamiltonian cycle is an MS_2-property that is not CMS_1 (see [Cou], 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: $\lfloor G \rfloor_\sharp := \langle V_G, \sharp edg_G \rangle$. This pair is not a relational structure because $\sharp edg_G$ is a function with values in the infinite 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 $\lfloor G \rfloor_\sharp$ and $\lfloor H \rfloor_\sharp$ are isomorphic (in the obvious sense).

The CMS formulas that specify CMS_1 graph properties are written with the binary relation symbol edg. We define 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 $\lfloor G \rfloor_\sharp$ as follows :

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

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

The notation $\lfloor G \rfloor_\sharp \models \varphi(X_1, ..., X_n)$ 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 identified 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 $Edg_G(X, Y)$ be 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 $X \cap Y$.

Proposition 11 : For every p and q in $\mathcal{N}$, there exist $C_qMS_\sharp$-formulas that express, for every directed graph G and sets of vertices X and Y that the set $Edg_G(X,Y)$ has cardinality q and that its cardinality is equivalent to p modulo q (where in this case, we assume that $q \geq 2$ and $0 \leq p < q$). Similar formulas exist for undirected graphs. For simple graphs, these constructions yield respectively MS-formulas and C_qMS-formula.□

Proof: If G is a graph and $X, Y \subseteq V_G$, we let $\sharp Edg_G(X,Y)$ denote the cardinality of $Edg_G(X,Y)$.

(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 $(X \times Y) \cap edg_G$ and this pair is in edg_{2G},
>
> or there are exactly two pairs in $(X \times Y) \cap edg_G$ 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: $\sharp Edg_G(\{x\}, Y) = \sum_{i \in \mathcal{N}} i. \mid \{y \in Y \mid \sharp edg_G(x,y) = i\} \mid$. Let us compute this modulo q:

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

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 $\lfloor G \rfloor_\sharp$:

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

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

By similar observations, we get that $\sharp Edg_G(X,Y) \equiv p \pmod{q}$ if and only if there exists a q-tuple $(p_0, ..., p_{q-1}) \in A(p,q)$ such that, for each $i = 0, ..., q-1$, we have $p_i = \mathrm{mod}_q(\mid \{x \in X \mid \sharp Edg_G(\{x\}, Y) \equiv i \pmod{q}\} \mid)$. 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 $\lfloor G \rfloor_\sharp$:

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

(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 $\neg edg$ 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

$$\begin{aligned}\sharp Edg_G(X,X) =& \mid \{x \in X \mid (x,x) \in edg_G\} \mid + \\ & \mid \{(x,y) \in (X \times X) \cap edg_G \mid x \neq y\} \mid /2,\end{aligned}$$

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

$$\begin{aligned}\sharp Edg_G(X,X) =& \mid \{x \in X \mid (x,x) \in edg_G\} \mid + \\ & \mid \{(x,y) \in (X \times X) \cap edg_G \mid x \neq y\} \mid .\end{aligned}$$

If X and Y are disjoint, then $\sharp Edg_G(X,Y) =\mid (X \times Y) \cap edg_G \mid$ for simple undirected graphs, as for simple directed ones. In all cases, $Edg_G(X,Y)$ is the disjoint union of $Edg_G(X,Y-X)$, $Edg_G(X-Y,X\cap Y)$ and $Edg_G(X\cap Y,X\cap Y)$. 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

Definitions 12 : Assignments encoded in the terms that define graphs

Let F be a fixed finite subsignature 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 definable by a finite F-automaton. We will extend the proof given in [Cou], Chapter 6 to the language $CMS_\sharp$. This proof uses an induction on the structure of the sentences that define the properties P. Hence, we need automata associated with formulas having free variables to handle inductively the case of sentences of the form $\exists X_1,\ldots,X_n.\varphi$. Hence, we generalize the previous definition.

Let $P(X_1,\ldots,X_n)$ be a property of sets of vertices $X_1,\ldots,X_n$ of the graphs denoted by terms in $T(F)$. If a term t evaluates to G, there is a bijection of $Occ_0(t)$, the set of occurrences in t of the constant symbols different from $\varnothing$, onto V_G. In other words, by applying the definitions of the operations of F^{VRd} and F^{VRu}, cf. Definition 2, one can construct a concrete graph $cval(t)$ isomorphic to $val(t)$ with vertex set $Occ_0(t)$.

For example, consider the term $t = add_{a,b1}(add_{a,b2}(\mathbf{a}_3 \oplus_4 \mathbf{b}_5) \oplus_6 \mathbf{b}_7)$ where the indices from 1 to 7 designate the occurrences in t of the operation and constant symbols. We have $Occ_0(t) = \{3, 5, 7\}$ and the graph $cval(t)$ is $5_b = 3_a - 7_b$. (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 the signature obtained from F by replacing each constant symbol $\mathbf{a}$ by the constant symbols $(\mathbf{a}, w)$ where $w \in \{0,1\}^n$. For $1 \leq m \leq n$, we let $pr_m : F^{(n)} \longrightarrow F^{(n-m)}$ be the mapping, usually called a *projection,* that transforms $(\mathbf{a}, w)$ into $(\mathbf{a}, w')$ where w' is obtained from w by removing the last m Booleans. A term t in $T(F^{(n)})$ defines two things: first, 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 constants $(\mathbf{a}, w)$ 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 define $L_{P(X_1,...,X_n)}$ as the set of terms $t * (V_1, ..., V_n) \in T(F^{(n)})$ such that $P(V_1, ..., V_n)$ is true in $cval(t)$. If P is defined by a formula φ with free variables in $\{X_1, ..., X_n\}$, then we denote $L_{P(X_1,...,X_n)}$ by $L_{\varphi,(X_1,...,X_n)}$. (Some variables in $\{X_1, ..., X_n\}$ may not occur or may not be free in φ). The relevant signature F is fixed by the context.

Theorem 13: Let F be a finite subsignature 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 defining it can be constructed from a $CMS_\sharp$ formula that defines P.

Proof: The proof is a small extension of that given in [Cou], Chapter 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. Details are in [Cou], Section 6.3.

First, monadic second-order formulas can be written without first-order variables and without universal quantifications.

