B. COURCELLE

The monadic second-order logic of graphs III : tree-decompositions, minors and complexity issues

Informatique théorique et applications, tome 26, nº 3 (1992), p. 257-286

<http://www.numdam.org/item?id=ITA_1992__26_3_257_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 26, n° 3, 1992, p. 257 à 286)

THE MONADIC SECOND-ORDER LOGIC OF GRAPHS III: TREE-DECOMPOSITIONS, MINORS AND COMPLEXITY ISSUES (*)

by B. COURCELLE [1]

Communicated by A. ARNOLD

Abstract. – *We relate the tree-decompositions of hypergraphs introduced by Robertson and Seymour to the finite and infinite algebraic expressions introduced by Bauderon and Courcelle. We express minor inclusion in monadic second-order logic, and we obtain grammatical characterizations of certain sets of graphs defined by excluded minors. We show how tree-decompositions can be used to construct quadratic algorithms deciding monadic second-order properties on hypergraphs of bounded tree-width.*

Résumé. – *On étudie les liens entre les décompositions arborescentes d'hypergraphes introduites par Robertson et Seymour et les expressions algébriques d'hypergraphes finies ou infinies de Bauderon et Courcelle. On exprime l'inclusion au sens des mineurs en logique monadique du second ordre, et on obtient des caractérisations grammaticales de certains ensembles de graphes définis par mineurs exclus. On utilise les décompositions arborescentes pour construire des algorithmes quadratiques qui décident les propriétés des hypergraphes de largeur arborescente bornée exprimables en logique monadique du second ordre.*

INTRODUCTION

This paper continues the study of graphs, hypergraphs and sets thereof using methods of formal language theory, universal algebra and logic, a study initiated in Bauderon and Courcelle [7] and Courcelle [13, 17].

In the present work, we show the relations between algebraic notions introduced in [7, 13, 17] and combinatorial notions introduced independently by Robertson and Seymour [25-30] in their study of graph minors.

(*) Received December 1990, revised February 1991.
This work has been supported by the "Programme de Recherches Coordonnées : Mathématiques et Informatique" and the ESPRIT-BRA projet 3299 on "Graph Transformations".
[1] Université Bordeaux-I, Laboratoire d'Informatique, Unité de Recherche Associée au Centre National de la Recherche Scientifique, 351, Cours de la Libération, 33405 Talence Cedex, France.

Operations on hypergraphs have been defined in Bauderon and Courcelle [7, 13, 17], that make it possible to denote finite hypergraphs by finite algebraic *expressions*, and infinite ones (countably infinite ones, to be precise) by infinite algebraic expressions (comparable to formal power series).

An expression is actually a tree and in a natural way defines a tree-structuring of the hypergraph it denotes. By a *tree-structuring*, we mean a hierarchical construction of the hypergraph in terms of smaller ones, and, recursively, of these latter ones in terms of others, etc. Expressions actually denote *sourced hypergraphs*, *i.e.*, hypergraphs equipped with a sequence of distinguished vertices. The length of this sequence is called the *type* of the hypergraph.

A notion of width of a hypergraph arises in a natural way: the *width* of an expression is the largest type of the hypergraph denoted by some subexpression, and the *width* of a hypergraph is the minimal width of an expression denoting it. (It is closely related to the *tree-width* introduced by Robertson and Seymour as we shall establish.)

In Section 2, we establish a correspondence between the tree-decompositions of hypergraphs introduced by Robertson and Seymour [25-27] and their tree-structurings defined by expressions. We show that the width of a hypergraph is linearly related to its *tree-width*, a notion that follows naturally from that of a tree-decomposition and is essential in the theory of Robertson and Seymour. Our proof works for finite as well as infinite hypergraphs, and the result for infinite hypergraphs is used in Courcelle [15].

The notion of a *hyperedge replacement grammar* (Habel, Kreowski [21, 22], Bauderon and Courcelle [7]) can be considered as an extension to graphs and hypergraphs of the notion of a context-free grammar defining words. Other types of graph grammars can be considered as *context-free* (*see* Courcelle [12]) but hyperedge replacement grammars are presently the most useful and well-studied ones. A set of hypergraphs will be called *context-free* iff it is generated by an HR (hyperedge replacement) grammar.

Every context-free set of hypergraphs is of bounded width [7], hence of bounded tree-width. Since the set of finite hypergraphs of tree-width at most k is context-free, it follows that every statement of the form: "for every set of hypergraphs of bounded tree-width..." also holds, "for every subset of a context-free set of hypergraphs" and vice-versa. There are many complexity results of this form. We refer the reader to Arnborg *et al.* [3] for an exhaustive list of results that subsumes many partial results obtained previously. (*See* Courcelle [17] and Van Leeuwen [33] for other references.)

In Section 2, we also define a mapping from hypergraphs to graphs that makes it possible to transfer several known results from graphs to hypergraphs.

Hypergraphs can also be considered as logical structures, and logical formulas can be used to write their properties in a formal syntax. *Monadic second-order logic* has proved to be quite powerful while having important decidability properties (Courcelle [13, 14, 17]). In particular the validity of a monadic second-order (*MS*) formula in every hypergraph of a context-free set is decidable. Another important result is that the validity of an *MS* formula in a hypergraph G, given with a derivation sequence d, relative to an *HR* grammar, is decidable in linear time in the length of d [7, 17]. The major difficulty is to construct d from G. Constructing a tree-decomposition of width $\leqq k$ is an attractive alternative but is still difficult for $k \geqq 4$. (*See* Section 3 for details.)

In Section 3, we show how some results of Robertson and Seymour [28, 29], appropriately extended to hypergraphs, make it possible to overcome this difficulty.

Section 4 is concerned only with graphs (as opposed to hypergraphs). We exploit the easy observation that if a set of graphs is characterized by finitely many excluded minors (this is the case of planar graphs by a variant of Kuratowski's theorem), then is definable by an *MS* formula.

By using other results of Robertson and Seymour [27] (whom we owe a lot for this paper) we obtain the existence of *HR*-grammars generating minor-closed sets of graphs that do not contain all planar graphs. We also show how the minimal excluded minors can be effectively constructed from *MS* formulas in certain cases. Our construction uses *HR*-grammars and gives alternative proofs to some results of Fellows and Langston [20]. Section 1 is devoted to a review of definitions from [7, 13, 14].

1. PRELIMINARIES

We denote by $\mathbb{N}$ the set of nonnegative integers, and by $\mathbb{N}_+$ the set of positive ones. We denote by $[n]$ the interval $\{1, 2, 3, \ldots, n\}$ for $n \geqq 0$ (with $[0] = \emptyset$).

For sets A and B, we denote by $A - B$ the set $\{a \in A \mid a \notin B\}$. The cardinality of a set A is denoted by **Card** (A). The powerset of A is denoted by $\mathscr{P}(A)$.

The domain of a partial mapping $f : A \to B$ is denoted by **Dom** (f). The restriction of f to a subset A' of A is denoted by $f \restriction A'$. The partial mapping with an empty domain is denoted by $\emptyset$, as the empty set. If two partial mappings $f: A \rightarrow B$ and $f': A' \rightarrow B$ coincide on $\mathbf{Dom}(f) \cap \mathbf{Dom}(f')$ then we denote by $f \cup f'$ their common extension into a partial mapping: $A \cup A' \rightarrow B$, with domain $\mathbf{Dom}(f) \cup \mathbf{Dom}(f')$.

A binary relation R on a set A is considered as a subset of $A \times A$. Hence, $x R y$ and $(x, y) \in R$ are equivalent notations. The transitive closure of R is denoted by R^+, and its reflexive and transitive closure is denoted by R^*. The set of equivalence relations on A is denoted by $\mathbf{Eq}(A)$.

The set of nonempty sequences of elements of a set A is denoted by A^+. The generic sequence is denoted by $(a_1, \ldots, a_n)$ with commas and parentheses. The empty sequence is denoted by $(\ )$, and A^* is $A^+ \cup \{(\ )\}$. When A is an alphabet, *i.e.*, when its elements are letters, a sequence $(a_1, \ldots, a_n)$ in A^+ can be written unambiguously $a_1 a_2 \ldots a_n$. The empty sequence is denoted by ε, a special symbol reserved for this purpose. The elements of A^* are called words. The length of a sequence μ is denoted by $|\mu|$.

Hypergraphs

As in [7, 13, 14, 15, 17], we deal with labeled, directed hypergraphs equipped with a sequence of distinguished vertices called the sequence of sources. The labels are chosen in a ranked alphabet, *i.e.*, in a set A, each element of which has an associated integer (in $\mathbb{N}$) that we call its type. The type mapping is $\tau: A \rightarrow \mathbb{N}$. The type of the label of a hyperedge must be equal to the length of its sequence of vertices. (This type may be 0, *i.e.*, we allow hyperedges with no vertex.)

Let A (and τ) be as above, let $n \in \mathbb{N}$. A *concrete n-hypergraph* is a quintuple $G = \langle \mathbf{V}_G, \mathbf{E}_G, \mathbf{lab}_G, \mathbf{vert}_G, \mathbf{src}_G \rangle$ where:

– $\mathbf{V}_G$ is the set of vertices of the graph,

– $\mathbf{E}_G$ is its set of edges,

– $\mathbf{lab}_G: \mathbf{E}_G \rightarrow A$ is a total mapping that assigns to each hyperedge of G a label in the alphabet A,

– $\mathbf{vert}_G: \mathbf{E}_G \rightarrow \mathbf{V}_G^*$ is a total mapping that associates with a hyperedge e of G, the sequence of its vertices (this sequence must be of length $\tau(e) := (\mathbf{lab}_G(e))$ and its i-th element is also denoted by $\mathbf{vert}_G(e, i)$, and finally

– $\mathbf{src}_G$ is a sequence of length n in $\mathbf{V}_G^*$ (or equivalently a mapping: $[n] \rightarrow \mathbf{V}_G$), called the *sequence of sources.*

We shall denote by $\mathbf{src}_G(i)$ the i-th element of the sequence $\mathbf{src}_G$. (If $n = 0$, then G has no source.) "Source" is just an easy sounding word for "distinguished vertex". There is no notion of flow involved. The integer n is the type of G. We shall also denote by $\mathbf{SRC}_G$ the *set of sources* of G.

