An Algebraic Theory of Graph Reduction

STEFAN ARNBORG

The Royal Institute of Technology, Stockholm, Sweden

BRUNO COURCELLE

Bordeaux-1 University, Talence, France

ANDRZEJ PROSKUROWSKI

University of Oregon, Eugene, Oregon

AND

DETLEF SEESE

University of Karlsruhe, Karlsruhe, Germany

Abstract. We show how membership in classes of graphs definable in monadic second-order logic and of bounded treewidth can be decided by finite sets of terminating reduction rules. The method is constructive in the sense that we describe an algorithm that will produce, from a formula in monadic second-order logic and an integer k such that the class defined by the formula is of treewidth $\leq k$, a set of rewrite rules that reduces any member of the class to one of finitely many graphs, in a number of steps bounded by the size of the graph. This reduction system yields an algorithm that runs in time linear in the size of the graph. We illustrate our results with reduction systems that recognize some families of outerplanar and planar graphs.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**], Nonnumerical Algorithms and Problems—*computations on discrete structures*; F.4.3 [**Mathematical Logic and Formal Languages**]; Formal Languages—*classes defined by grammars or automata*; G.2.2 [**Discrete Mathematics**]; Graph Theory—*graph algorithms, trees*

General Terms: Algorithms, Languages, Performance, Theory.

Additional Key Words and Phrases: Graph algebra, graph rewriting, monadic second-order logic, regular set of graphs, treewidth.

S. Arnborg was supported by the Swedish Research Council for Engineering Sciences and the Swedish Board for Technical Development.

B. Courcelle was supported by the "Programme de Recherches Coordonnées: Mathématiques et Informatique" and the ESPRIT-BRA project 3299 "Computing by Graph Transformations".

The research of A. Proskurowski was supported in part by National Science Foundation (NSF) grants CCR 92-13439 and INT 92-14108.

Authors' addresses: S. Arnborg, The Royal Institute of Technology, NADA, KTH, S-100-44, Stockholm, Sweden; B. Courcelle, Bordeaux-1 University, Laboratoire d'Informatique (associé au CNRS), 351 Cours de la Libération, 33405, Talence, France; A. Proskurowski, University of Oregon, CIS Department, Eugene, OR 97403; D. Seese, University of Karlsruhe, Institut fur Angewandte Informatik und Formale Beschreibungsuerfahren, PF 6980, D-7500 Karlsruhe, Germany.

1. *Introduction*

There are several ways to define sets of finite graphs by finite devices. The main ones are graph-grammars [21] (and in particular context-free ones, see, e.g., Courcelle [18, 19]), systems of equations (i.e., their least solutions, see Bauderon and Courcelle [8] and Courcelle [15]), logical formulas (and in particular monadic second-order ones, see Courcelle [16–19] and Arnborg et al. [4]), by forbidden configurations (and in particular forbidden minors, see, e.g., Robertson and Seymour [29], Arnborg et al. [7]), and finally by reduction.

A terminating reduction system (R, K) consists of a rewriting relation $\rightarrow_R$ and a finite set K of accepting graphs. Given any graph G, every sequence of $\rightarrow_R$-rewritings terminates (with a graph called a *normal form* of G). A rewriting system defines a class L of graphs if every normal form of a graph $G \in L$ is in K and if no normal form of $H \notin L$ is in K.

Classical examples of graph reduction concern trees, series-parallel graphs, flowcharts (Hecht and Ullmann [23]). As an example, consider a graph. Remove a pendant edge with its end vertex. Repeat this process until a graph with no pendant edge is obtained. This graph is an isolated vertex if and only if the original graph was a tree.

In this paper, we present an algebraic theory of graph reduction. Our main theorem says that every set of graphs of bounded treewidth that is definable by a monadic second-order formula is also definable by reduction. In addition, the corresponding reduction system can be obtained effectively and a corresponding decision method can be constructed that runs in time linear in the size of the input graph. Although the construction method is intractable in general, it can be applied to specific cases of interest, and we provide examples concerning outerplanar graphs and partial 3-trees.