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 $edg(X_i, X_j)$ (in the constructions of [Cou]), and $edg_q(X_i, X_j)$, $edg_{p,q}(X_i, X_j)$ (in the present case). Their meanings, if not already defined 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 $\mathcal{A}_{\varphi,(X_1,...,X_n)}$ that recognizes the language $L_{\varphi,(X_1,...,X_n)}$. (All automata will be *finite, complete and bottom-up deterministic* unless otherwise specified). The construction is by induction on the structure of φ:

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

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 $(\mathbf{a}, wi)$, where i is 0 or 1, by $(\mathbf{a}, w)$, so that $pr_1(T(F^{(n)})) = T(F^{(n-1)})$). It is straightforward to obtain from the deterministic $F^{(n)}$-automaton $\mathcal{A}_{\theta,(X_1,...,X_n)}$ that recognizes $L_{\theta,(X_1,...,X_n)}$, a nondeterministic $F^{(n-1)}$-automaton $\mathcal{A}$ 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 $\mathcal{A}$, which gives $\mathcal{A}_{\varphi,(X_1,...,X_{n-1})}$. This determinization step increases (from N to at most 2^N) the number of states of $\mathcal{A}$.

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 $\mathcal{A}_{\varphi,(X_1,...,X_6)}$ has to accept the terms that contain one and only one occurrence of a constant symbol of the form $(\mathbf{a}, w)$ where the third component of w is 1 (here $w \in \{0, 1\}^6$).

> *Convention*: If a state called *Error* is present, 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 $\mathcal{A} := \mathcal{A}_{edg_q(X_1,X_2)}$ for the signature $F := F_C^{\mathrm{VRd}}$ and $q \geq 1$. Its set of states is :

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

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 $t * (V_1, V_2)$ in $T(F^{(2)})$, we have:

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

The notation $t' \in L(\mathcal{A}, s)$ means that the unique run of $\mathcal{A}$ 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: $\oplus[ab(0), a(2)] \to Error$. 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	$V_1 = V_2 = \emptyset$
$1(a)$	$V_1 = \{v\}, V_2 = \emptyset, port_{cval(t)}(v) = a$
$2(a)$	$V_1 = \emptyset, V_2 = \{v\}, port_{cval(t)}(v) = a$
$a(i)$	$V_1 = V_2 = \{v\}$, $port_{cval(t)}(v) = a$, $edg_{icval(t)}(v, v)$
$ab(i)$	$V_1 = \{v_1\}, V_2 = \{v_2\}$, $v_1 \neq v_2$, $port_{cval(t)}(v_1) = a$, $port_{cval(t)}(v_2) = b$ and $edg_{icval(t)}(v_1, v_2)$
$Error$	All other cases

Table 1: Meanings of the states of $\mathcal{A}$.

Transition rules	Conditions
$\varnothing \to 0$ $(\mathbf{a}, 00) \to 0$ $(\mathbf{a}, 10) \to 1(a)$ $(\mathbf{a}, 01) \to 2(a)$ $(\mathbf{a}, 11) \to a(0)$	
$relab_h[0] \to 0$ $relab_h[i(a)] \to i(c)$ $relab_h[a(j)] \to c(j)$ $relab_h[ab(j)] \to cd(j)$	 $c = h(a), d = h(b)$, $i \in [2], j \in [0, q]$
$add_a^{loop}[s] \to s$	$s \notin \{a(0), \cdots, a(q)\}$
$add_a^{loop}[a(i)] \to a(i+1)$ $add_a^{loop}[a(q)] \to Error$	$i < q$
$\overrightarrow{add}_{a,b}[s] \to s$	$s \notin \{ab(0), \cdots, ab(q)\}$
$\overrightarrow{add}_{a,b}[ab(i)] \to ab(i+1)$ $\overrightarrow{add}_{a,b}[ab(q)] \to Error$	$i < q$
$\oplus[1(a), 2(b)] \to ab(0)$ $\oplus[2(b), 1(a)] \to ab(0)$ $\oplus[s, 0] \to s$ $\oplus[0, s] \to s$	 (possibly $a = b$) $s \in S$

Table 2: The transition rules of $\mathcal{A}$.

The states $aa(0)$ could be identified with $Error$ because, as one can check, no run including such a state can reach an accepting state. This shows that the automaton $\mathcal{A}$ is not minimal. However, keeping these states yields a more uniform description of the transitions.

If, in this automaton, we replace the transitions $add_a^{loop}[a(q)] \to Error$ and $\overrightarrow{add}_{a,b}[ab(q)] \to Error$ by $add_a^{loop}[a(q)] \to a(q)$ and $\overrightarrow{add}_{a,b}[ab(q)] \to ab(q)$ respectively, then we obtain an automaton $\mathcal{A}_1$ 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 $r \geq q$. Furthermore, the automata $\mathcal{A}$ and $\mathcal{A}_1$ with accepting states $ab(p)$ and $a(p)$ for $0 < p < q$ recognize the language $L_{edg_p(X_1,X_2)}$.

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

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

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

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

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

$$V_1 = V_2 = \{v\},\ port_{cval(t)}(v) = a,\ (v, v) \in edg_{i,qcval(t)},$$

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

$$\overrightarrow{add}_{a,b}[ab(i)] \to ab(\mathrm{mod}_q(i)) \text{ and } add_a^{loop}[a(i)] \to a(\mathrm{mod}_q(i))$$

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 [CouDur] and in [Cou], Chapter 6).

We fix q and the set of port labels C. The states of the automaton $\mathcal{B}_p$ equivalent to $\mathcal{A}_{\mu_{p,q}(X_1,X_2)}$ 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 $V_1 - V_2$,

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

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

$A_3 = port_{cval(t)}(V_1 \cap V_2)$ and $f_3(a) = \mathrm{mod}_q(\mid port_G^{-1}(a) \cap V_2 \cap V_1 \mid)$ 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 $k = \mid C \mid$. The transitions are easy to define from the above specifications. Let us describe some cases:

i) For a constant symbol, we have for an example $(\mathbf{a}, 11) \longrightarrow (\emptyset, \emptyset, \{(a, 1)\}, 0)$ (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:

$$\oplus[(A_1, f_1, A_2, f_2, A_3, f_3, , j), (A'_1, f'_1, A'_2, f'_2, A'_3, f'_3, , j')] \longrightarrow$$
$$(A_1 \cup A'_1, f_1 + f'_1, A_2 \cup A'_2, f_2 + f'_2, A_3 \cup A'_3, f_3 + f'_3, , j + j')$$

where additions are modulo q (hence $(f_1 + f'_1)(a) := \mathrm{mod}\, q(f_1(a) + f'_1(a))$).