Whenever we need to specify the alphabet A, we say that G is a concrete $n-$hypergraph *over A*. Let G and H be concrete graphs of the same type n. A homomorphism: $G \rightarrow H$ is a pair of mappings $h=(h_{\mathbf{V}}, h_{\mathbf{E}})$ where $h_{\mathbf{V}}: \mathbf{V}_G \rightarrow \mathbf{V}_H$, $h_{\mathbf{E}}: \mathbf{E}_G \rightarrow \mathbf{E}_H$, and such that:

$\mathbf{lab}_H \circ h_{\mathbf{E}} = \mathbf{lab}_G$

$h_{\mathbf{V}}(\mathbf{vert}_G(e, i)) = \mathbf{vert}_H(h_{\mathbf{E}}(e), i)$ for all $i \in [\tau(e)]$, all $e \in \mathbf{E}_G$,

$h_{\mathbf{V}}(\mathbf{src}_G(i)) = \mathbf{src}_H(i)$ for all $i \in [n]$.

If no ambiguity can arise, we denote $h_{\mathbf{V}}$ and $h_{\mathbf{E}}$ by h. An *isomorphism* is a homomorphism such that $h_{\mathbf{V}}$ and $h_{\mathbf{E}}$ are bijective. The isomorphism class of a concrete hypergraph is called an *abstract hypergraph*, or simply a *hypergraph* in the sequel.

A *graph* is a hypergraph, all hyperedges of which are of type 2. A hypergraph G is *finite* if $\mathbf{V}_G$ and $\mathbf{E}_G$ are finite. Otherwise, a hypergraph has at most countably many vertices and edges (in this paper).

We denote by $\mathbf{FCG}(A)_n$, by $\mathbf{FCG}(A)$, by $\mathbf{FG}(A)_n$, and by $\mathbf{FG}(A)$, the sets of finite concrete n-hypergraphs, of finite concrete hypergraphs, of finite n-hypergraphs, and of finite hypergraphs respectively, over A. The notations $\mathbf{CG}(A)_n$, $\mathbf{CG}(A)$, $\mathbf{G}(A)_n$ and $\mathbf{G}(A)$ are used similarly for hypergraphs.

A vertex v *belongs to an edge e* if $v=\mathbf{vert}_G(e, i)$ for some i. A vertex is *isolated* if it belongs to no edge.

For every n in $\mathbb{N}$, we denote by $\mathbf{n}$ the unique n-graph with n pairwise distinct sources. For every a in A of type n, we denote by a the n-hypergraph with a single edge e labeled by a, no internal vertex, and a sequence of n pairwise distinct sources equal to the sequence of vertices of e.

Let G be a concrete hypergraph; let $\approx$ be an equivalence relation on $\mathbf{V}_G$. We denote by $[[v]]$ the equivalence class of v w.r.t. $\approx$. We denote by $G/\approx$ the concrete graph H such that

$$\mathbf{V}_H = \mathbf{V}_G/\approx, \qquad \mathbf{E}_H = \mathbf{E}_G, \qquad \mathbf{lab}_H = \mathbf{lab}_G, \qquad \mathbf{vert}_H(e, i) = [[\mathbf{vert}_G(e, i)]]$$

for all $e \in \mathbf{E}_H (= \mathbf{E}_G)$, all $i \in [\tau(e)]$, and $\mathbf{src}_H(i) = [[\mathbf{src}_G(i)]]$ for all $i \in [\tau(G)]$. We call $G/\approx$ the *quotient graph* of G by $\approx$.

Operations on hypergraphs

We recall the definitions of the basic operations on hypergraphs introduced in Bauderon and Courcelle [7] (*see* also [6, 13, 14, 17]). The first one is the *disjoint union* $\underset{n,m}{\oplus}$ such that, if $G \in \mathbf{G}(A)_n$, $H \in \mathbf{G}(A)_m$, then $K = G \underset{n,m}{\oplus} H$ is such that:

$$\mathbf{V}_K = \mathbf{V}_G \cup \mathbf{V}_H$$

$$\mathbf{E}_K = \mathbf{E}_G \cup \mathbf{E}_H$$

(we assume that $\mathbf{V}_G \cap \mathbf{V}_H = \varnothing$ and $\mathbf{E}_G \cap \mathbf{E}_H = \varnothing$)

$$\mathbf{vert}_K = \mathbf{vert}_G \cup \mathbf{vert}_H$$

$$\mathbf{lab}_K = \mathbf{lab}_G \cup \mathbf{lab}_H$$

$\mathbf{src}_K = \mathbf{src}_G . \mathbf{src}_H$, namely is equal to the concatenation of the sequences of sources of the two hypergraphs.

The second operation is the source *redefinition*. For each mapping α from $[p]$ to $[n]$, we have an operation σ_α such that for G in $\mathbf{G}(A)_n$, $\sigma_\alpha(G) = \langle \mathbf{V}_G, \mathbf{E}_G, \mathbf{lab}_G, \mathbf{vert}_G, \mathbf{src}_G \circ \alpha \rangle$. If $p = 0$, then α is necessarily the empty map (always denoted by $\varnothing$), and $\sigma_\alpha(G)$ is the 0-graph obtained from G by "forgetting" its sources. We also denote it by G^0.

When p is small it is convenient to write $\sigma_{i_1, i_2, \ldots, i_p}(G)$ instead of $\sigma_\alpha(G)$, with $i_j = \alpha(j)$ for $j = 1, \ldots, p$.

The third operation is the *source fusion*. For every equivalence relation δ on $[n]$, we let θ_δ be the mapping: $\mathbf{G}(A)_n \to \mathbf{G}(A)_n$ that transforms a hypergraph G into its quotient by the equivalence relation generated by the set of pairs of vertices $\{(\mathbf{src}_G(i), \mathbf{src}_G(j)) / (i, j) \in \delta\}$.

Intuitively, $\theta_\delta(G)$ is obtained from G by fusing the i-th and j-th sources, whenever i and j are equivalent w.r.t. δ. If δ is the equivalence relation on $[n]$ generated by a single pair (i, j), then we denote θ_δ by $\theta_{i,j}$. It is clear that if δ is the equivalence relation generated by a set of pairs $\{(i_1, j_1), \ldots, (i_k, j_k)\}$, then:

$$\theta_\delta = \theta_{i_1, j_1} \circ \ldots \circ \theta_{i_k, j_k}.$$

These operations and the constants $\mathbf{n}$ (for all $n \in \mathbb{N}$) and a (for all a in A) introduced above form a many-sorted signature denoted by $\mathbf{H}_A$. Every finite, well-formed term written with them denotes a finite hypergraph. These terms will be called *hypergraph expressions*. The *width* of an expression is the maximal sort of any symbol in it. The *width* $\mathbf{wd}(G)$ of a hypergraph G is the minimal width of an expression denoting it.

Finite terms with variables denote mappings on hypergraphs called *derived operations*. More precisely, if t is a term of sort p written with variables $x_1, \ldots, x_n$ of respective sorts $m_1, \ldots, m_n$, then t denotes in a standard way a total mapping: $\mathbf{G}(A)_{m_1} \times \ldots \times \mathbf{G}(A)_{m_n} \to \mathbf{G}(A)_p$.

Infinite expressions without variables have been used in Courcelle [14] and Bauderon [6] in order to denote infinite hypergraphs. Such expressions can be considered as infinite trees, the nodes of which are labeled by symbols from $\{ \underset{n,m}{\oplus}, \theta_\delta, \sigma_\alpha, \mathbf{n}, a \}$. The formal definition of the hypergraph denoted by an infinite expression will be recalled from Courcelle [14] just prior to its use, in the proof of Theorem (2.2). The notion of *width* is as for finite hypergraphs. Some infinite hypergraphs have an infinite width, whereas the width of a finite hypergraph is always finite.

Logical structures representing hypergraphs

A hypergraph can be considered as a two-sorted logical structure with two domains: the set of vertices and the set of hyperedges. Constants $\mathbf{s}_1, \ldots, \mathbf{s}_n$ denote the n sources of an n-graph, and for each a in A, a relation $\mathbf{edg}_a(e, v_1, \ldots, v_k)$ expresses that e is a hyperedge with label a, and sequence of vertices $(v_1, \ldots, v_k)$.

Hence, one can express graph properties by logical formulas. We shall consider monadic second-order (*MS*) formulas, where quantified variables can denote edges, vertices, sets of edges, and sets of vertices. The symbol $\mathscr{L}$ will refer to this logical language. In particular, a set of graphs is *definable* if it is the set of all graphs satisfying such a formula. The *monadic (second-order) theory* of a set of graphs L is the set of all closed *MS* formulas that are valid in all graphs of L. It is denoted by $\mathbf{Th}(L)$.

2. TREE-DECOMPOSITIONS AND HYPERGRAPH EXPRESSIONS

We establish that from every (finite or infinite) expression defining a hypergraph, one can construct a tree-decomposition of this hypergraph and conversely, from a tree-decomposition, one can construct an expression. It follows from these two constructions that the *tree-width* of a hypergraph as defined by Robertson and Seymour [26] and its *width* as recalled above from Bauderon and Courcelle [7, 14] are linearly related.

(2.1) DEFINITION: *Tree-width.*

Let G be an n-hypergraph. A *tree-decomposition* of G is a pair (T, f) consisting of a tree T with root $\mathbf{r}(T)$, and a mapping $f: \mathbf{V}_T \rightarrow \mathcal{P}(\mathbf{V}_G)$ such that:

(1) $\mathbf{V}_G = \bigcup \{f(i)/i \in \mathbf{V}_T\}$,

(2) every hyperedge of G has all its vertices in $f(i)$ for some i,

(3) if $i, j, k \in \mathbf{V}_T$, and if j is on the unique, cycle-free, undirected path in T from i to k, then $f(i) \cap f(k) \subseteq f(j)$,

(4) $\mathbf{SRC}_G \subseteq f(\mathbf{r}(T))$.

The *width* of such a decomposition is defined as:

$$\mathbf{Max}\{\mathbf{Card}(f(i))/i \in \mathbf{V}_T\} - 1.$$

The *tree-width* of G is the minimum width of a tree-decomposition of G. It is denoted by $\mathbf{twd}(G)$, and belongs to $\mathbb{N} \cup \{\infty\}$.

For a 0-hypergraph, condition (4) is always satisfied in a trivial way. Similarily, condition (2) is always satisfied for the hyperedges of type 0 or 1 (provided condition (1) holds). Such hyperedges can be added to or deleted from a hypergraph without changing its tree-width.

If i and j are two adjacent nodes of T, one may have $f(i) \cap f(j) = \varnothing$. Hence, G is not necessarily connected, although it is described by a (connected) tree. It is not hard to see that for every 0-hypergraph G, $\mathbf{twd}(G) = \mathbf{Sup}\{\mathbf{twd}(G')/G'$ is a connected component of $G\}$. Our aim is to establish the following result:

(2.2) THEOREM: *Let A be ranked alphabet, such that $\tau(A) := \mathbf{Max}\{\tau(a)/a \in A\} < \infty$. For every hypergraph G over A we have:*

(1) $\mathbf{twd}(G) \leqq \mathbf{wd}(G) - 1$,

(2) $\mathbf{wd}(G) \leqq \mathbf{Max}\{2\,\mathbf{twd}(G) + 2, \mathbf{twd}(G) + 1 + \tau(A), \tau(G)\}$.