Membership in the classes of trees, forests (partial 1-trees), two-terminal series-parallel graphs, partial 2-trees, and partial 3-trees can be decided with help of terminating sets of reduction rules [5, 24, 34]. Moreover, using these reduction rules gives an embedding of a partial k-tree in a (full) k-tree for $k = 1, 2, 3$, and this embedding can be produced in linear time in the size of the given graph [28]. The straightforward generalization of this method does not work beyond $k = 3$ [26]. Once an embedding in a k-tree is given for a graph, many combinatorial problems can be solved in time linear in the size of the graph [2–4, 6, 9, 11, 17, 18, 19, 35] (the constant of proportionality depends however on the value of k). An $O(n^2)$ approximate embedding algorithm was developed by Robertson and Seymour [30, 31]. Various improvements are possible (see e.g., Courcelle [17], Lagergren [25], and Bodlaender [12, 13]. A probabilistic and approximate algorithm with $O(n \log n)$ performance was developed by Matoušek and Thomas [28]. Since no linear-time embedding algorithm is known for arbitrary k, this problem is in a sense the bottleneck for fast solution of a large number of combinatorial optimization problems on partial k-trees (also known as graphs of bounded treewidth, see e.g., [29]). The method presented here will make it possible to solve some of these problems in linear time without access to a k-tree embedding, tree-decomposition, elimination order or parse tree.

We develop the theory of rewriting and reduction systems in a universal algebra setting. We consider subsets of the carrier of a many-sorted algebra that are defined as unions of equivalence classes of a congruence with finitely many classes. Membership in such sets can be decided by a rewriting system

allowing a value $f[l]$ to be replaced by $f[r]$, whenever $f[\,]$ is a context and (l, r) is a pair in a finite list of rewrite rules. In general, this rewriting relation will be neither computable nor Noetherian and thus of little value algorithmically. But if there is an integer valued *size* function defined for the carrier of the algebra obeying certain monotonicity properties and such that there are only finitely many objects of each size, then the rewriting system can be modified to a reduction system that is guaranteed to reduce the size of the object on each rewrite step. A reduction system will always rewrite an object from the desired set to one of finitely many representatives of its congruence class, in a number of rewrite steps bounded by the size of the object. We apply the universal algebraic framework to algebras of graphs, as developed in [8], [16], [17], [18], [19], and [20], in such a way that the reduction relation is easily computable on the graph itself and not only on the expression evaluating to the graph. The algebra of graphs that we shall use here is related to the construction of k-terminal recursive graph families [35, 36]. We have a sequence of domains $(\mathbf{G}_i)_{i=0}^{\infty}$ where the domain $\mathbf{G}_i$ is the family of i-sourced graphs, that is, of graphs with a sequence of i distinguished vertices or *sources*. We need only two operations to combine graphs of $\mathbf{G}_i$ into a new graph of $\mathbf{G}_i$, namely the generalized *parallel* and *series* composition. We also need a set of "lifting" operators to add sources to a given graph, other operators to remove sources, and basic nullary operators to introduce vertices and edges. The operator sets for i-sourced graphs give rise to an infinite sequence of finite signatures, $(F_k)_0^{\infty}$, with $F_k \subset F_{k+1}$, where the algebra generated by the signature F_k has a carrier that is the class of graphs of treewidth at most k.

Based on the algebra of i-sourced graphs, we define a rewriting relation on graphs. Namely, given a rewrite rule (l, r) of two i-sourced graphs, an application of the corresponding graph rewriting rule to a graph G first finds a *redex*, a subgraph of G isomorphic to the graph l and such that if the bijection of the isomorphism maps the sources of l to a vertex set s in G and the nonsources to a set i in G, then the nonsourced graph underlying l is isomorphic to the subgraph of G induced by $s \cup i$. It then replaces the redex with an isomorphic copy of r. This leads to a rewriting relation that can be implemented locally on a graph and thus more efficiently than the rewriting relation that depends on all global parses of a graph.

The monadic second order logic was used by Courcelle [16–20] and Arnborg et al. [4] as a powerful tool to formalize graph properties (hereafter, called *MS-definable* properties). We prove that all MS-definable properties of graphs of bounded treewidth can be decided in linear time. We give applications to some families of outerplanar and planar graphs.