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 such that $f'_1(a)$ 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 $V_1 \cap V_2$. 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 $\overrightarrow{add}_{a,b}$. The transition must updates the number of edges from V_1 to V_2 that are added by this operation. These edges are from $V_1 - V_2$ to $V_2 - V_1$, from $V_1 - V_2$ to $V_1 \cap V_2$ and from $V_1 \cap V_2$ to $V_2 - V_1$, and these cases are mutually exclusive. It follows that j becomes $\mathrm{mod}_q(j + f_1(a).f_2(b) + f_1(a).f_3(b) + f_3(a).f_2(b))$.□

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.

5 Special tree-width

We define special tree-width by means of terms over the signatures F^{VRd} and F^{VRu}. An equivalent definition by means of tree-decompositions will be given later.

Definition 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 $t \in T(F^{\mathrm{VRd}}) \cup T(F^{\mathrm{VRu}})$, then $\pi(t)$ denotes $\pi(val(t))$ and $\pi_1(t)$ denotes $\pi_1(val(t))$. A term t in $T(F^{\mathrm{VRd}}) \cup T(F^{\mathrm{VRu}})$ is a *special VR-term* if it satisfies the following conditions:

1) $\pi(t') - \pi_1(t') \subseteq \{\bot\}$ for every subterm t' of t (we consider t as one of its subterms),

2) if $t_1 \oplus t_2$ is a subterm of t, then $\pi(t_1) \cap \pi(t_2) \subseteq \{\bot\}$,

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

4) for every operation $\overrightarrow{add}_{a,b}$, $add_{a,b}$, add_a^{loop} that occurs in t, we have $a \neq \bot$ and $b \neq \bot$,

5) the constant symbol $\bot$ has no occurrence in t.

If C is a finite set of port labels, we denote by $SpT(F_C^{\mathrm{VRd}})$ and $SpT(F_C^{\mathrm{VRu}})$ the sets of special VR-terms in $T(F_C^{\mathrm{VRd}})$ and in $T(F_C^{\mathrm{VRu}})$ 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 $SpT(F_C^{\mathrm{VRd}}) \cup SpT(F_C^{\mathrm{VRu}})$ such that $\mid C - \{\bot\} \mid = k+1$. Since we identify a graph with a p-graph whose vertices are labelled by $\bot$, the set C must always contain $\bot$, except if G is the empty graph. The comparison with tree-width will justify the "+1" in the definition. 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 $SpT(F_C^{\mathrm{VRd}})$ and $SpT(F_C^{\mathrm{VRu}})$ are regular.

Example 16 : Trees

Trees have special tree-width 1. To prove this, we let $C := \{\bot, 1, 2\}$. 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 $\bot$. Let T_1, T_2 be two such trees, defined by terms $t_1, t_2 \in SpT(F_C^{\mathrm{VRu}})$. Then, we let $T := T_1 \ltimes T_2$ be defined by the term

$$t := relab_{2 \longrightarrow \bot}\ (add_{1,2}(t_1 \oplus relab_{1 \longrightarrow 2}(t_2))) \in SpT(F_C^{\mathrm{VRu}}).$$

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 defined as that of T_1. Every rooted and undirected tree is generated by $\ltimes$ from the trees reduced to isolated roots, that are defined (up to isomorphism) by the constant symbol **1**. Hence, every rooted and undirected tree is defined by a term in $SpT(F_C^{\mathrm{VRu}})$. One can forget the root by applying the operation $relab_{1 \longrightarrow \bot}$.□

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 $x \leq_T y$ if and only if y is on the directed path from the root to x.

Definition 17 : Special tree-decompositions.
A tree-decomposition (T, f) of a graph G is *special* if it satisfies the following condition, in addition to the three conditions of Definition 6:

4) For each vertex x, the set $f^{-1}(x)$ 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 $SpT(F_C^{\mathrm{VRd}}) \cup SpT(F_C^{\mathrm{VRu}})$ into a special tree-decomposition of width $\mid C - \{\bot\} \mid -1$ of the graph $val(t)$ and vice-versa.

Proof: *From terms to decompositions.* We will define for every term t in $SpT(F_C^{\mathrm{VRd}}) \cup SpT(F_C^{\mathrm{VRu}})$ a special tree-decomposition $S(t)$ of the graph $G := cval(t)$, the boxes of which have at most $\mid C - \{\bot\} \mid$ vertices. The proof is by induction on the structure of t.

For every t, we will define $S(t)$ so that its root box consists of the vertices of G that are not $\bot$-ports. By the definition of special terms, each element of $C - \{\bot\}$ labels at most one vertex, hence the root box has at most $\mid C - \{\bot\} \mid$ vertices.

If $t = \mathbf{a}$, 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 $t = relab_h(t_1)$ 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 define f as the extension of f_1 such that $f(r)$ is the set of vertices of $G := val(t)$ that are not $\perp$-ports. By the definition of a special VR-term, we have $f(r) \subseteq f_1(r_1)$. We obtain a special tree-decomposition $S(t) := (T, f)$ of G.

If $t = t_1 \oplus t_2$, 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 define 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 $\perp$-ports and $S(t) := (T, f)$ is a special tree-decomposition of G with the required property.

Since each box of $S(t)$ has been 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 $\mid C - \{\perp\} \mid -1$.

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 finite set of port labels that contains $\perp$. Let (T, f) be a tree-decomposition of a graph G and $\gamma : V_G \to C - \{\perp\}$ 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 $N_{T/u} = \{w \in N_T \mid w \leq_T u\}$. Its root is u.

We denote by $G(u)$ the p-graph $(G(u)^\circ, port_{G(u)})$ where $G(u)^\circ$ 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 $port_{G(u)}(x) := \perp$ if $x \in V_{G(u)} - f(u)$. Hence, $G(u)$ is a p-graph such that $\pi(G(u)) - \pi_1(G(u)) \subseteq \{\perp\}$. We have $G = G(root_T)^\circ$.

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

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

Every injective mapping $\delta : f(u) \to C - \{\perp\}$ can be extended into a mapping $\gamma : V_{G(u)} \to C - \{\perp\}$ 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 find an injective mapping $\delta_i : f(u_i) \to C - \{\perp\}$ that coincides with δ on $f(u_i) \cap f(u)$. By the induction hypothesis, it can be extended into γ_i defined 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 Definition 6), and so $\gamma_i(x) = \gamma_j(x) = \delta(x)$.

It is routine work to construct a linear algorithm computing γ.$\square$

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 define the p-graphs $G(u)$ (their port labels depend on γ) so that: $G = relab_{C \longrightarrow \perp}(G(root_T))$ (where for every subset B of C, we let $relab_{B \longrightarrow \perp}$ denote the composition, in any order, of the operations $relab_{b \longrightarrow \perp}$ for all $b \in B - \{\perp\}$).