For proving this theorem, a few preliminary definitions and lemmas are necessary.

(2.3) DEFINITION: *Tree-gluings.*

Let T be a rooted tree. For every x in $\mathbf{V}_T$, let G_x be a hypergraph. We assume that G_x is disjoint from G_y, for $x \neq y$. For every x in $\mathbf{V}_T$, let E_x be a (possibly empty) subset of $\mathbf{V}_{G_x} \times \mathbf{V}_{G_x}$ and for every pair (x, y) of nodes such that y is a son of x, we let R_{xy} be a subset of $\mathbf{V}_{G_x} \times \mathbf{V}_{G_y}$.

Let $K := \bigcup\{G_x/x \in \mathbf{V}_T\}$, equipped with $\mathbf{src}_{\mathbf{r}(T)}$ as a sequence of sources.

Let $\approx$ be the equivalence relation on $\mathbf{V}_K$ generated by

$$R := \bigcup \{ E_x / x \in \mathbf{V}_T \} \cup (\bigcup \{ R_{xy} / x,\ y \in \mathbf{V}_T \})$$

(*i. e.*, $\approx$ is the reflexive, symmetric, and transitive closure of R).

Let $G := K/\approx$, *i.e.*, be the quotient of K by the equivalence relation $\approx$ on $\mathbf{V}_K$. (Any two equivalent vertices of K become identical in G.) We denote this hypergraph by $\Sigma_{(T, E, R)} \{ G_x / x \in \mathbf{V}_T \}$ and call it a *tree-gluing* of the family $(G_x)_{x \in \mathbf{V}_T}$.

A tree-decomposition of G can be obtained canonically from this construction. Let $h : K \to K/\approx = G$ be the canonical surjective homorphism, and:

$$f(x) := h_{\mathbf{V}}(\mathbf{V}_{G_x}) \subseteqq \mathbf{V}_G \quad \text{for all } x \in \mathbf{V}_T.$$

With this notation:

(2.4) LEMMA: *(T, f) is a tree-decomposition of G. If* $\mathbf{Card}(\mathbf{V}_{G_x}) \leqq k$ *for all $x \in \mathbf{V}_T$, then the width of this tree-decomposition is at most $k-1$.*

Proof : We only have to verify Condition (3) of Definition (2.1), the other conditions being clearly satisfied. Let $x,\ y \in \mathbf{V}_T$, let $(x, z_1, z_2, \dots, z_k, y)$ be the unique path linking x to y in T.

Let v be a vertex of $f(x) \cap f(y)$. We shall prove that v belongs to all the sets $f(z_i)$, $i = 1, \dots, k$. There exist w_x in $\mathbf{V}_{G_x}$ and w_y in $\mathbf{V}_{G_y}$ such that $h(w_x) = h(w_y) = v$. Since $h(w_x) = h(w_y)$, $w_x \approx w_y$. From the definition of $\approx$, it is clear that there exist vertices $w_1, w'_1, \dots, w'_k, w'_y$ of G_y such that $w_x \approx w'_x$, $(w'_x, w_1) \in S_{x, z_1}$, $w_1 \approx w'_1$, $(w'_1, w_2) \in S_{z_1, z_2}$, $w_2 \approx w'_2, \dots w_k \approx w'_k$, $(w'_k, w'_y) \in S_{z_k, y}$, $w'_y \approx w_y$, where $S_{u, v} := R_{u, v} \cup (R_{u, v})^{-1}$ for all u, v.

Hence

$$h(w_x) = h(w'_x) = h(w_1) = h(w'_1) = \dots = h(w_i) = \dots = h(w'_k) = h(w'_y) = h(w_y) = v.$$

This establishes that $v = h(w_i)$ belongs to $f(z_i)$ for all $i = 1, \dots, k$. □

(2.5) *Remark:* Let G be a hypergraph, and (T, f) be one of its tree-decompositions. Then $G = \Sigma_{(T, E, R)} \{ G_x / x \in \mathbf{V}_T \}$, and (T, f) is the associated tree-decomposition as defined by the construction of Definition (2.3) for well-chosen E, R, and G_x. To see this, let:

- G_x be a subhypergraph of G, the set of vertices of which is $f(x)$, and such that every edge of G belongs to G_x for one and only one x in $\mathbf{V}_T$;
- $R_{xy} = \{ (v, w) \in \mathbf{V}_{G_x} \times \mathbf{V}_{G_y} / v = w \}$ for all pair (x, y) of nodes of T such that y is a son of x;
- $E_x = \varnothing$;

• the sequence of sources of $G_{\mathbf{r}(T)}$ is that of G.

It is clear that $G=\Sigma_{(T,\,E,\,R)}\{G_x/x\in \mathbf{V}_T\}$ and that (T, f) is the tree-decomposition of G associated with (T, E, R) by Definition (2.3).

Proof of Theorem (2.2), *assertion* (1): We denote by $\mathbf{H}_A^{[k]}$ the restriction of $\mathbf{H}_A$ to symbols of sort at most k. We denote by $\mathbf{M}^\infty(\mathbf{H}_A^{[k]})$ the set of infinite (expression) trees formed over $\mathbf{H}_A^{[k]}$. It follows that $\mathbf{wd}(G)\leqq k$ iff $G=\mathbf{val}(T)$ for some T in $\mathbf{M}^\infty(\mathbf{H}_A^{[k]})$. Let G satisfy this condition. We shall build from T a tree-decomposition of G. The construction of $G=\mathbf{val}(T)$ of Courcelle [14, Definition 5.3] can be formulated with the notation of Definition (2.3) as:

$$G=\Sigma_{(T,\,E,\,R)}\{G_x/x\in \mathbf{V}_T\}$$

where E, R and the hypergraphs G_x are defined as follows:

(1) for every node x of T having a label in $\mathbf{H}_A$-A, we let $G_x := \mathbf{m}$ where $m=\tau(x)$;

(2) for every node x with label a in A we let $G_x := a$;

(3) for every node x, we let E_x be δ if the label of x is θ_δ, and be $\varnothing$ otherwise;

(4) for every node x labeled by $\underset{n,\,m}{\oplus}$ then, if y and z are its first and second successors, we let

$$R_{x,\,y} := \{(\mathbf{src}_{G_x}(i),\ \mathbf{src}_{G_y}(i))/i=1,\ \ldots,\ n\},$$

$$R_{x,\,z} := \{(\mathbf{src}_{G_x}(i+n),\ \mathbf{src}_{G_z}(i))/i=1,\ \ldots,\ m\};$$

(5) for every node x labeled by θ_δ with $\delta\in\mathbf{Eq}([n])$, then, if y is its unique successor, we let:

$$R_{x,\,y} := \{(\mathbf{src}_{G_x}(i),\ \mathbf{src}_{G_y}(i))/i\in[n]\};$$

(6) for every node x labeled by σ_α, where $\alpha:[n]\to[m]$, then, if y is its unique successor, we let

$$R_{xy} := \{(\mathbf{src}_{G_x}(i),\ \mathbf{src}_{G_y}(\alpha(i)))/i\in[n]\}.$$

Each graph G_x has at most k vertices since every symbol of $\mathbf{H}_A$ occurring in T is, by hypothesis, of type at most k. It follows from Lemma (2.4) that the associated tree-decomposition of $\mathbf{val}(T)$ is of width at most $k-1$.

Hence, for every graph G, $\mathbf{twd}(G)\leqq\mathbf{wd}(G)-1$. □

Proof of Theorem (2.2), *assertion* (2): From a tree-decomposition (T, f) of G, of minimal width $k-1$, so that $k = \mathbf{Max}\{\mathbf{Card}(f(i))/i \in \mathbf{V}_T\}$, we shall construct an expression defining G, of width bounded by a function of k. This construction takes as input a finite or infinite tree T. It is effective whenever T is finite or infinite as long T is given in some effective way.

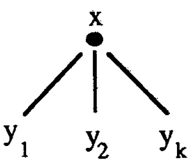
It is useful to *order* T in some fixed way. We do this in such a way that every node has a finite or infinite (possibly empty) sequence of successors.

First step: We transform (T, f) into a tree-decomposition (T', f') of G, having the same width, and such that T' is a binary tree, *i.e.* such that every node of T' has either zero or two successors. For each node x of $\mathbf{V}_T$, we do the following.

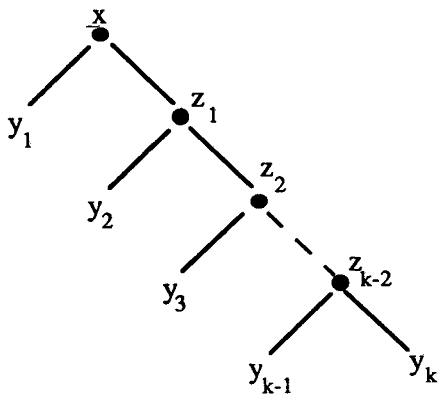
If x has 0 or 2 successors, we do nothing.

If x has one successor y, we add a second one y', and we let $f'(y') := \varnothing$.

If x has a finite sequence of successors $(y_1, y_2, \ldots, y_k)$, then we replace



by



and we let $f'(z_1) := f'(z_2) := f'(z_{k-2}) := f(x)$.

If x has an infinite sequence of successors $(y_1, y_2, y_3, \ldots)$, we add similarily an infinite sequence of nodes $(z_1, z_2, \ldots)$, and we let $f'(z_i) := f(x)$ for all i.

Second step: We now let (T, f) denote the newly obtained (T', f'), in order to simplify the notations. By remark (2.5), we can express G as $\Sigma_{(T, \varnothing, R)}\{G_x/x \in \mathbf{V}_T\}$ where R and G_x are constructed appropriately, and each G_x has at most k vertices.

For every x in $\mathbf{V}_T$, let us choose, in addition, a sequence of sources s_x enumerating without repetition the set of vertices of G_x. Then let $\tau(x) := |s_x|$. For each pair (x, y) of nodes of T where y is a son of x, we define

$\bar{R}_{xy} \subseteq [|s_x|] \times [|s_y|]$ by:

$$(i, j) \in \bar{R}_{xy} \quad \text{iff} \quad (s_x(i), s_y(j)) \in R_{xy}.$$

For every $x \in \mathbf{V}_T$, let g_x be an expression defining $\langle G_x, s_x \rangle$. Since G_x has at most k vertices, and since the length of its sequence of sources is also at most k, we can construct g_x, that defines G_x, and is of width at most $k + \tau(A)$. (One constructs G_x by starting from the graph $\mathbf{k}'$, where $k' = |s_x|$, and by gluing successively all necessary edges; each new gluing is of the form $\sigma_\alpha(\theta_\delta(g \oplus a))$, where g is an expression of type k'; hence, the width of $\sigma_\alpha(\theta_\delta(g \oplus a))$ is $\mathbf{Max}\{\mathbf{wd}(g), k' + \tau(a)\}$.)