2. *Basic Graph-Theoretic and Algebraic Definitions*

2.1 GRAPHS. We consider graphs that are undirected loop-free multigraphs[1] given by finite disjoint sets V and E of vertices and edges, respectively, and the incidence relation $I \subset V \times E$ that is constrained to make every edge incident to exactly two vertices. These two *end-vertices* of the edge are said to be *adjacent*. A *simple graph* is a graph without multiple edges. A *subgraph* H of a graph G, $H \subseteq G$, is a graph that has the sets V_H, E_H, I_H that are subsets of

[1]The extension to graphs with loops, edge labels, edge directions or even to directed hypergraphs is straightforward.

the corresponding sets defining G. A *partial graph* of G is a subgraph of G with the same vertex set as G. A subgraph is *induced* by a set $S \subseteq V$ if it contains all edges of G incident only with vertices in S. A set of m vertices is called a *clique* in G if it induces a complete subgraph of G (the subgraph itself is often called the m-clique and is denoted by K_m if it is simple).

The number of incident edges is called the *degree* of a vertex. A graph is *connected* if there is a path between any two of its vertices. If there is no single vertex that disconnects a graph, the graph is called *nonseparable*. A set of vertices, $S \subset V$, is a *separator* in a graph G if the removal of S and the edges incident to its vertices disconnects G. We use other standard concepts of graph theory presented by Bondy and Murty [14].

A connected graph with no K_{k+2} subgraph and such that every minimal separator (with respect to set inclusion) induces a K_k subgraph is called a *k-tree*. An alternative—and more intuitive—definitionof this class of graphs is given by the following construction process: the $k + 1$-clique is a k-tree, and any k-tree with $n > k + 1$ vertices can be constructed from a k-tree with $n - 1$ vertices by adding a vertex adjacent to all vertices of a K_k subgraph of that k-tree (note that we work with multigraphs and thus allow multiple edges in k-trees). Subgraphs of k-trees are called partial k-trees. A graph G is of treewidth at most k if a family $\{X_n\}_{n \in N}$ of vertex subsets of G can be arranged as nodes in a tree T so that those nodes containing a given vertex induce a subtree of T (i.e., a connected subgraph of T), every pair of adjacent vertices share membership of some X_n, and $|X_n| \leq k + 1$ for all $n \in N$. Such an arrangement is called a *tree-decomposition of width at most k*. The class of partial k-trees is exactly the class of graphs of treewidth at most k (see e.g., [35]).

2.2 ALGEBRAS. Let $\mathcal{S}$ be a finite set of sorts. A set F is a finite $\mathcal{S}$-sorted signature if F is finite and every f in F has a profile $s_1 \times \cdots \times s_\beta \to s$, where β is nonnegative and finite (it may be zero, which corresponds to a constant, i.e., nullary operator), and $s_1, \ldots, s_\beta, s$ are all in $\mathcal{S}$.

An *F-algebra* is a tuple $M = ((M_s)_{s \in \mathcal{S}}, (f_M)_{f \in F})$, where $M_s \cap M_{s'} = \emptyset$ if $s \neq s'$ and f_M is a total mapping $M_{s1} \times \cdots \times M_{s_\beta} \to M_s$ whenever f is of profile $s_1 \times \cdots \times s_\beta \to s$. The sets M_s, $s \in \mathcal{S}$ are its *domains*, their union its *carrier*. We denote the carrier $|M| = \bigcup_{s \in \mathcal{S}} M_s$, as customary. A *finite algebra* is an algebra with a finite carrier.

We denote by $T(F)$ the initial F-algebra (term algebra over F), and write $h_M: |T(F)| \to |M|$ for the unique homomorphism associated with M. Let t be a linear term with variables $x_1, \ldots, x_n$ (i.e., each variable x_i occurs exactly once and has a fixed sort $\sigma(x_i)$ compatible with its usage in t). Let $d_i \in M_{\sigma(x_i)}$ for $1 \leq m < i \leq n$. Then we have a mapping f of profile $\sigma(x_1) \times \cdots \times \sigma(x_m) \to \sigma(t)$ defined by $f(a_1, \ldots, a_m) = t[a_1, \ldots, a_m, d_{m+1}, \ldots, d_n]$. These mappings will be called *derived operations* or, if $m = 1$, *contexts*. If in addition $n = 1$, that is, if t is an expression over F in which only variable x_1 occurs, and it occurs exactly once, then we say that the context is *generated by F*. Note that, if $h_M(|T(F)|)$ is a proper subset of $|M|$, then some contexts may not be generated by F. We shall write $f[\,]$ for a general context. In this way, we need not specify the variable used to represent the argument; $f[t]$ denotes the result of the substitution of t for this variable (and assumes that the resulting expression has correct sorts of subexpressions).