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

$$G(u) = ADD(relab_{B_1 \longrightarrow \perp}(G(u_1)) \oplus \cdots \oplus relab_{B_p \longrightarrow \perp}(G(u_p)) \oplus \mathbf{a}_1 \oplus ... \oplus \mathbf{a}_s)$$

where $\{a_1, ..., a_s\} := \pi(G(u)) - (\{\perp\} \cup \pi(G(u_1)) \cup ... \cup \pi(G(u_p))$, $B_i := \{\gamma(x) \mid x \in f(u_i) - f(u)\}$ 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 $\pi_1(relab_{B_i \longrightarrow \perp}(G(u_i)))$ are pairwise disjoint. Hence, we can define:

$$t(u) := ADD(relab_{B_1 \longrightarrow \perp}(t(u_1)) \oplus \cdots \oplus relab_{B_p \longrightarrow \perp}(t(u_p)) \oplus \mathbf{a}_1 \oplus ... \oplus \mathbf{a}_s),$$

$t(u)$ belongs to $SpT(F_C^{\mathrm{VRd}}) \cup SpT(F_C^{\mathrm{VRu}})$ and defines $G(u)$.

The term $relab_{C \longrightarrow \perp}(t(root_T))$ defines G.

This construction can be done by a linear time algorithm, where the size of the input is $\mid V_G \cup E_G \cup N_T \mid$.$\square$

Proposition 19 : For every graph G we have:
(1) $twd(G) \leq sptwd(G) \leq pwd(G)$.
(2) $cwd(G) \leq sptwd(G) + 2$.

Proof: The first assertion follows from Proposition 18, and the second one from Definition 15.$\square$

Note the difference between Proposition 7 and Assertion (2).

We will denote by $STWD(\leq k)$ 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. (We exclude the case of a vertex incident with a loop and with no other edge).

Proposition 20 : For each k, the class $STWD(\leq k)$ 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 $\leq_T$. We let similarly v be the minimal element of $f^{-1}(z) \cap f^{-1}(y)$. Since u and v are on the directed path $f^{-1}(y)$, they are comparable, say, $u \leq_T v$. 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' such that $V_{G'} = V_G - \{y\}$. Then, we define f' by:

$f'(w) := (f(w) - \{y\}) \cup \{x\}$ for every w on the path in T from v to u (including u and v),

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

It is easy to see that (T, f') is a special tree-decomposition of G. In particular, x and z belong both to $f'(v)$. The set $f'^{-1}(x)$ 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') is a special tree-decomposition of G and its width that is no larger than that of (T, f), which shows that $sptwd(G') \leq sptwd(G)$.□

It follows from items 1) and 5) of this proposition that the class $STWD(\leq k)$ is closed under taking *topological minors* ([Die]). It is not closed under taking minors as we will see in Proposition 25 below. 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:

$$pwd(L) < \infty \Longrightarrow sptwd(L) < \infty \Longrightarrow twd(L) < \infty \text{ and}$$

$$sptwd(L) < \infty \Longrightarrow cwd(L) < \infty,$$

whereas the converse implications do not hold.□

Proof : We will use the following claim.

Claim : For every graph G, the special tree-width of $G \otimes *$ is equal to its path-width.

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

For each vertex x of G, the directed paths $f^{-1}(*) = P$ and $f^{-1}(x)$ have a nonempty intersection, hence $x \in f(u)$ for some u in P, and every edge linking $*$ and x has its two ends in $f(u)$.

If y is another vertex of G that is adjacent to x, then $P \cap f^{-1}(y)$ is not empty and contains some node v. Let u and v be the minimal such nodes with respect to $\leq$. 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 $f^{-1}(x) \cap f^{-1}(y)$ is not empty, it must contain v, and the edges between x and y have their ends in $f(v)$. The pair $(P, f \restriction P)$ satisfies also the connectivity condition, hence it is a path-decomposition of $G \otimes *$ of no larger width than (T, f). Since we have $sptwd(G \otimes *) \leq pwd(G \otimes *)$ by Proposition 19, we have an equality.□

For proving the proposition, we assume that every graph of tree-width 2 has special tree-width at most k. If T is any tree, then $T \otimes *$ 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 $T \otimes *$, has path-width at most k, but trees have unbounded path-width (see [Die]), 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 $sptwd(L) < \infty \Longrightarrow cwd(L) < \infty$ is false if L the set of cliques, of maximal clique-width 2 and of unbounded tree-width and special tree-width. □

Definition 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 $f : N_T \longrightarrow \mathcal{P}(V_G)$ 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 defined as the maximal cardinality of a box, (no -1 here !), and the *tree-partition-width* of a graph G is the minimal width of its tree-partitions. We denote it by $tpwd(G)$. We will prove that $sptwd(G) \leq 2.tpwd(G) - 1$. The wheels, i.e. the graphs $C_n \otimes *$ where C_n is the undirected cycle with n vertices have path-width (and special tree-width) 3 but unbounded

tree-partition width (see [BodEng], [Wood]). $MaxDeg(G)$ denotes the maximum degree of a graph G.

Proposition 23: For every graph G :

1) $sptwd(G) \leq 2.tpwd(G) - 1$,

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

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:

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

where $Core(G)$ is the simple, loop-free 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', f') of G such that $N_T = N_{T'}$ and $f(u) \subseteq f'(u)$ for every $u \in N_T$. We choose an arbitrary linear order $\leq$ on N_T and we let T' 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' 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' is a tree with root $root_T$. The root has no right son. For every $u \in N_T = N_{T'}$, we define:

$f'(u) := f(u)$ if $u = root_T$,

$f'(u) := f(u) \cup f(w)$ if w is the father of u in T.

It is straightforward to verify that (T', f') 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 $\mathbf{A}, \mathbf{B}, \mathbf{C}, ...$ represent pairwise disjoint sets of vertices), and, to the right, the corresponding special tree-decomposition. (The box of the node $\mathbf{XY}$ is $\mathbf{X} \cup \mathbf{Y}$). Unless T has $root_T$ as single node, it can be delete from T'.