Next we have to put together all the expressions g_x, $x \in \mathbf{V}_T$, and from a single one, defining the graph $\Sigma_{(T, \varnothing, R)}\{G_x / x \in \mathbf{V}_T\}$.

For every node x of T having two successors y and z, we let p_x be the derived operation:

$$p_x(u, v) : = \sigma_\mu(\theta_\nu(\sigma_\alpha(\theta_\delta(g_x \oplus \sigma_\beta(u)) \oplus \sigma_\gamma(v))))$$

where we let:

$q : = \tau(x)$;

$\tau(u) : = n := \tau(y)$;

$\tau(v) : = m := \tau(z)$;

$\beta : [n'] \to [n]$ be the mapping such that $\beta(i)$ is the rank in s_y of the i-th elements of s_y belonging to $\mathbf{Im}(R_{xy})$, hence, $n' \leqq n$);

$\delta \in \mathbf{Eq}([q + n'])$ be the equivalence relation generated by the set $\{(i, q + j)/(i, j) \in \bar{R}_{xy}\}$;

α be the inclusion map: $[q] \to [q + n']$;

$\gamma : [m'] \to [m]$ be such that $\gamma(i)$ is the rank in s_z of the i-th element of s_z belonging to $\mathbf{Im}(R_{xz})$, hence $m' \leqq m$);

$\nu \in \mathbf{Eq}([q + m'])$ be generated by $\{(i, q + j)/(i, j) \in \bar{R}_{xz}\}$, and, finally,

μ be the inclusion map: $[q] \to [q + m']$.

This derived operation is of width $\mathbf{Max}\ \{n, m, q, q + n', q + m', \mathbf{wd}(g_x)\}$; hence, its width is at most $\mathbf{Max}\{2k, k + \tau(\mathrm{A})\}$. It has been constructed in such a way that, for all graphs H_y and H_z of respective types $n = \tau(y)$ and $m = \tau(z)$, we have:

$$p_x(H_y, H_z) = \Sigma_{(T', \varnothing, S)}\{G_x, H_y, H_z\}, \text{ where}$$
$$S_{xy} = \{(\mathbf{src}_{G_x}(i), \mathbf{src}_{H_y}(j))/(i, j) \in \bar{R}_{xy}\},$$
$$S_{xz} = \{(\mathbf{src}_{G_x}(i), \mathbf{src}_{H_z}(j))/(i, j) \in \bar{R}_{xz}\}$$

and T' is the tree:

$$\begin{tikzcd} & x \arrow[dl, no head] \arrow[dr, no head] & \\ y & & z \end{tikzcd}$$

This is illustrated in Figure 1. (The relation S_{xy} is indicated by straight lines, the relation S_{xz} by dotted ones, $n'=4$ and $m'=3$. The big dots ● indicate the sources).

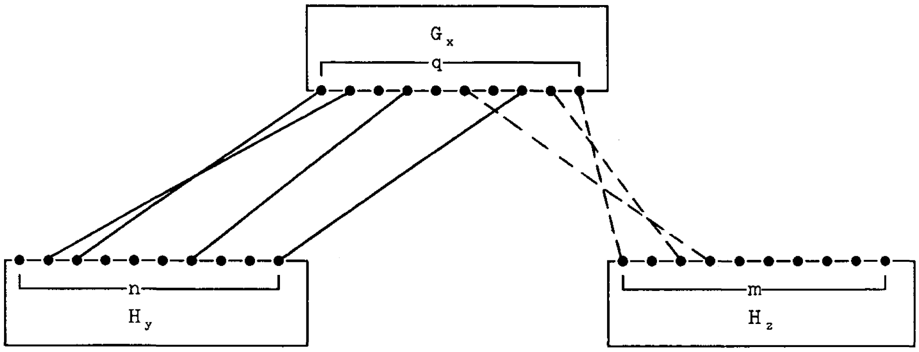


Figure 1

Let P be the set of derived operations consisting of:

all the operations p_x such that x is in $\mathbf{V}_T$ and has two successors, and

all the expressions (that are derived operations without arguments) of the form g_x, where x is a leaf of T.

Let t be the tree in $\mathbf{M}^{\infty}(P)$ such that $\mathbf{dom}(t)=\mathbf{V}_T$, $t((x))=g_x$ if x is a leaf, $t((x))=p_x$ if x is not a leaf.

The value of t considered as an expression over the derived signature P is the hypergraph

$$G'=\Sigma_{(T,\,\varnothing,\,R)}\{G_x/x\in\mathbf{V}_T\}, \qquad \text{and} \qquad \mathbf{wd}(t)\leqq\mathbf{Max}\{2k,\, k+\tau(A)\}.$$

This G' is "almost equal" to G: to be precise $G'^0=G^0$, and $\mathbf{SRC}_G\subseteqq\mathbf{SRC}_{G'}=s_{r(T)}$. Hence, $G=\mathbf{val}(\sigma_\alpha(t))$ for some appropriately chosen mapping $\alpha:[\tau(G)]\to[|s_{r(T)}|]$.

It follows that:

$$\mathbf{wd}(G)\leqq\mathbf{Max}\{2k,\, k+\tau(A),\, \tau(G)\}.$$

This has been obtained under the assumption that $\mathbf{twd}(G)=k-1$. Hence:

$$\mathbf{wd}(G) \leqq \mathbf{Max}\{\tau(G),\ 2\,\mathbf{twd}(G)+2,\ \mathbf{twd}(G)+1+\tau(A)\}. \quad \square$$

(2.6) *Remarks:* (1) The correspondence between width and tree-width is not exact: the two graphs $\mathbf{1}$ and $(a)-\bullet-(b)$ have the same tree-width, namely 0; the first one has width 1, and the second one has width 2.

These examples show in addition that the upper bounds of (1) and (2) of Theorem (2.2) can be reached: for the first graph we have $0=\mathbf{twd}(\mathbf{1})=\mathbf{wd}(\mathbf{1})$-1, and for the second one, let us call it G, we have $2=\mathbf{wd}(G)=\mathbf{Max}\{2\,\mathbf{twd}(G)+2,\ \mathbf{twd}(G)+1+\tau(A),\ \tau(G)\}$ since $\mathbf{twd}(G)=0$, $\tau(A)=1$ and $\tau(G)=0$.

(2) If G is finite, then the expression constructed from a tree-decomposition (T, f) of G is of size at most $18\,\mathbf{Card}(\mathbf{V}_T)+4\,\mathbf{Card}(\mathbf{E}_G)$. This is a consequence of the following observations:

– the initial transformation of T into a binary tree multiples $\mathbf{Card}(\mathbf{V}_T)$ at most by 2,

– $\mathbf{size}(g_x) \leqq 1+4\,\mathbf{Card}(\mathbf{E}_{G_x})$

– $\mathbf{size}(p_x) \leqq 8+\mathbf{size}(g_x)$. $\square$

Here are a few easy consequences of Theorem (2.2). The width (or tree-width) of a set of hypergraphs is the least upper bound of the widths (or tree-widths) of its members.

(2.7) COROLLARY: *Let A be a finite ranked alphabet.*

(1) *A subset L of* $\mathbf{FG}(A)_n$ *has a finite width iff it has a finite tree-width. If L is generated by an HR-grammar, then its tree-width is finite.*

(2) *A hypergraph G in* $\mathbf{G}(A)_n$ *has a finite width iff it has a finite tree-width. If G is equational then its tree-width is finite.*

Hyperedge replacement grammars have been considered in Bauderon and Courcelle [7]. It is proved in [7, Proposition 4.17] that they generate sets of hypergraphs of finite width, hence, of finite tree-width.

Equational graphs have been introduced in Courcelle [14]. They are of finite width by [14, Definition 5.8], hence, of finite tree-width.

A number of results have been established for graphs, results that we wish to extend to hypergraphs. For this purpose, we introduce a transformation K from hypergraphs to 0-graphs that preserves tree-width and will yield the desired applications.

Letting A be a ranked alphabet, we let B consist of the symbols of A of rank 2. We create a new symbol if necessary, so that we can assume that $B \neq \varnothing$. We define a mapping $K : \mathbf{G}(A)_n \to \mathbf{G}(B)_0$ as follows. For every $G \in \mathbf{G}(A)_n$, we construct $\bar{K}(G)$ by doing the following:

– we delete the hyperedges of type 0 or 1 (while keeping the vertices of unary edges);

– for every hyperedge of type $m \geqq 3$ we substitute a complete graph in $\mathbf{G}(B)_m$.

Hence, $\bar{K}(G)$ is an n-graph. We make it into a 0-graph $K(G)$ in $\mathbf{G}(B)_0$ by adding a new edge between any two sources, and turning the sources into internal, *i.e.*, nonsource vertices.

By the following lemma, this construction makes it possible to use for hypergraphs the known algorithms that decide whether a graph is of tree-width at most k and that construct tree-decompositions of width at most k, when they exist. We shall say more about that in Section 3.

(2.8) LEMMA: $\mathbf{twd}(G) = \mathbf{twd}(K(G)) = \mathbf{twd}(\bar{K}(G))$.

Proof: Since $K(G)$ and $\bar{K}(G)$ are obtained from G by deletions and additions of edges, we can assume that the vertices are the same, *i.e.*, that $\mathbf{V}_{K(G)} = \mathbf{V}_{\bar{K}(G)} = \mathbf{V}_G$. It is clear that every tree-decomposition of G is also a tree-decomposition of $K(G)$ and of $\bar{K}(G)$.

Let us now consider a tree-decomposition (T, f) of $K(G)$ [or of $\bar{K}(G)$]. For every complete subgraph K of $K(G)$, [or $\bar{K}(G)$] there is an element $i \in \mathbf{V}_T$ such that $f(i)$ contains all vertices of K. (This fact is easy; a proof can be found, for instance, in Bodlaender [8]). It follows that (T, f) is a tree-decomposition of G. □

D. Seese has proved in [31, 32] that if a set or finite or infinite graphs has a decidable monadic theory, then its tree-width is finite. We shall extend his result to hypergraphs. For doing this, we now specify more precisely the transformation $K : \mathbf{G}(A)_n \to \mathbf{G}(B)_0$. We replace a hyperedge of G labeled by a of type k, with a sequence of vertices $(v_1, \ldots, v_k)$, by binary edges with label $(a, (i, j))$, linking v_i to v_j, for all i and j such that $i<j$. Edges labeled by $(\S, (i, j))$ are added, linking the i-th source of G to the j-th one, for $i<j$. In this way, we obtain a graph $K(G)$, the edges of which are labeled in the alphabet:

$$B := \{(a, (i, j))/a \in A,\ \tau(a)>1,\ 1 \leqq i \leqq j \leqq \tau(a)\} \cup \{(\S, (i, j))/1 \leqq i < j \leqq n\}.$$

We let $\mathbf{U}$ denote $\mathbf{G}(B)_0$.