An equivalence relation $\approx$ is *stable* under the operations of M if, for every f_M of M and pairs $(v_i, u_i)_{i=1}^{\beta}$, if $v_i \approx u_i$ for $0 < i \leq \beta$, then $f_M(v_1, \ldots, v_\beta) \approx f_M(u_1, \ldots, u_\beta)$. The class that includes an element d is denoted $[d]$ when the intended equivalence is clear from the context.

A *congruence* on M is an equivalence relation $\approx$ on $|M|$ such that

(i) any two elements equivalent under $\approx$ are of the same sort and
(ii) the relation $\approx$ is stable under the operations of M (and, as a consequence, under the derived operations of M).

Let $L \subset |M|$. We denote by $\sim_L$ the congruence on M defined by: $m \sim_L m'$ if and only if, for every context $f[\,]$, $f[m] \in L$ if and only if $f[m'] \in L$. We say that L is *generated* by F if $L \subset h_M(|T(F)|)$, that is, if every member of L can be written as an expression over F. We say that L is *M-recognizable* if L is a union of classes of a finite congruence on M, all of the same sort.

LEMMA 2.1. *Let M be an F-algebra and $L \subset M_s$ for some $s \in \mathscr{S}$. The following conditions are equivalent*:

(1) *L is M-recognizable.*
(2) *$\sim_L$ is finite.*
(3) *$L = h^{-1}(C)$ where h is an F-homomorphism onto a finite algebra M' and $C \subset M'_s$.*

PROOF

(1) $\Rightarrow$ (2): Every class of the congruence used to recognize L must be contained in a class of the congruence $\sim_L$. So if (2) is violated, every congruence defining L as a union of its classes must have infinitely many classes and L is thus not recognizable.

(2) $\Rightarrow$ (1): If $\sim_L$ is finite, it can be used in the definition of recognizability to define L. The equivalence of (3) is no more difficult, see [17]. □

3. *Rewriting Systems on an Algebra*

A rewriting system R on an algebra M of signature F is a finite list of pairs of elements of $|M|$, $R = \{(l_i, r_i)_{i \in I}\}$, where each r_i is of the same sort as the corresponding l_i. We write $m \rightarrow_{(R,F)} m'$ if and only if $m = f[l_i]$, $m' = f[r_i]$ and $m \neq m'$, for some context f and some $i \in I$. Let $\rightarrow^*_{(R,F)}$ be the reflexive transitive closure of the relation $\rightarrow_{(R,F)}$. For a subset K of $|M|$ and a rewriting system R, we let $L_w((R, F), K)$ be the set $\{m |$ there is an m' such that $m \rightarrow^*_{(R,F)} m'$ and $m' \in K\}$. When the set of operations F is clear from the context, we write $\rightarrow_R$ and $L_w(R, K)$. We say that $L_w((R, F), K)$ is the set *weakly defined* by (R, F) and K.

PROPOSITION 3.1. *Let $L \subset |M|$ be M-recognizable and generated by F. Then, $L = L_w(R, K)$ for some rewriting system R and some finite subset K of L.*

PROOF. L is recognizable and therefore is the union of classes of a finite congruence $\approx$. Let D be a set of representatives, one from each equivalence class of $\approx$. For every $m \in |M|$, we let $\overline{m}$ be the representative of $[m]$. Let us construct a rewriting system R as follows: For every nullary operator f_M of F and such that $\overline{f_M()} \neq f_M()$, we add the pair $(f_M(), \overline{f_M()})$ to R. For every operator f_M of profile $s_1 \times \cdots \times s_\beta \rightarrow s$ and every sequence of representatives $d_1, \ldots, d_\beta$ in D such that d_i is of sort s_i, and $f_M(d_1, \ldots, d_\beta) = d$ is such that

$c[d] \in L$ for some context c, and d is not the representative of $[d]$, we add the pair $(d, \bar{d})$ to R. Let K be the finite set $\{\overline{m} | m \in L\}$. Since the rewriting relation respects the congruence $\approx$, we have $L_w(R, K) \subset L$. On the other hand, every $l \in L$ is the value (under h_M) of some expression $t \in |T(F)|$ since F generates L, and it is not difficult to show by induction over the structure of t that either $l = \bar{l}$ or l can be rewritten into $\bar{l}$ using a sequence of rules and contexts that can be obtained from the syntactic structure of t. □

An explicit membership decision algorithm for L will follow from some assumptions on computability and termination of rewritings.