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

This result suggests a question:

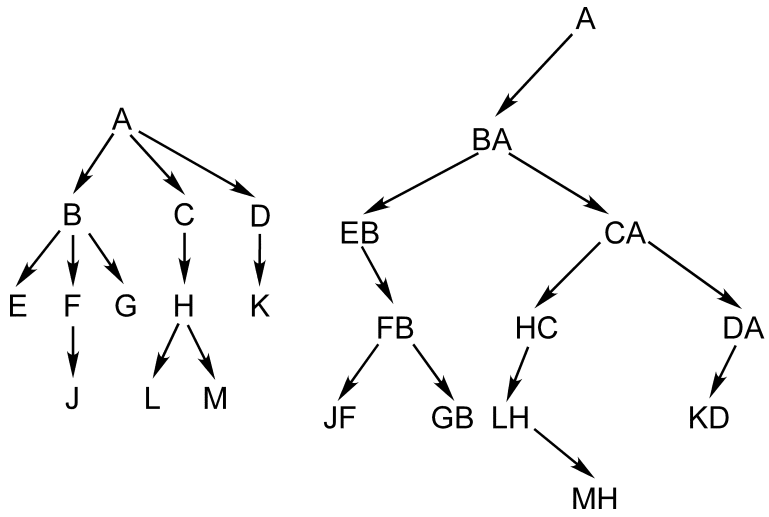

Figure 1: A tree-partition and the associated tree-decomposition

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 fixed 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 [Cou], Chapter 9).

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 first 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 $SPTWD(\leq k)$ 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.

Proof: We first consider the assertion about biconnected components. Let G, H and H' be concrete graphs. We write $G = H \cup_x H'$ if $V_H \cap V_{H'} = \{x\}$, $E_H \cap E_{H'} = \emptyset$, and G is the union of H and H'. If (T, f) and (T', f') are special tree-decompositions of H and H' such that x is in the root box of (T', f'), then, we construct as follows a special tree-decomposition (T'', f'') of G:

> we let u be the least node of T (least with respect to $\leq_T$) such that $x \in f(u)$,
>
> T'' is obtained from the union of T and T' (assumed to be disjoint) augmented with a directed edge from u to $root_{T'}$ (so that $root_{T''} = root_T$),
>
> f'' is $f \cup f'$, the common extension of f and f'.

Its width is the maximum of those of (T, f) and (T', f').

Let G be a connected graph with biconnected components $H_1, ..., H_p$, for which we have special tree-decompositions of width at most k. We can assume that $H_1, ..., H_p$ are numbered in such a way that, for each $1 < i \leq p$, the graph $G_i := H_1 \cup ... \cup H_i$ satisfies $G_i = G_{i-1} \cup_x H_i$ for some vertex x. By induction on i, we construct as follows a special tree-decomposition of G_i of width at most $k+1$. For $i = 1$, we have it by the hypothesis. For $i > 1$ and $G_i = G_{i-1} \cup_x H_i$, we modify if necessary the decomposition of H_i so that x is in its root box (if this is not the case, we add x to all boxes above the ones that contain x). By induction we have a special tree-decomposition of G_{i-1} of width at most $k+1$, and by using the first construction, we obtain a special tree-decomposition of G_i of width at most $k+1$. Since $G = G_p$, we have the result.

If G is the union of disjoint graphs H and H' and (T, f) and (T', f') are special tree-decompositions of H and H', then, we construct a special tree-decomposition (T'', f'') of G as above, letting u be any node of T. Its width is the maximum of those of (T, f) and (T', f'). Hence, if G is a union of connected components $H_1, ..., H_p$ for which we have special tree-decompositions of maximum width k, we construct as above a special tree-decomposition of G of width k. By Proposition 20(1), $sptwd(G) \geq sptwd(H_i)$ for any i, which proves the first assertion.

We now define 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 $\{a, b, c, d\}$ and edges $a \longrightarrow b, b \longrightarrow c$ and $b \longrightarrow d$. We let $V_H := [13]$ and f be the mapping: $N_T \longrightarrow \mathcal{P}(V_H)$ such that $f(a) := \{1, 2, 4, 5, 6\}$, $f(b) := \{3, 4, 5, 6, 7\}$, $f(c) := \{3, 4, 8, 9, 10\}$ and $f(d) := \{6, 7, 11, 12, 13\}$.We define 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

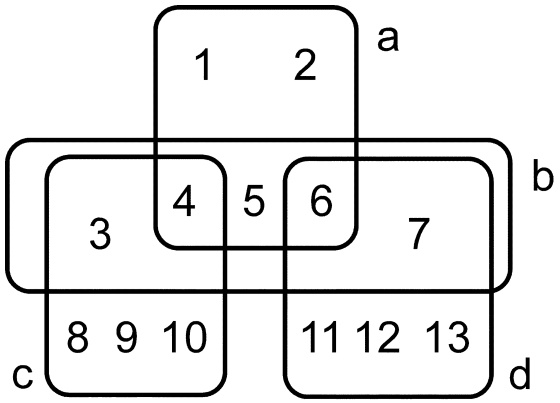

Figure 2: The special tree-decomposition (T, f) of H.

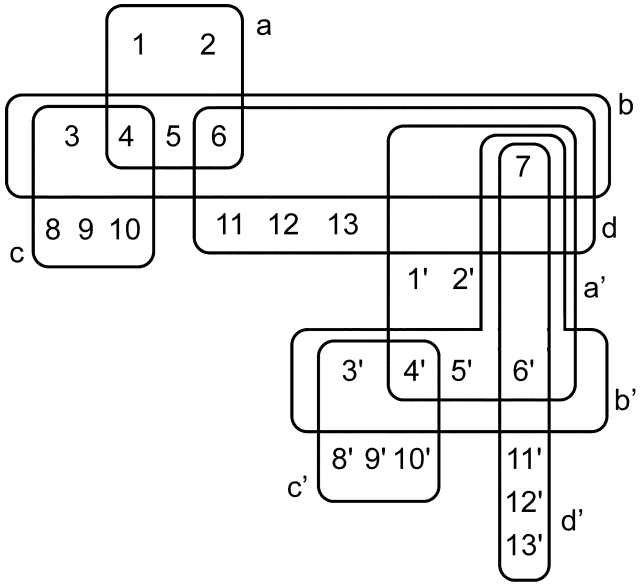

Figure 3: The special tree-decomposition (T'', f'') of G.

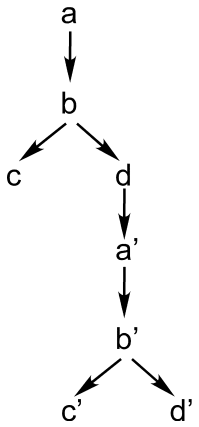

Figure 4: The tree T''

tree-width 4 and also special tree-width 4: the pair (T, f) is a special tree-decomosition of H (see Figure 2).