(2.9) LEMMA: *If L is a definable subset of* $\mathbf{U}$, *then*

$$K^{-1}(L) := \{G \in \mathbf{G}(A)_n / K(G) \in L\}$$

is definable (for fixed n and A).

Proof sketch: Let $L = \{H \in \mathbf{U}/H \models \varphi\}$ for some monadic second-order formula φ. We wish to construct ψ such that, for every G in $\mathbf{G}(A)_n$, $G \models \psi$ iff $K(\mathrm{G}) \models \varphi$.

For simplicity, we shall assume that all symbols of A are of type 3, and that $n=0$. The general case is similar, but more complicated.

The graph $K=K(G)$ can be constructed in such a way that: $\mathbf{V}_K = \mathbf{V}_G$, $\mathbf{E}_K \subseteq \mathbf{E}_G \times C$, where C is the set of pairs

$$(i, j), \quad 1 \leqq i < j \leqq 3,$$
$$\mathbf{lab}_K((e, (i, j))) = (\mathbf{lab}_G(e), (i, j)),$$

and the two vertices of $(e, (i, j))$ are the i-th and the j-th ones of e in G.

We sketch the translation of φ in $\hat{\varphi}$ such that:

$$K \models \varphi \quad \text{iff} \quad G \models \hat{\varphi}.$$

The basic remark is that a subset X of $\mathbf{E}_K = \mathbf{E}_G \times C$ can be represented by a triple $(X_{1,2}, X_{1,3}, X_{2,3})$ of subsets of $\mathbf{E}_G$.

We construct $\hat{\varphi}$ by induction on the structure of φ. We can assume that φ is written with simplified syntax using set quantifications only (*see* Section 1).

For each set variable Y of sort $\mathbf{e}$, we shall use set variable $Y_{i,j}$, $1 \leqq i \leqq j \leqq 3$ of the same sort. In the definitions below, X, X′ denotes variables of sort $\mathbf{v}$, and Y, Y' denote variables of sort $\mathbf{e}$.

If φ is $X \subseteq X'$, then $\hat{\varphi}$ is $X \subseteq X'$.

If φ is $Y \subseteq Y'$, then $\hat{\varphi}$ is $(Y_{1,2} \subseteq Y'_{1,2}) \wedge (Y_{1,3} \subseteq Y'_{1,3}) \wedge (Y_{2,3} \subseteq Y'_{2,3})$.

If φ is $\mathbf{edg}_{(a,i,j)}(Y, X, X')$ then $\hat{\varphi}$ is $\mathbf{edg}_a(Y_{i,j}, X, X')$.

If φ is $\neg\varphi_1$ or $\varphi_1 \vee \varphi_2$ or $\exists \mathrm{X}.\varphi_1$, then $\hat{\varphi}$ is $\neg\hat{\varphi}_1$ or $\hat{\varphi}_1 \vee \hat{\varphi}_2$ or $\exists X.\hat{\varphi}_1$, respectively.

If φ is $\exists Y.\varphi_1$, then $\hat{\varphi}$ is $\exists Y_{1,2}, Y_{1,3}, Y_{2,3}.\hat{\varphi}_1$.

It is quite easy to verify that $\hat{\varphi}$ satisfies the desired property. □

We denote by $\mathbf{G}(A)_n^{[k]}$ the set of n-graphs over A of width at most k, *i.e.* of graphs denoted by expressions in $\mathbf{M}^{\infty}(\mathbf{H}_A^{[k]})$.

(2.10) PROPOSITION: *Let $L \subseteq \mathbf{G}(A)_n$. If the monadic theory of L is decidable, then L is of finite width and tree-width.*

Proof: Let $L \subseteq \mathbf{G}(A)_n$ have a decidable monadic theory. Then, by Lemma (2.9) the set $K(L) \subseteq \mathbf{D}_0$ also has a decidable monadic theory. (For every formula φ in $\mathscr{L}$, one can construct ψ such that for every G, $G \models \psi$ iff $K(G) \models \varphi$; hence, $\varphi \in \mathbf{Th}(K(L))$ iff $\psi \in \mathbf{Th}(L)$.)

By Seese's result, $K(L)$ is of finite tree-width. The same holds for L by Lemma (2.8), and L is of finite width by Corollary (2.7). □

3. COMPLEXITY ISSUES

We have proved in [13, Proposition (4.14)] that every monadic second-order property of a hypergraph can be decided in linear time if the hypergraph is given either by an expression over a fixed finite subsignature of $\mathbf{H}_A$ or by a derivation sequence relative to a fixed hyperedge replacement grammar. This gives efficient decision algorithms for quite a lot properties (including *NP*-complete ones like 3-vertex colorability or the existence of a Hamiltonian circuit) restricted to particular classes of graphs and hypergraphs.

By the results of Section 2, the class of hypergraphs of tree-width at most k (for fixed k) is one of these classes, because its elements can be defined by expressions of width at most some fixed m, hence, that are written over the finite signature $\mathbf{H}_A^{[m]}$.

The algorithms derived from Proposition (4.14) of [13] take as input an expression or a derivation sequence defining the hypergraph of interest, and are linear in these data. If the hypergraph is given without these auxiliary data one must construct an expression for it, (or equivalently, a tree-decomposition by the results of Section 2), or a derivation sequence. This is not necessarily an easy task, as shown by the following results:

Result 1: For each k, one can decide whether a hypergraph G is of tree-width at most k by an $O(|\mathbf{V}_G|^{k+2} + |\mathbf{E}_G|)$-algorithm that produces a tree-decomposition of width $\leq k$ if there exists one (Arnborg *et al.* [1]). In the special cases of $k \leq 3$, there exist alternative $O(|\mathbf{V}_G| + |\mathbf{E}_G|)$-algorithms (Arnborg *et al.* [4]).

Result 2: For each k, there exists an $O(|\mathbf{V}_G|^2 + |\mathbf{E}_G|)$-algorithm that decides whether G is of tree-width at most k by the results of Robertson and Seymour [28, 29] and Bodlaender [11], but it is not explicitly known because it depends on knowledge of the (finitely many) minimal forbidden minors characterizing graphs of tree-width at most k. (*See* Section 4.) This algorithm also constructs a tree-composition of width $\leq k$ when such exists.

These two results yield polynomial algorithms for deciding monadic second-order properties of the sets of hypergraphs of tree-width at most k.

Nothing is known yet concerning the complexity of expressing a hypergraph in terms of a finite set of operations from $\mathbf{H}_{\mathrm{A}}$. This problem is actually a special case of the *parsing problem* for hyperedge replacement grammars, *i.e.*, the problem of either finding a derivation sequence producing a given hypergraph or reporting that no such derivation sequence exists. The main known results concerning this problem are the following ones:

Result 3: Some context-free sets of hypergraphs have an *NP*-complete membership problem. This is the case of the set of graphs of cyclic bandwidth at most k for each $k \geqq 2$, by the result of Leung *et al.* [24] showing *NP*-completeness. (Hyperedge replacement grammars generating these sets of graphs are easy to construct.) Hence, polynomial algorithms cannot be obtained from *arbitrary* hyperedge replacement grammars.

Result 4: Some hyperedge replacement grammars considered by Lautemann [23] and Vogler [34] have polynomial parsing algorithms. However, these results do not apply to the construction of tree-decompositions of width $\leqq k$.

In the above discussion, we have considered the construction of polynomial algorithms for deciding monadic second-order properties φ on classes of hypergraphs L, that, for every hypergraph G, would deliver the following exclusive answers:

(1) $G \notin L$

(2) $G \in L$ and $G \models \varphi$

(3) $G \in L$ and $G \models \neg \varphi$.

The following proposition is based on results by Robertson and Seymour [28, 29] and shows how to construct quadratic algorithms delivering the following non exclusive answers

(1') $G \notin L$

(2') $G \models \varphi$

(3') $G \models \neg \varphi$

where answers (2') or (3') are obtained whenever $G \in L$, but also in some cases, when $G \notin L$.

(3.1) Proposition: *Given φ, k, and n such that $\varphi \in \mathscr{L}$ and $k \geqq n$, one can construct an algorithm that says correctly for every finite n-graph G over A,* *and in time* $\mathbf{O}(\mathbf{size}(G)^2)$ (*or* $\mathbf{O}(\mathbf{Card}(\mathbf{V}_G))$ *in the special case where* $\mathbf{E}_G = \varnothing$):

$$\text{either that } \mathbf{twd}(G) > k,$$
$$\text{or that } \mathbf{twd}(G) \leqq 5k+9 \text{ and } G \models \varphi,$$
$$\text{or that } \mathbf{twd}(G) \leqq 5k+9 \text{ and } G \models \neg \varphi,$$

Proof: We shall use the notion of *branch-width* of a graph introduced by Robertson and Seymour in [28]. This notion is defined in term of *branch-decompositions*, as tree-width is defined in terms of tree-decompositions. We need not give the definition. We shall only use the result of [28] saying that the branch-width $\beta(G)$ of a graph G is related to its tree-width as follows:

$$\beta(G) \leqq \mathbf{twd}(G) + 1 \leqq 3\,\beta(G)/2$$

whenever $\beta(G) \geqq 2$. The proof provides effective transformations between tree-decompositions and branch-decompositions that can be done in linear time. The sizes of the decompositions are related linearly.

The same authors have given in [29] an algorithm that, for every integer w, for every graph G, gives in time $\mathbf{O}(\mathbf{Card}(\mathbf{E}_G)\,\mathbf{Card}(\mathbf{V}_G))$ the following outputs (we assume that $\mathbf{E}_G \neq \varnothing$):

– either the answer that $\beta(G) \geqq w$,

– or a branch-decomposition of width at most $3w$.

By applying this algorithm to a graph G (with $\mathbf{E}_G \neq \varnothing$), and to $w = k+2$, one obtains:

– either the answer that $\mathbf{twd}(G) > k$ (*since* $\mathbf{twd}(G) \geqq \beta(G) - 1$),

– or a tree-decomposition of G of width at most $(9/2)\ (k+2) < 5k+9$, (from the eventual branch decomposition of width at most $3w$).