The relation $\rightarrow_R$ is said to be *Noetherian* if there is no infinite sequence $m_0, m_1, m_2, \ldots$ such that $m_i \rightarrow_R m_{i+1}$ for all $i \geq 0$. An element m' such that $m' \rightarrow_{(R,F)} m''$ for no m'' is called (R, F)-*irreducible* or if (R, F) is clear from the context, *irreducible*. If $m \rightarrow^*_{(R,F)} m'$ and m' is irreducible, then m' is called a (R, F)-*normal form* of m or just a *normal form*.

Let R be a rewriting system and K be a finite subset of $|M|$. We say that $((R, F), K)$ *defines* L and we write this $L = L((R, F), K)$ if the following conditions hold:

(i) $L = L_w((R, F), K)$,
(ii) $\rightarrow_{(R,F)}$ is Noetherian,
(iii) K is a set of (R, F)-irreducibles;
(iv) For every m, and for every (R, F)-normal form m' of m, either $m' \in K$ and $m \in L$, or $m' \notin K$ and $m \notin L$

PROPOSITION 3.2. *Let L, R, K, F be as constructed in the proof of Proposition* 3.1. *Assume also that K is a set of irreducibles and that $\rightarrow_R$ is Noetherian. Then $L = L((R, F), K)$ and every element of L has a unique normal form.*

PROOF. Let $\approx$ be the finite congruence used to construct R. Let $m \in |M|$. Since $\rightarrow_R$ is Noetherian, m has normal forms. Let m' be one of them. Since $\rightarrow_R$ respects $\approx$, $m \approx m''$ for every normal form m'' of m.

If $m' \in K$, then $m \in L$ since $m \approx m' \in K \subset L$. Let m'' be any normal form of m. By the same argument, $m \approx m''$ and $m'' \in K$. But $[m]$ has only one representative in K, so $m' = m''$ and the normal form is unique.

Otherwise, $m' \notin K$. By the preceding case, no other normal form of m is in K, and thus $m \notin L$. □

We now consider how to find a Noetherian $\rightarrow_R$ in an important special case. A *size function* on an algebra M is a mapping $m \mapsto |m|$ associating a nonnegative integer $|m| \in \mathbf{N}$ with every $m \in |M|$. We require that there are only finitely many elements in $|M|$ of each given size and that if $|m| < |m'|$, then $|f[m]| < |f[m']|$ for every context $f[\,]$ (recall that a context contains exactly one occurrence of the variable and thus cannot be constant). Note that this definition requires $f[m]$ to depend on m, so that, for example, annihilators are not permitted in an algebra with a size function.

We call a rewriting system R a *reduction system* if $|r| < |l|$ for every pair (l, r) in R. Obviously, a reduction system is Noetherian. It produces a normal form of m in at most $|m|$ steps.

PROPOSITION 3.3. *If M has a size function, then every recognizable set L in $|M|$ is defined by $((R', F), K)$ for some reduction system R' and some finite set K.*

PROOF. Consider, as in Proposition 3.1, a finite congruence $\approx$ such that L is the union of some of its classes. For each congruence class $[m]$, select one element $\overline{m}$ of minimum size as representative of the class and let D_m be the set of elements in the class that are of minimum size (by our assumptions, each D_m is finite). Let D be the union of the sets D_m. For every sort-compatible combination of an operation f and operands $d_1, \ldots, d_\beta$ from D, consider $d = f(d_1, \ldots, d_\beta)$ such that $c[d] \in L$ for some context c. If $d \notin D$, then put in R' the pair $(d, \bar{d})$. Since D contains all smallest elements of $[d]$, $|\bar{d}| < |d|$ and the system R' will be a reduction system. Also, all normal forms of elements generated by F are in the finite set D. Let $K = D \cap L$. Since the rewriting relation $\rightarrow_{(R',F)}$ respects the congruence, $L_w((R', F), K) = \bigcup_{m \in K}[m]$, and this set is equal to L since K is the set of normal forms in L. By an argument similar to that in the proof of Proposition 3.2, L is defined by $((R', F), K)$ (but normal forms need not be unique). □