For every tree-decomposition of any graph, a clique in this graph is contained in some box. 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 the 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' be the isomorphic copy of H where each vertex i is made into i'. We let G be obtained from the union of H and H' by the fusion of vertices 7 and $7'$ (let us say that we delete $7'$ and we connect 7 with the neighbours of $7'$ in H'). Let (T', f') be the corresponding "isomorphic" special tree-decomposition of H'.

The biconnected components of G are H and H' hence, G has tree-width 4. It has special tree-width at most 5: Figure 3 shows the special tree-decomposition (T'', f'') of G arising from the construction of the beginning of the proof. Note that the box $f''(a')$ contains vertex 7 hence has 6 elements.

Assume that G has a special tree-decomposition (T_2, f_2) of width 4. It must have nodes $b_2, d_2, a'_2, b'_2, d'_2$ such that $f_2(b_2) = f(b), f_2(d_2) = f(d)$, $f_2(a'_2) = f'(a') = \{1', 2', 4', 5', 6'\}$, $f_2(b'_2) = \{3', 4', 5', 6', 7\}$ and $f_2(d_2) := \{6', 7, 11', 12', 13'\}$. 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 and d'_2. By the observation made above for H (which applies also to H'), we see that we must have

b_2 before d_2 and b'_2 before d'_2. But then we must also have a'_2 on this path. We cannot do that without having $7 \in f_2(a'_2)$. Hence, (T_2, f_2) cannot exist and G has special tree-width 5.□

Open question: The parsing problem.

Does there exist fixed 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 fixed c instead of $O(n^{g(k)})$. Since by a result by Bodlaender (presented in detail in [DF]) such an algorithm exists for tree-width, with $f(k) = k$ and $c = 1$, one can think that this algorithm can be adapted in order to find special tree-decompositions.

6 Finite automata for monadic second-order formulas with edge set quantifications

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

Definition 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 quantifications (and set cardinality predicates), we will represent a graph G by the relational structure $\lceil G \rceil := Inc(G)$ defined in Definition 9. If G is undirected, then $\lceil G \rceil := \langle V_G \cup E_G, in_G \rangle$ where in_G is the set of pairs (e, x) such that $e \in E_G$ and x is an end vertex of e. If G is directed, $\lceil G \rceil := \langle V_G \cup E_G, in_{1G}, in_{2G} \rangle$ 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).

As in the proof of Theorem 13, we will use formulas with a particular "normalized" syntax. They will be written without first-order variables and universal quantifications, 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 $\exists X_n.\theta$, the formula θ has no free variables in $\{X_{n+1}, ...\}$, and similarly for $\exists Y_m.\theta$. 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, $X_i \subseteq X_j$, $Y_i \subseteq Y_j$, $Z = \emptyset$, $Sgl(Z)$, $Card_{p,q}(Z)$ where Z is X_i or Y_j. Their meanings, if not already defined are as follows for a graph G:

$in(Y_i, X_j)$ means that Y_i and X_j are singletons, respectively $\{y\}$ and $\{x\}$, and that $(y, x) \in in_G$,

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 $cval(t)$ (cf. Definition 2). Its vertices are the elements of $Occ_0(t)$ (they are leaves of t). Its edges are pairs $(u, (x, y))$, $(u, \{x, y\})$ or $(u, \{x\})$ where u is a useful occurrence (cf. Proposition 5) of an edge addition operation f. 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 $Occ_1(t)$.

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

We will use the projections pr_s as in Theorem 13 and the projections pr'_s, that delete the last s Booleans in the unary operations (f, w). It is clear that a term $t * \gamma \in T(F_C^{\mathrm{VRd}(n,m)})$ such that t is a special VR-term and the occurrences of edge addition operations in t are all useful, defines a concrete graph $cval(t)$ and an $\{X_1, ..., X_n, Y_1, ..., Y_m\}$-assignment γ such that $\gamma(X_i)$ is a set of vertices (for $i \in [n]$) and $\gamma(Y_j)$ is a set of edges (for $j \in [m]$).

However, the terms t are not any terms in $T(F_C^{\mathrm{VRd}(n,m)})$. We will denote by $RT(F_C^{\mathrm{VRd}(n,m)}) \subseteq T(F_C^{\mathrm{VRd}(n,m)})$ the set of *reduced terms*, defined as the set of special VR-terms in which every occurrence of an edge addition operation is useful. (If a special VR-term is not reduced, it can be transformed into a smaller equivalent reduced term by deletions of the edge addition operations that are not useful). Whether t in $T(F_C^{\mathrm{VRd}(n,m)})$ is reduced or not does not depend on the Boolean components of its constant symbols and unary edge addition operations. In other words, $RT(F_C^{\mathrm{VRd}(n,m)}) = pr'^{-1}_m(pr_n^{-1}(RT(F_C^{\mathrm{VRd}})))$.

Let us sketch the construction of an F_C^{VRd}-automaton $\mathcal{R}$ that recognizes $RT(F_C^{\mathrm{VRd}})$. Its set of states is $\{A \mid A \subseteq C - \{\bot\}\} \cup \{Error\}$ with the following characteristic properties, expressed as in Table 1, for a term t in $T(F_C^{\mathrm{VRd}})$:

$P_A \Leftrightarrow t \in RT(F_C^{\mathrm{VRd}})$ and $A = \pi_1(t)$,

$P_{Error} \Leftrightarrow t \notin RT(F_C^{\mathrm{VRd}})$.

The transition rules are in Table 3.

Transition rules	Conditions
$\varnothing \to \emptyset$ $\mathbf{a} \to \{a\}$	
$relab_h[A] \to h(A) - \{\perp\}$	if $h(a) = h(b)$ with $a \neq b$, then $h(a) = \perp$
$add_a^{loop}[A] \to A$	$a \in A$
$\overrightarrow{add}_{a,b}[A] \to A$ $\oplus[A, B] \to A \cup B$	$a, b \in A$ $A \cap B = \emptyset$

Table 3: The transition rules of $\mathcal{R}$.

All states except $Error$ are accepting. By replacing in this table every edge addition operation f by the unary operations (f, w) for $w \in \{0,1\}^m$ and every constant symbol $\mathbf{a}$ by $(\mathbf{a}, w)$ for $w \in \{0,1\}^n$, we obtain an automaton with the same set of states that recognizes $RT(F_C^{\mathrm{VRd}(n,m)})$. Similar constructions can be done for $RT(F_C^{\mathrm{VRu}(n,m)})$.