If G is a graph with $\mathbf{E}_G = \varnothing$, then $\mathbf{twd}(G) = 0$, and one can construct in time $\mathbf{O}(\mathbf{card}(\mathbf{V}_G))$ a tree-decomposition of width 0.

Let us now assume that G is a hypergraph. One can express it as $G' \oplus G''$ where G'' is a 0-hypergraph consisting of all hyperedges of type 0, of all isolated vertices that are not sources, and of all unary hyperedges that are not adjacent to any hyperedge of type $\geqq 2$, and that are not incident to sources. Note that $K(G)$ is the disjoint union of $K(G')$ and $K(G'')$.

This determination of G' and G'' can be done in time $\mathbf{O}(\mathbf{size}(G))$, and one can also construct a tree-decomposition of G'' of width 0 within this time-bound.

We now consider G' and apply the above algorithm to $K(G')$. By Lemma (2.8), it gives the desired results in time $\mathbf{O}(\mathbf{Card}(\mathbf{E}_{G'}) . \mathbf{Card}(\mathbf{V}_{G'}))$, since for fixed A and n, the cardinalities of the sets of edges of G' and $K(G')$ are linearly related. In particular, if $\mathbf{twd}(K(G')) \leq k$, the algorithm constructs a tree-decomposition (T', f') of $K(G')$ of width at most $5k+9$, and such that $\mathbf{Card}(\mathbf{V}_{T'}) \leq \mathbf{Card}(\mathbf{V}_{G'})$. This tree-decomposition is actually also a tree-decomposition of G' by the proof of Lemma (2.8), and can be extended into a tree-decomposition of G of the same width.

By Theorem (2.2.2), an expression e defining G can be constructed from (T, f). Its size is $\mathbf{O}(\mathbf{Card}(\mathbf{V}_T) + \mathbf{Card}(\mathbf{E}_G))$, hence, is $\mathbf{O}(\mathbf{size}(G))$ (for fixed k). By applying to e the result of [13, Proposition (4.14)], one gets in time $\mathbf{O}(\mathbf{size}(G))$ that either $G = \mathbf{val}(e) \models \varphi$ or that $G = \mathbf{val}(e) \models \neg \varphi$. The overall time complexity is $\mathbf{O}(\mathbf{Card}(\mathbf{E}_G) . \mathbf{Card}(\mathbf{V}_G))$. □

This technique applies to the formulas of *counting monadic second-order logic* introduced in Courcelle [13] (that have special atomic formulas of the form $\mathbf{Card}_{p,q}(X)$ testing whether the cardinality of a set X is equal to p modulo q), and to those of the *extended monadic second-order logic* introduced in Arnbord *et al.* [3].

4. MINORS

We establish that minor inclusion can be expressed in monadic second-order logic, and we use some results of Robertson and Seymour to construct hyperedge replacement grammars generating certain minor closed sets of graphs.

We shall deal only with graphs in this section. The set A of hyperedge labels will consist of symbols all of type 2. Every graph over A is thus directed.

For every such graph G, we denote by $\mathbf{und}(G)$ the underlying, undirected, unlabeled 0-graph. We shall denote by $\mathbf{U}$ the class of undirected, unlabeled 0-graphs with possible loops and multiple edges. The set of finite ones will be denoted by $\mathbf{FU}$.

4.1. Definition: *Minor inclusion.*

Let G and H be finite graphs. We say that H is *included as a minor in* G, or more simply that H *is a minor of* G (this is denoted by $H \trianglelefteq G$) if H can be obtained from G by a finite sequence of edge contractions, edge deletions, and removals of isolated vertices.

The following characterization is equivalent for finite graphs, and can be taken as a definition of minor inclusion for infinite ones. It says that $H \trianglelefteq G$

iff there exist three mappings:

$$f : \mathbf{V}_H \to \mathcal{P}(\mathbf{V}_G)$$
$$f' : \mathbf{V}_H \to \mathcal{P}(\mathbf{E}_G)$$
$$g : \mathbf{E}_H \to \mathbf{E}_G$$

satisfying the following conditions:

(1) for every $v \in \mathbf{V}_H$, $(f(v), f'(v))$ is a connected subgraph of G,

(2) for every $v, v' \in \mathbf{V}_H$ with $v \neq v'$, then $f(v) \cap f(v') = \varnothing$,

(3) g is injective and $g(e)$ belongs to no set $f'(v)$, for any $e \in \mathbf{E}_H$ and $v \in \mathbf{V}_H$,

(4) if $e \in \mathbf{E}_H$ links v and v', then $g(e)$ links a vertex of $f(v)$ and a vertex of $f'(v')$. (We may have $v = v'$.)

Intuitively, H is obtained from G by the contraction of all edges in $f'(v)$, for all $v \in \mathbf{V}_H$, the deletion of all edges neither in $f'(v)$ nor in $g(e)$ for any v and e, and the deletion of all vertices not in $f(v)$ for any v and e, and the deletion of all vertices not in $f(v)$ for any v.

These definitions are independent of labels and orientations so that:

$$H \trianglelefteq G \quad \text{iff} \quad \mathbf{und}(H) \trianglelefteq \mathbf{und}(G).$$

Let us note that we work with abstract graphs. It follows that there is no difference between "H is a minor of G" and "H is isomorphic to a minor of G". It follows also that minor inclusion is a partial order on finite graphs, and not a quasi-order. This is so because, for finite graphs H and G,

$$H \trianglelefteq G \;\Rightarrow\; H = G \quad \text{or} \quad \mathbf{size}(H) < \mathbf{size}(G)$$

[where $\mathbf{size}(G) = \mathbf{Card}(\mathbf{V}_G) + \mathbf{Card}(\mathbf{E}_G)$]. Hence,

$$H \trianglelefteq G \trianglelefteq H \Rightarrow H = G.$$

This argument does not apply to infinite graphs. One can easily construct non-isomorphic infinite trees H and G such that $H \trianglelefteq G \trianglelefteq H$.

If M is a set of finite graphs, $M \subseteq \mathbf{FU}$, we denote by $\mathbf{FORB}(M)$ the set of all finite graphs G such that $H \trianglelefteq G$ for no H in M.

A well-known example is the characterization of planar graphs as $\mathbf{FORB}(\{\mathbf{K}_{3,3}, \mathbf{K}_5\})$ established by Wagner [35] and similar to that of Kuratowski using subgraph homeomorphism instead minor inclusion. Our objective is to express minor inclusion in monadic second-order logic, in order to obtain logical characterizations of sets of graphs of the form $\mathbf{FORB}(M)$.

(4.2) LEMMA: *For every finite graph H, one can construct a formula φ_H in $\mathscr{L}$ defining $\{G/H \trianglelefteq G\}$. If H is a simple loop-free graph, one can construct φ_H without using edge quantifications.*

Proof: Let $\mathbf{V}_H=\{v_1, \ldots, v_n\}$, $\mathbf{E}_H=\{e_1, \ldots, e_m\}$.

A function $f: \mathbf{V}_H \to \mathscr{P}(\mathbf{V}_G)$ can be represented by an n-tuple of sets of vertices $(X_1, \ldots, X_n)$ such that $X_i=f(v_i)$. Similarly, a function $f': \mathbf{V}_H \to \mathscr{P}(\mathbf{E}_G)$ can be represented by an n-tuple $(Y_1, \ldots, Y_n)$ of sets of edges, and g: $\mathbf{E}_H \to \mathbf{E}_G$, by an m-tuple $(z_1, \ldots, z_m)$ of edges. Hence, the existence for a given G of f, f' and g satisfying conditions (1) to (4) of Definition (4.1) can be written as a formula ψ_H of the form $\exists X_1, \ldots, X_n, Y_1, \ldots, Y_n, z_1, \ldots, z_m[\varphi_H]$ where $X_1, \ldots, X_n$ are set variables of sort $\mathbf{v}$, $Y_1, \ldots, Y_n$ are set variables of sort $\mathbf{e}$, and $z_1, \ldots, z_m$ are object variables of sort $\mathbf{e}$. A monadic second-order formula φ_H with free variables $X_1, \ldots, X_n, \ldots, z_m$ expressing Conditions (1) to (4) can easily be written. Its writing uses the fact that the connectedness of a graph can be written in monadic second-order logic (Courcelle [13], Proposition (3.8)]). Hence, for every graph G, $H \trianglelefteq G$ iff $G \models \psi_H$.

We now consider the case where H is simple and loop-free. Conditions (1), (3) and (4) of Definition (4.1) can be replaced by the two conditions:

(1′) for every $v \in \mathbf{V}_H$, the induced subgraph $G \restriction f(v)$ is connected.

(3′) if there is an edge in H linking v and v', then there is an edge in H linking a vertex of $f(v)$ and a vertex of $f(\mathrm{v}')$.

From this formulation, a formula of the form $\exists X_1, \ldots, X_n . \psi'_H$ can be written, where $X_1, \ldots, X_n$ are set variables and ψ'_H expresses the fact that the function $f: \mathbf{V}_H \to \mathscr{P}(\mathbf{V}_G)$ associated as above with $X_1, \ldots, X_n$ satisfies conditions (1′), (2) and (3′). Such a formula can be written without variables denoting edges or sets of edges. It uses atomic formulas of the form $\mathbf{edg}(x, y)$ expressing the existence of an edge between two vertices x and y. $\square$

As an application we get that the class of planar graphs is definable by a monadic second-order formula. That this formula does not use edge quantifications is interesting because some results of Courcelle [12] are applicable to formulas of this latter form, and not to general ones. (*See* also Courcelle [16] for a comparison of the two variants of monadic second-logic with and without edge quantifications.)

In the constructions of Lemma (4.2), the potential minor H of the considered graph G is fixed. Now we give an alternative, more complicated construction where this is no longer the case. We need some notation.

(4.3) Definition: *Let G be a finite or infinite graph.* Let $X \subseteq \mathbf{V}_G$, let Y, $Z \subseteq \mathbf{E}_G$. We say that (Y, Y, Z) *defines a minor of* G if the following conditions hold:

(α) if $x, x' \in X$, $x \neq x'$, then there is no Z-path in G linking them (a *Z-path* is a path all edges of which are in Z; edges can be traversed in any direction);

(β) each end y of an edge in Y is linked to some vertex x of X by some Z-path; (one may have $x=y$;)

(γ) $Y \cap Z = \varnothing$.

Notice that by condition (α), the vertex x in condition (β) is uniquely defined. We shall denote it by $\hat{y}$.

The *minor defined by* the tuple (X, Y, Z) is then H where $\mathbf{V}_H = X$, $\mathbf{E}_H = Y$ and $\mathbf{vert}_H(e) = \{\hat{y}, \hat{y}'\}$ where y and y' are such that $\mathbf{vert}_G(e) = \{y, y'\}$. We shall denote it by $\mathbf{Minor}(X, Y, Z)$. The corresponding mappings f, f', and g of Definition (4.1) are as follows: $f(v)$ is the set of vertices of G linked to v by some Z-path (including v); $f'(v)$ is the set of edges of Z having their ends in $f(v)$; and g is the identity.