The following easy observation will be useful for applications:

PROPOSITION 3.4. *If $L = L(R, K)$ and $R' \subset R$ is such that $G \rightarrow_R^* G'$ for every $(G, G') \in R$, then $L = L(R', K)$.*

4. *Graph Reductions*

A comprehensive account of algebras defined to construct graphs is given in [8]. We will consider an algebraic definition of unlabeled, undirected multigraphs. It is easy to extend or modify this algebra to vertex- and edge-labeled graphs, simple graphs, directed graphs, hypergraphs, and combinations of these.

4.1 GENERATING PARTIAL k-TREES. Let $\mathcal{S}_k$ be the set $\{g_0, \ldots, g_k\}$, where g_i is the sort of *i-sourced graphs*. An *i-sourced graph* is an undirected multigraph and a sequence of i distinct vertices, $G(V, E, I, s)$, where V and E are finite disjoint sets (the vertices and edges, respectively), $I \subset V \times E$ is the incidence relation required to make every edge incident to exactly two vertices, and s:

$\{1, \ldots, i\} \rightarrow V$ is the injective map indicating the jth source for $1 \leq j \leq i$. The graphs of sort g_i form the domain $\mathbf{G}_i$ of i-sourced graphs. A vertex that is not a source will be called an *internal vertex* and a source vertex is also called an *external vertex*. The *underlying graph* of an i-sourced graph is the same graph without its source map, an ordinary graph. Most graph properties of an i-sourced graph can be inherited from the underlying graph in a natural way.

Consider a sequence of operation sets, $(D_i)_{i=0}^{\infty}$. For $i \geq 0$, the set D_i consists of the following operators:

P_i: $\mathbf{G}_i \times \mathbf{G}_i \rightarrow \mathbf{G}_i$; the parallel composition of two i-sourced graphs. It is obtained by fusing corresponding sources of the two i-sourced graphs. P_0 is the special case of disjoint union of two graphs.

l_i^j: $\mathbf{G}_{i-1} \rightarrow \mathbf{G}_i$ for $1 \leq j \leq i$. The lifting of an $(i-1)$-sourced graph to an i-sourced graph by insertion of a new isolated source vertex at position j among the sources.

r_i: $\mathbf{G}_i \rightarrow \mathbf{G}_{i-1}$, $i \geq 1$. This removes the last element from the sequence of sources of an i-sourced graph (but keeps the corresponding vertex).

S_i: $\mathbf{G}_i^i = \mathbf{G}_i \times \cdots \times \mathbf{G}_i \to \mathbf{G}_i$. The series composition of i i-sourced graphs into new i-sourced graph. $S_i(G_1, \ldots, G_i)$ can be defined in terms of the operators l_{i+1}^j, r_{i+1} and P_{i+1} as

$$r_{i+1}\big(P_{i+1}\big(l_{i+1}^1(G_1), P_{i+1}\big(l_{i+1}^2(G_2), \ldots, l_{i+1}^i(G_i)\cdots\big)\big)\big),$$

for $i \geq 2$. Intuitively, each of the operands will have $i-1$ of its i sources identified with $i-1$ sources of the result and one with an $(i+1)$th vertex that is subsequently not regarded as a source. For $i = 1$, $S_1(G_1)$ is defined to have a new vertex that is a source and is connected by an edge with the source of the argument (which is not a source of the result). Thus, $S_1(G_1) = r_2(P_2(e_2, l_2^1(G_1)))$, where e_2 is defined below.

e_i $\in \mathbf{G}_i$ for $i = 1, 2$ is a nullary operator that evaluates to an edge with its two end-vertices. One $(i = 1)$ or both $(i = 2)$ of the end-vertices are sources. (Note that $e_1 = r_2(e_2)$.)

i $\in \mathbf{G}_i$ is the empty graph if $i = 0$. For $i > 0$, $\mathbf{i}$ is a derived nullary operator, evaluating to the i-sourced edgeless graph $l_i^1(\cdots l_1^1(0)\cdots)$.

r_i^*: $\mathbf{G}_i \to \mathbf{G}_0$ is introduced as a derived operator, $r_i^*(G) = r_1(r_2(\cdots r_i(G)\cdots))$, that is, r_i^* is the operator that removes all sources (as sources, not as vertices) from an i-sourced graph.