If F is a finite subsignature of F^{VRd} or of F^{VRu}, then it is a subsignature of F_C^{VRd} or of F_C^{VRu} for some finite set C, and $F^{(n,m)}$ denotes the corresponding subsignature of $F_C^{\mathrm{VRd}(n,m)}$ or of $F_C^{\mathrm{VRu}(n,m)}$. Let us fix such F (to simplify notation). For every CMS_2 formula φ with free variables in $\{X_1, ..., X_n, Y_1, ..., Y_m\}$ written with in if it concerns undirected graphs, or in_1 and in_2 if it concerns directed graphs, we define $L_{\varphi,(X_1,...,X_n,Y_1,...,Y_m)}$ as the set $\{t * \gamma \in RT(F^{(n,m)}) \mid (\lceil cval(t) \rceil, \gamma) \models \varphi\}$. The language $L_{P(X_1,...,X_n,Y_1,...,Y_m)}$ can be defined similarly for a graph property P independently of its logical expression.

Theorem 27: Let F be a finite subsignature of F^{VRd} or of F^{VRu}. For every CMS_2 graph property $P(X_1, ..., X_n, Y_1, ..., Y_m)$, the language $L_{P(X_1,...,X_n,Y_1,...,Y_m)}$ is regular and an F-automaton recognizing it can be constructed from a CMS_2 formula that defines P.

Proof: As for proving Theorem 13, we will construct by induction on the structure of φ an F-automaton $\mathcal{A}_{\varphi,(X_1,...,X_n,1,...,Y_m)}$ that recognizes the language $L_{\varphi,(X_1,...,X_n,Y_1,...,Y_m)}$.

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

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

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

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

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

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

$$L_{\varphi,(X_1,\ldots,X_n,Y_1,\ldots,Y_{m-1})} = pr'_1(L_{\theta,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}).$$

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 $\mathcal{A}'$ for $edg(X_1, X_2)$ is close to the automaton $\mathcal{A}$ of Theorem 13. Its set of states is (we name the states as for $\mathcal{A}$):

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

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 $k = |C-\{\bot\}|$. The transition rules are in Table 4. The missing transitions yield $Error$. Here is an example: $relab_{a\to\bot}[ab(0)] \to Error$. The unique accepting state is Ok.

Transition rules	Conditions
$\varnothing \to 0$ $(\mathbf{a}, 00) \to 0$ $(\mathbf{a}, 10) \to 1(a)$ $(\mathbf{a}, 01) \to 2(a)$ $(\mathbf{a}, 11) \to a(0)$	
$relab_h[0] \to 0$ $relab_h[i(a)] \to i(c)$ $relab_h[a(0)] \to c(0)$ $relab_h[ab(0)] \to cd(0)$	 $i \in [2]$ $c = h(a) \neq \bot, d = h(b) \neq \bot,$
$add_a^{loop}[s] \to s$ $add_a^{loop}[a(0)] \to Ok$	$s \neq a(0)$
$\overrightarrow{add}_{a,b}[s] \to s$ $\overrightarrow{add}_{a,b}[ab(0)] \to Ok$	$s \neq ab(0)$
$\oplus[1(a), 2(b)] \to ab(0)$ $\oplus[2(b), 1(a)] \to ab(0)$ $\oplus[s, 0] \to s$ $\oplus[0, s] \to s$	 (possibly $a = b$) all s

Table 4: The transition rules of $\mathcal{A}'$.

However, the automaton $\mathcal{A}'$ has been constructed to work correctly on reduced terms, not on all terms. The automaton $\mathcal{A}_{edg(X_1,X_2)}$ is then obtained by a product with the one that recognizes reduced terms, so that it recognizes $L(\mathcal{A}') \cap RT(F_C^{\mathrm{VRd}(2,0)})$. Its number of states is thus $2^k.O(k^2)$ instead of $k^2 + 3(k + 1)$. In the following remark, we will overcome this difficulty.

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

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

Their meanings are described in Table 5, where W_1 denotes the value of Y_1. As examples of *Error* transitions we have:

$\oplus[Ok, 1(a)] \to Error$, $(add_a^{loop}, 1)[1(b)] \to Error$ if $b \neq a$, and
$(\overrightarrow{add}_{a,b}, 1)[Ok] \to Error$.

The unique accepting state is Ok.

State s	Property P_s
0	$V_1 = W_1 = \emptyset$
$1(a)$	$V_1 = \{v\}, W_1 = \emptyset, port_{cval(t)}(v) = a$
Ok	$V_1 = \{v\}$, $W_1 = \{e\}$, $in_{1cval(t)}(e, v)$
Error	All other cases

Table 5: Meanings of the states of $\mathcal{B}$.

Transition rules	Conditions
$\varnothing \to 0$ $(\mathbf{a}, 0) \to 0$ $(\mathbf{a}, 1) \to 1(a)$	
$relab_h[0] \to 0$ $relab_h[Ok] \to Ok$ $relab_h[1(a)] \to 1(b)$	 $b = h(a) \neq \bot$
$(add_a^{loop}, 0)[s] \to s$ $(add_a^{loop}, 1)[1(a)] \to Ok$	all s
$(\overrightarrow{add}_{a,b}, 0)[s] \to s$ $(\overrightarrow{add}_{a,b}, 1)[1(a)] \to Ok$	all s
$\oplus[s, 0] \to s$ $\oplus[0, s] \to s$	all s

Table 6: The transition rules of $\mathcal{B}$.

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 $\mathcal{A}_{\theta,(X_1,...,X_n,Y_1,...,Y_m)}$ that recognizes only reduced terms. This means that each of these automata repeats the verification that the input term is reduced. One can actually postpone this verification to the very end.

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

$$L_{\alpha,(X_1,...,X_n,Y_1,...,Y_m)} = L(\mathcal{B}_{\alpha,(X_1,...,X_n,Y_1,...,Y_m)}) \cap RT(F^{(n,m)}).$$

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

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

$$L(\mathcal{B}_{\neg\varphi_1,(X_1,...,X_n,Y_1,...,Y_m)}) = T(F^{(n,m)}) - L(\mathcal{B}_{\varphi_1,(X_1,...,X_n,Y_1,...,Y_m)}).$$

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

$$L(\mathcal{A}_{\varphi,(X_1,...,X_n,Y_1,...,Y_m)}) = L(\mathcal{B}_{\varphi,(X_1,...,X_n,Y_1,...,Y_m)}) \cap RT(F^{(n,m)}).$$

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

$$(L \cap R) \cap (M \cap R) = ((L \cap M) \cap R),$$
$$(L \cap R) \cup (M \cap R) = ((L \cup M) \cap R),$$
$$R - (L \cap R) = (T - L) \cap R,$$
$$pr(L' \cap R') \cap R = pr(L') \cap R,$$

where $L, M, R, ...$ are sets such that $L, M, R \subseteq T$ and $L', R' \subseteq T'$, and pr is a mapping from T' to T such that $T' = pr^{-1}(T)$ and $R' = pr^{-1}(R)$.□