Hence, $\mathbf{Minor}(X, Y, Z)$ is indeed a minor of G. It is not hard to see that every minor H of G is of this form for appropriate sets X, Y, Z.

We shall say that $H = \mathbf{Minor}(X, Y, Z)$ is a *strict minor* of G if it is finite and $H \neq G$. This is equivalent to requiring that at least one of the following three conditions holds: $Z \neq \varnothing$, $Y \neq \mathbf{E}_G$, or $X \neq \mathbf{V}_G$.

(4.4) Proposition: *Let Ψ be a monadic second-order formula describing properties of graphs in* $\mathbf{U}$. *One can construct a monadic second-order formula φ defining $\{G/H \trianglelefteq G$, for some H in* $\mathbf{U}$, $H \models \Psi\}$.

Proof: We can take φ to be of the form:

$$\exists X, Y, Z[\varphi_1 \wedge \varphi_2]$$

where φ_1 says that (X, Y, Z) defines a minor of the considered graph, and φ_2 says that this minor satisfies Ψ. The formula φ_1 is just the conjunction of MS-formulas expressing conditions α, β, and γ of Definition (4.3). We now consider φ_2. Let $H = \mathbf{Minor}(X, Y, Z)$ where we assume that φ_1 holds. Then $\mathbf{V}_H = X$, $\mathbf{E}_H = Y$ and the incidence relation between edges and vertices of H can be expressed by an MS-formula. It follows that Ψ can be translated into a formula φ_2 such that $H \models \Psi$ iff $G \models \varphi_2(X, Y, Z)$. $\square$

Note that the first part of Lemma (4.2) is a consequence of this lemma because every finite graph H can be characterized (up to isomorphism) by a first-order formula.

We now come to applications of Lemma (4.2) and Proposition (4.4). A set of graphs is *minor-closed* if it contains all minors of all its members. (This implies that it is closed under relabellings and reversals of orientations of edges).

Robertson and Seymour have established in [30] that if a set L of finite graphs is minor-closed, then it is characterized by a finite set of forbidden minors, *i.e.*, it is of the form $\mathbf{FORB}(M)$ for some finite set M. There is a canonical such set M, called the *obstruction set of* L and characterized as:

$$\mathbf{OBST}(L)=\{H\in \mathbf{FU\text{-}und}(L)/\text{every strict minor of } H \text{ is in } \mathbf{und}(L)\}.$$

We obtain immediately the following result:

(4.5) THEOREM: *If a set of finite graphs is minor-closed, then it is definable.*

Proof: Let L be such a set and $M=\{H_1, \ldots, H_m\}$ be its obstruction set. Then L is defined by the formula $\neg\varphi_{H_1}\wedge\neg\varphi_{H_2}\wedge\ldots\wedge\neg\varphi_{H_m}$, where the formulas φ_{H_i} are as explained in Lemma (4.2). □

Note that in order to construct the formula defining a minor-closed set L of finite graphs (and to use it in algorithms like the ones reviewed above in Section 3), one needs to know the obstruction set of L.

For each $k\geq 4$, the set of graphs of tree-width at most k is minor-closed. We know that the members of the obstruction set are all of tree-width exactly $k+1$, but we do not know yet how to construct them. For $k=3$, there are four, and they have been determined by Arnborg *et al.* [5]. For $k=2$ the list reduces to one, namely $\mathbf{K}_4$ (Wald and Colbourn [36]; *see* also Arnborg and Proskurowski [4]). For $k=1$, there is only one, namely the loop with a single vertex. (Graphs of tree-widths at most 1 are forest having possibly multiple edges.)

We shall now consider sets of finite graphs that are both minor-closed and of bounded tree-width. Quite many interesting examples are of this type: in particular, the classes of graphs of tree-width at most k, for each k. The *tree-width of a set of graphs* is the least upper bound of the tree-widths of its elements.

(4.6) THEOREM: *Let L be a minor-closed set of finite graphs. The following conditions are equivalent:*

(1) $L=\mathbf{FORB}(M)$ *where* $M\subseteq\mathbf{FU}$ *and M contains some planar graph;*

(2) *The tree-width of L is finite;*

(3) *L is context-free.*

Proof: (1)⇔(2) is proved in Robertson and Seymour [26, 27].

$(3) \Rightarrow (2)$ holds by Corollary (2.7).

$(1) \Rightarrow (3)$ The set L is definable by a monadic second-order formula φ by Theorem (4.5). On the other hand, if P is a planar graph, then $\mathbf{twd}(\mathbf{FORB}(\{P\})) \leq k(P)$ where $k(P)$ is a (huge) constant computable from P by [27]. For each k, the set $\mathbf{TWD}(k)$ of finite graphs in $\mathbf{FG}(A)_0$ that have a tree-width at most k is context-free. (This is an easy consequence of the construction of the proof of Theorem (2.2.2) and of Theorem (4.11) of [7].) Hence, $L = \mathbf{TWD}(k(P)) \cap \{G/G \models \varphi\}$ is context-free since the intersection of a context-free set and a definable one is context-free (Courcelle [13, Corollary 4.8.1]). □

Note that condition (1) of this theorem is equivalent to:

(1′) $\mathbf{OBST}(L)$ contains some planar graph.

This result raises effectivity questions that we shall now discuss. Concerning a set of graphs L, one may know the following objects or information concerning it:

– a finite set M such that $L = \mathbf{FORB}(M)$;

– the information that L is minor-closed (denoted: **MC**);

– the obstruction set $\mathbf{OBST}(L)$ when L is minor-closed;

– the information that the tree-width of L is finite
(denoted: **FTW**);

– an upper-bound $k < \infty$ on $\mathbf{twd}(L)$, when **FTW** holds;

– a monadic second-order formula φ defining L;

– an HR-grammar Γ generating L.

In the following proposition, we review the possibilities of determining effectively some of these informations or objects from the others.

(4.7) Proposition: *Let L be a set of finite graphs.*

(1) *From M such that* $L = \mathbf{FORB}(M)$, *one can compute* $\mathbf{OBST}(L)$ *and, of course, condition* **MC** *holds. If in addition M contains a planar graph, one can compute k and* Γ, *and property* **FTW** *also holds.*

(2) *We now assume that condition* **MC** *holds. From* φ *and an upper bound on* $\mathbf{twd}(\mathbf{OBST}(L))$, *one can construct* $\mathbf{OBST}(L)$. *If we assume* **MC** *and* **FTW**, *if we know* φ, *then we can compute k,* $\mathbf{OBST}(L)$, *and* Γ.

(3) *From* Γ, *one can compute k.*

Proof: (1) The graphs in $\mathbf{OBST}(L)$ are minors of the graphs in M. One can thus determine them as the minimal minors of the graphs in M that are not in **FORB**(M). The constructions of k, φ, and Γ when M contains a planar graph are given in the proof of Theorem (4.6).

(2) Let L be minor-closed and defined by an *MS*-formula φ. Then $M_0 := \mathbf{OBST}(L)$ is characterized as $\{H \in \mathbf{FU}\text{-}L/\text{every strict minor of } H \text{ is in } L\}$. It follows that M^0 is the set of graphs H in **FU** such that:

$H \models \neg \varphi$

$H \models \forall X, Y, Z$["if (X, Y, Z) defines a strict minor of H, then **Minor** (X, Y, Z) satisfies φ"].

By the technique used in the proof of Proposition (4.4), one can express the second condition by an *MS*-formula. Hence M_0 is definable, say by a formula ψ. We know that M_0 is finite, but this is not enough to be able to construct it from ψ. We need something more.

We also assume that we know $k' \geqq \mathbf{twd}(M_0)$. From this upperbound and the formula ψ, one can construct an *HR*-grammar Γ' generating M_0. One can decide whether a given *HR*-grammar generates a finite set and construct this set explicitly when it is finite, by a result of Habel [21, Corollary 2.5]. By applying this result to Γ', one can construct M_0.

Let us now assume that L is given by a formula φ and is known to be minor-closed and of finite tree-width. There exists a square grid G_n such that $G_n \notin L$ (because $\mathbf{twd}(L) < \infty$ and L is minor-closed). One can find the smallest one by enumerating them and testing for each n whether $G_n \varphi$. The integer $k = k(G_n)$ gives an upper bound on $\mathbf{twd}(L)$. (Again, we are using the main result of Robertson and Seymour [27].)

Since $\mathbf{twd}(L) \leqq k$, we know that $M_0 := \mathbf{OBST}(L)$ is of tree-width at most $k+1$ (because each edge-contraction, edge-deletion, and vertex removal decreases the tree-width by at most one). Hence, M_0 can be constructed by the first part of the proof.

(3) Immediate consequence of Bauderon and Courcelle [7, Proposition (4.17)]. □

The second part of Proposition (4.7) can also be obtained by the method of Fellows and Langston [20]. They use congruences of finite index on an appropriate algebra of graphs. Every definable set of graphs saturates a finite index congruence relation, as proved in Courcelle [13]. This relation can be effectively determined from a given logical formula, hence the method of [20] is applicable to (4.7.2).

(4.8) *Remarks:* These results show the following in particular. In order to define a minor-closed set of graphs of bounded tree-width, it is equivalent to have the obstruction set or an *MS*-formula, because either of them can be constructed from the other. Both of them can be used to construct an *HR*-grammar. However, we do not presently know how to construct an *MS*-formula or the obstruction set *from the grammar*, though we have no theoretical result saying that this is impossible. The result of Fellows and Langston [19] (*see* also Van Leeuwen [33]) showing that one cannot determine $\mathbf{OBST}(L)$ from a membership algorithm for L does not apply if L is given by a grammar.

There may exist an algorithm that takes an *HR*-grammar Γ and produces a finite set of graphs $A(\Gamma)$ such that, whenever $L(\Gamma)$ is minor-closed, then $A(\Gamma)=\mathbf{OBST}(\mathbf{L}(\Gamma))$. This algorithm would yield another one, producing $\mathbf{OBST}(\mathbf{TWD}(k))$ for every given $k \geqq 1$. The existence of such an algorithm, and *a fortiori* an explicit construction of it, is an open problem. □

The following result is due to Fellows and Langston [20]. We obtain it as an immediate consequence of Proposition (4.7), assertions (1) and (2). The proof of [20] uses a different technique.