Thus, for instance, D_0 consists of the operators $\mathbf{0}$ and P_0, D_1 consists of P_1, S_1, l_1^1, r_1, $\mathbf{1}$ and e_1. D_2 consists of the ordinary parallel composition P_2 and a variant of the usual series composition, S_2, as well as l_2^1, l_2^2, r_2, and e_2.

If we were interested in labeled graphs or hypergraphs, more operators would be needed to define labels and hyperedges of different sizes (see [8] and [18]). We let the signature F_k be the union of operator sets D_i for $i = 0, \ldots, k$. We let F_∞ be the union of the signatures F_k. We shall denote by M_k the F_k-algebra with set of sorts $\{g_0, \ldots, g_k\}$ and domains $\{\mathbf{G}_0, \ldots, \mathbf{G}_k\}$. Sometimes, we shall call *parse* of a graph G over a signature F an expression over F that evaluates to G. Let us make precise here that we consider abstract graphs, that is, isomorphism classes of concrete graphs. This is necessary for a correct algebraic treatment.

The following proposition shows that F_k generates a proper subset of the domains of M_k.

PROPOSITION 4.1. *The graphs of sort g_0 generated by F_k are the graphs of tree-width at most k.*

The proposition will follow from these two lemmas:

LEMMA 4.2. *For any term $t \in |T(F_k)|$ evaluating to a graph G, and every partial graph H of G, there is a term $t' \in |T(F_k)|$ which evaluates to H.*

PROOF. Those edges introduced by the nullary operators e_1 and e_2, can be removed in G by replacing a pendant edge e_1 by $l_1^1(r_1(\mathbf{1}))$ and an edge e_2 by $l_2^1(\mathbf{1})$ in t. If an edge was introduced by operator S_1, we can remove it by replacing $S_1(x)$ by $l_1^1(r_1(x))$. □

LEMMA 4.3. *For a K_k-subgraph induced by vertex set K in any k-tree T, there is a term $t \in |T(F_k)|$ that evaluates to the graph T with the vertices in K as sources, in any given order.*

PROOF. Let T be the smallest k-tree in which there is a k-clique K of vertices ordered $(v_1, \ldots, v_k)$, contradicting the hypothesis of the lemma (i.e., there is no term evaluating to T with sources $(v_1, \ldots, v_k)$). T must have more

than $k+1$ vertices (since $r_{k+1}(K'_{k+1}) = S_k(K'_k, \ldots, K'_k)$, where K'_k is K_k with all vertices made sources, and K'_k is generated by F_k). By the construction process of k-trees, there is a vertex v_{k+1} such that K is contained in a $k+1$-clique of T induced by $Q = K \cup \{v_{k+1}\}$. For any $v \in \{v_1, \ldots, v_{k+1}\}$, let $K^v = Q - \{v\}$ and let T_v be the subgraph of T induced by K^v and those vertices separated from v by K^v. For edges between vertices in Q, we require, however, that the edge is present in exactly one of the T_v. The sources of T_i are the vertices K^v ordered as $v_1, \ldots, v_{k+1}$. By the assumption about T, and since T_v is at least one vertex smaller than T, $T_v = h_M(t_v)$ for some t_v in $|T(F_k)|$. Moreover, $t = r_k^*(P_k(S_k(t_{v_1}, \ldots, t_{v_k}), t_{v_{k+1}}))$, a contradiction. □

PROOF (OF PROPOSITION 4.1). By the previous two Lemmas, every partial k-tree is generated by F_k. For an expression $t \in |T(F_k)|$, we can produce a tree-decomposition of width at most k for $h_M(t)$ as follows: The tree of the decomposition is the undirected tree T corresponding to the parse tree of t. The vertex set X_n for node n of T consists of the sources of the value of the subexpression corresponding to n, except if the operation of n is e_1 or S_i, $1 \le i \le k$. In the latter case, the set X_n is the union of the sources of the operands of n (these are merged by the operator S_i, so there are $i+1$ vertices for such a node). In the former case, both end points of the edge will be in the corresponding vertex set. It is easy to show by induction over the structure of t that this gives a tree-decomposition of width at most k for $h_M(t)$. □