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

Definition 29 : **Special HR-terms.**

We let F^{HRd} be the signature obtained from F^{VRd} by replacing the operation $\oplus$ by $//$. This operation symbol will be interpreted as follows: for directed p-graphs G and H such that, as in Definition 15, $\pi(G) - \pi_1(G) \subseteq \{\bot\}$ and $\pi(H) - \pi_1(H) \subseteq \{\bot\}$, we let $G//H$ be obtained from $G \oplus H$ by the fusion of any two vertices having the same port label $a \neq \bot$. An *HR-term* is a term t in $T(F^{\mathrm{HRd}})$ such that:

1) $\pi(t') - \pi_1(t') \subseteq \{\bot\}$ for every subterm t' of t,

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

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

4) the constant symbol $\bot$ has no occurrence in t.

We denote by $HT(F^{\mathrm{HRd}})$ the set of HR-terms. The notations F_C^{VRd} and $F_C^{\mathrm{VRd}(n,m)}$ extend in the obvious way, yielding sets like $HT(F_C^{\mathrm{HRd}(n,m)})$, that are, clearly, regular languages. These definitions also extend to undirected graphs, giving F^{VRu} , F_C^{VRu}, $F_C^{\mathrm{VRu}(n,m)}$ 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 $\mid C - \{\bot\} \mid -1$ such that this graph is the value of a term in $HT(F_C^{\mathrm{HRd}}) \cup HT(F_C^{\mathrm{HRu}})$. There are linear-time algorithms for converting a term t in $HT(F_C^{\mathrm{HRd}}) \cup HT(F_C^{\mathrm{HRu}})$ into a tree-decomposition of width $\mid C - \{\bot\} \mid -1$ 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 [Cou], Chapter 2 (with slightly different definitions).□

Let us go back to Definition 26, where we discuss the encoding of assignments in terms. Let t be an HR-term and G be the concrete graph $cval(t)$. Its edges are in bijection with $Occ_1(t)$, defined as for special VR-terms.

However, its vertex set is isomorphic to a quotient of $Occ_0(t)$, by the equivalence relation $\approx$ expressing that two leaves x and y in $Occ_0(t)$ have a least common ancestor u that is an occurrence of //, and that $port_t(x,u) = port_t(y,u) \neq \bot$. This implies that they are fused at some stage and yield the same vertex of G. Hence, we loose the nice bijection between vertices of $val(t)$ and particular occurrences of symbols in t. It follows that a set $X \subseteq Occ_0(t)$ represents correctly a set of vertices of $val(t)$ if and only it is saturated for $\approx$ (is a union of classes of this equivalence). The automata analogous to $\mathcal{B}_{\varphi,(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$ would have to check this saturation property, which would increase substancially their numbers of states.

There is actually another possibility for representing vertices in terms. Let us assume that $G = val(t)$ is a graph (and not a p-graph), hence that its vertices are all $\bot$-ports. This implies that each vertex corresponds to a unique occurrence of an operation $relab_{a \longrightarrow \bot}$. 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 difficulty for constructing an automaton for the atomic formulas $in_1(Y_1,X_1), in_1(Y_1,X_1)$ and $in_1(Y_1,X_1)$ like $\mathcal{B}$ in the proof of Theorem 27. These automata have also $k+3$ states (where $k = \mid C - \{\bot\} \mid$), 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 $\exists Y_1(in_1(Y_1,X_1) \wedge in_2(Y_1,X_2))$, 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 $\wedge$ of a product of two automata, and the exponentiation is due to the determinization that is needed because of $\exists Y_1$). However, it is proved in [Cou], Chapter 6 that every deterministic automaton for $edg(X_1,X_2)$ must have at least $2^{k(k-1)}$ states. 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 $edg(X_1, X_2)$, $in_1(Y_1, X_1), in_1(Y_1, X_1)$ and $in_1(Y_1, X_1)$ have, say, $O(k^2)$ states?

7 Conclusion

Optimizations for the constructions of automata are developped in [Cou], Chapter 6. However, these constructions are difficult if not impossible in practice, because of the sizes of the automata. This difficulty is not avoidable as proved by [Wey], [FriGro] and [StoMey]. It is not avoidable for general monadic second-order formulas, but even for basic graph properties like connectedness, the minimal $F_{[k]}^{\mathrm{VRu}}$-automaton has more than 2^{2^k} states ([Cou], Chapter 6). A more attractive possibility is to avoid "compiling" automata, but to compute (and recompute) the transitions that are needed for a particular input term. Such *fly-automata* are introduced and used in [CouDur].

The main problems left open about clique-width and special tree-width are the *parsing problems*. In short, they are the problems of approximating in polynomial time the clique-width of a graph with multiple edges and the special tree-width of a simple graph. These problems are presented in Definition 8 (Section 2) and at the end of Section 5.

8 References

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

[TATA] H. Comon *et al.*, *Tree Automata Techniques and Applications*, On line for free at: http://tata.gforge.inria.fr/

[CorRot] D. Corneil, U. Rotics, On the Relationship Between Clique-Width and Treewidth. *SIAM J. Comput.* **34** (2005) 825-847

[Cou] 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

[CouDur] B. Courcelle, I. Durand, Verifying monadic second-order graph properties with tree automata, *European Lisp Symposium*, May 2010, Lisbon, Proceedings to appear.

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

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

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

[DF] R. Downey et M. Fellows, *Parameterized complexity*, Springer-Verlag, 1999

[FRSS] M. Fellows, F. Rosamond, U. Rotics, S. Szeider: Clique-width minimization is NP-hard. *Proceedings of the 38th Annual ACM Symposium on Theory of Computing,* Seattle, USA, 2006, pp. 354-362.

[FG] J. Flum, M. Grohe, *Parametrized complexity theory*, Springer, 2006.

[FriGro] M. Frick, M. Grohe: The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic* **130** (2004) 3-31

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

[Lib] L. Libkin, *Elements of finite model theory*, Springer, 2004.

[HliOum] P. Hlineny, S. Oum: Finding Branch-Decompositions and Rank-Decompositions. *SIAM J. Comput.* **38** (2008) 1012-1032.

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

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

[Wey] M. Weyer, Decidability of S1S and S2S. in *Automata, Logics, and Infinite Games: A Guide to Current Research.* Lecture Notes in Computer Science **2500,** Springer, 2002, pp. 207-230

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