(4.9) Corollary: *Let* $L=\mathbf{FORB}(M_1)\cup\mathbf{FORB}(M_2)$ *be given by* M_1 *and* M_2. *If we know an upper bound on* $\mathbf{twd}(\mathbf{OBST}(L))$, *then we can compute* $\mathbf{OBST}(L)$.

Note that L is minor-closed since it is the union of two minor-closed sets. Hence, $\mathbf{OBST}(L)$ is finite and so is $\mathbf{twd}(\mathbf{OBST}(L))$. It is not known how to determine $\mathbf{OBST}(L)$ from M_1 and M_2 without some additional assumption. An upper bound on $\mathbf{twd}(\mathbf{OBST}(L))$ can be obtained as explained in the proof of Proposition (4.7.2) if $M_1 \cup M_2$ contains a planar graph.

We now consider some decidability results obtained by these techniques.

For each k, the set of graphs of tree-width exactly k is context-free because it can be expressed as $\mathbf{TWD}(k)-\mathbf{TWD}(k-1)$, *i.e.*, as the intersection of the context-free set $\mathbf{TWD}(k)$ and the definable set $T(k):=\{G/\mathbf{twd}(G)>k-1\}$. However, we are presently unable to construct an *HR*-grammar generating it, because we do not know the *MS*-formula that defines $T(k)$. It follows that its monadic theory is decidable, but that the corresponding decision is presently *unknown.*

By contrast, the following result shows that, even if we do not know how to construct a formula φ defining a minor-closed context-free set L, nevertheless we can solve some decision problems concerning L that seem, at first sight, to require the construction of φ or at least, of an *HR*-grammar generating L.

We first recall that one cannot decide whether the intersection of two context-free sets of graphs is empty (simply because context-free sets of words can be considered as context-free sets of graphs via an appropriate encoding of words as graphs, and the corresponding problem for context-free languages is undecidable).

(4.10) PROPOSITION: *The emptiness of $L \cap L'$ is decidable if L is a context-free set of graphs (given by an HR-grammar), and L' is minor-closed, given by a membership algorithm.*

Proof: We first recall from Habel that the membership problem for a context-free set of graphs, given by an *HR* grammar, is decidable ([21, Chap. IV, Corollary 1.6]). Hence, the property $L \cap L' \neq \varnothing$ is semi-decidable. We let $\mathbf{D}$ be such L, $L' \subseteq \mathbf{D}$, where either $\mathbf{D} = \mathbf{FU}$ or $\mathbf{D} = \mathbf{FG}(A)_0$ for some finite alphabet A.

We need only show that the property $L \cap L' = \varnothing$ is semi-decidable.

Let $H_1, H_2, H_3, \ldots,$ be an effective enumeration of $\mathbf{D}$. By testing whether each graph H_i belongs to L', one can extract from it an enumeration $H'_1, H'_2, H'_3, \ldots$ of $\mathbf{D} - L'$. For every i, let $M_i = \mathbf{FORB}(\{H'_1, \ldots, H'_i\})$. Since L' is minor-closed,

$$L' = \mathbf{FORB}(\{H'_1, H'_2, \ldots\}) \qquad \text{and} \qquad M_1 \supseteq M_2 \supseteq \ldots \supseteq M_j \supseteq \ldots$$

By the main theorem of Robertson and Seymour [30], we have some j such that $L' = M_j = M_{j+n}$ for all n. (This j is not necessarily the first such that $M_j = M_{j-1}$, and we have no way to determine it in general because one cannot decide whether $L' = R$, for a definable set of graphs R; one can only decide whether $L' \subseteq R$.) For every i, one can test whether $L \subseteq \mathbf{D} - M_i$ (because $\mathbf{D} - M_i$ is definable by a known *MS*-formula). It is clear that $L \cap L' = \varnothing$ iff $L \subseteq \mathbf{D} - M_i$ for some i. Hence, the property $L \cap L' = \varnothing$ is semi-decidable. □

Note added at proof correction. An algorithm computing the obstruction sets of the sets of graphs of tree-width at most k, for $k \geq 4$ has been given in [37]. However, this algorithm is intractable and the sets are still unknown. Nevertheless, they can be computed, at least in principle.

ACKNOWLEDGEMENTS

I thank H. Bodlaender for helpful comments concerning the results of Section 3. I thank K. Callaway who helped me with the English style.

REFERENCES

1. S. Arnborg, D. Corneil and A. Proskurowski, Complexity of finding an embedding in a k-tree, *S.I.A.M. J. Alg. Disc. Methods*, 1987, *8*, pp. 277-284.
2. S. Arnborg, B. Courcelle, A. Proskurowski and D. Seese, An algebraic theory of graph reduction, Report 90-02, Bordeaux-I University, 1990. Short version in L.N.C.S. 532, 1991, pp. 70-83.
3. S. Arnborg, J. Lagergren et D. Seese, Easy problems for tree decomposable graphs, *J. Algorithms*, 1991, *12*, *pp.* 308-340.
4. S. Arnborg and A. Proskurowski, Characterization and recognition of partial 3-trees, *S.I.A.M. J. Alg. Disc. Meth.*, 1986, *7*, pp. 305-314.
5. S. Arnborg A. Proskurowski and D. Corneil, Forbidden minors characterization of partial 3-trees, *Discrete Math.*, 1990, *80*,pp. 1-19.
6. M. Bauderon, Infinite hypergraphs, I, Basic properties, *Theoret. Comput. Sci.*, 1991, *82*, pp. 177-214.
7. M. Bauderon and B. Courcelle, Graph expressions and graph rewritings, *Math. System Theory*, 1987, *20*, pp. 83-127.
8. H. Bodlaender, Classes of graphs with bounded tree-width, Report RUU-CS-86-22, University of Utrecht, The Netherlands, 1986.
9. H. Bodlaender, Polynomial algorithms for Chromatic Index and Graph Isomorphism on partial *k*-trees, Proc. First Scandinavian Workshop on Algorithm theory, 1988, *Lecture Notes in Comput. Sci.*, *318*, pp. 223-232.
10. H. Bodlaender, Dynamic programming on graphs with bounded tree width, Proceedings of ICALP'88, Tampere, Finland, L.N.C.S. 317, 1988, pp. 105-118.
11. H. Bodlaender, Improved self-reduction algorithms for graphs with bounded tree-width, Proceedings of WG'89, *Lecture Notes in Comput. Sci.*, 1990, *411*, pp. 232-244.
12. B. Courcelle, An axiomatic definition of context-free rewriting and its application to NLC graph grammars, *Theoret. Comput. Sci.*, 1987, *55*, pp. 141-181.
13. B. Courcelle, The monadic second-order theory of graphs I: Recognizable sets of finite graphs. *Inform. and Comput.* 1990, *85*, pp. 12-75.
14. B. Courcelle, The monadic second-order logic of graphs II: Infinite graphs of bounded with, *Math. Systems Theory*, 1989, *21*, pp. 187-221.
15. B. Courcelle, The monadic second-order logic of graphs IV, Definability results for equational graphs, *Ann. Pure Appl. Logic*, 1990, *49*, pp. 193-255.
16. B. Courcelle, The monadic second-order logic of graphs VI: On several representations of graphs by logical structures, Research report 89-99, Bordeaux I-University. *Discrete Appl. Math.* (in press). (*See* also *Logic in Comput. Sci.*, 1990. Philadelphia).
17. B. Courcelle, Graph rewriting: an algebraic and logic approach, *in* Handbook of Theoretical computer Science, vol. B, J. van Leeuwen Ed. 1990, Elsevier, pp. 193-242.
18. B. Courcelle and M. Mosbah, Monadic second-order evaluations on tree-decomposable graphs, Rapport 90-110, Bordeaux-I, University, 1990. *Theor. Comput. Sci.*, (to appear).
19. M. Fellows and M. Langston, On Search, decision and the efficiency of polynomial-time algorithms, A.C.M. Symp. on Theory of Computing 1989, pp. 501-512.

20. M. Fellows and M. Langston, An analogue of the Myhill-Nerode Theorem and its use in computing finite-basis characterization, 30th Annual I.E.E.E. Symp. on Foundations of Computer Science, 1989, pp. 520-525.
21. A. Habel, Hyperedge replacement: grammars and languages, Doctoral dissertation, University of Bremen 1989.
22. A. Habel and H. J. Kreowski, May we introduce to you: hyperedge replacement, *L.N.C.S.*, 1987, *291*, pp. 15-26.
23. C. Lautemann, Efficient algorithms on context-free graph languages, ICALP'88, Tampere, Finland, *L.N.C.S.*, 1987, *317*, pp. 362-378.
24. J. Leung, J. Witthof and O. Vornberger, On some variations on the bandwidth minization problem, *S.I.A.M. J. Comput.*, 1984, *13*, pp. 650-667.
25. N. Robertson and P. Seymour, Some new results on the well-quasi-ordering of graphs, *Ann. Discrete Math.*, 1984, *23*, pp. 343-354.
26. N. Robertson and P. Seymour, Graph Minors IV, Tree-width and well quasi-ordering, *J. Combin. Theory*, Ser. B. 48, 1990, pp. 227-254.
27. N. Robertson and P. Seymour, Graph Minors V, excluding a planar graph, *J. Combin. Theory*, Ser. B., 1986, *41*, pp. 92-114.
28. N. Robertson and P. Seymour, Graph Minors X, Obstructions to tree-decomposition, Revised version, Feb. 1988.
29. N. Robertson and P. Seymour, Graph Minors XIII, The disjoint paths problem, Preprint, September 1986.
30. N. Robertson and P. Seymour, Graph Minors XV, Wagner's conjecture, Revised version, March 1988.
31. D. Seese, Ein Unentscheidbarkeitskreiterium, *Wiss. Z. der Humbold Univ. Zu Berlin Math. Natur. Wiss.*, *R24*, 1975, pp. 772-780.
32. D. Seese, The structure of the models of decidable monadic theories of graphs. *Ann. Pure and Appl. Logic*, 1991, *53*, pp. 169-195.
33. J. van Leeuwen, Graph algorithms, Handbook of Theoretical Computer Science, volume A", J. van Leeuwen Ed., 1990, Elsevier, pp. 523-631.
34. W. Vogler, Recognizing edge replacement graphs languages in cubic time, Proceedings of the 4th Int. Workshop on Graph Grammars, Bremen 1990, L.N.C.S., *532*, 1991, pp. 676-687.
35. K. Wagner, Ueber eine Eigenshaft der ebenen Komplexe, *Math. Ann.*, 1937, *114*, pp. 570-590.
36. J. Wald and C. Colbourn, Steiner trees, partial 2-trees, and IFI networks, *Networks*, 1983, *13*, pp. 159-167.
37. J. Lagergren and S. Arnborg, Finding minimal forbiden minors using a finite congruence, L.N.C.S. 510, 1991, pp. 532-543.