COROLLARY 4.4. *An i-sourced graph G_i has a tree-decomposition of width at most k such that all sources of G_i are in one vertex set of the tree-decomposition, if and only if G_i is generated by F_k.*

The *monadic second-order logic* over unlabeled graphs is an extension of the standard first-order predicate logic consisting of individual variables, ranging over vertices and edges, parentheses, logical constants $\wedge$, $\vee$, $\neg$, $\leftrightarrow$, $\rightarrow$, $\forall$, and $\exists$, augmented with set variables, quantification over set variables, and symbols $\in$, $=$, $\subset$. We also use unary predicates E and V, and binary predicate I to denote the sets of vertices and edges, and the incidence relation, respectively. The unary predicate P_i denotes the ith source of the graph.

A k-sourced graph G can now be represented as a relational structure

$$(A, V, E, I, P_1, \ldots, P_k)$$

where A is the set of edges and vertices, V denotes the set of vertices, E the set of edges, I the incidence relation and P_i the ith source (if any). Finitely labeled graphs could be represented by introduction of a finite set of unary predicates. A formula ϕ can be used to define a set of graphs via the satisfaction relation which is a straightforward generalization of the first-order satisfaction relation, (see e.g., [4]). Sets definable by a formula in monadic second order logic will be called MS-definable. More extensive accounts for the logic and its power for defining graph properties can be found in [4] an [16–19]. The concept of MS-definability is very powerful since many NP-hard problems for graphs correspond to MS-definable sets. A substantial list of such problems can be found in Arnborg et al. [4]. As a simple example, consider two-colorability of graphs. It is defined by the last formula below which uses the definition

of adjacency expressed by the first formula:

$$Adj(x, y) \equiv (\neg\, x = y) \wedge \exists e(I(x, e) \wedge I(y, e))$$

$$Twocol \equiv \exists X \exists Y[\forall x V(x) \rightarrow ((x \in X \vee x \in Y) \wedge \neg(x \in X \wedge x \in Y))]$$
$$\wedge\ \neg \exists x_1 \exists x_2((x_1 \in X \Leftrightarrow x_2 \in X) \wedge Adj(x_1, x_2))$$

Courcelle [16, 18, 19] defines a *recognizable* set of graphs L to be a set definable as the union of classes of a congruence with finitely many congruence classes of every sort. The notion of a congruence is defined with respect to the infinite signature F_∞.

THEOREM 4.5 (COURCELLE [16]). *Every monadic second-order definable set of graphs is recognizable.*[2]

Courcelle [16] proved the above theorem by exhibiting a finite congruence over M_k fine enough to recognize all sets definable with a formula of height at most h (the height of logical formula is the largest level of nesting quantifiers). Under this congruence, two graphs G and G' are congruent if, for every monadic second-order formula ϕ with height at most h, $G \vDash \phi$ if and only if $G' \vDash \phi$. We can explain the finiteness of the resulting congruence by observing (and proving, e.g., by induction over h) that there are only finitely many such formulas that cannot be proven equal using renamings of quantified variables and the laws of propositional calculus. An alternative proof using interpretability into classes of labeled binary trees can be found in [4].

4.2 GRAPH REWRITING SYSTEMS. Graph rewriting systems can be either defined as concrete substitution mechanisms (by which a subgraph of a graph is replaced by another graph) or by a rewriting on the algebra of graphs. These two aspects are investigated and shown equivalent in [8]. The graph rewriting systems introduced below are simpler than the most general ones considered in [8], but are sufficient for our purposes.

A *graph rewriting system* is a finite set S of pairs of i-sourced graphs such that two graphs in any pair are of the same sort. We associate with S a binary relation $\Rightarrow_S$ on (0-sourced) graphs defined as follows:

$G \Rightarrow_S G'$ if and only if, for some pair (H, H') in S with H and H' of some sort g_i, there is an i-sourced graph K such that $G = r_i^*(P_i(H, K))$ and $G' = r_i^*(P_i(H', K))$. Intuitively, $G \Rightarrow_S G'$ means that if G can be expressed as a parallel composition of H, a left side in S, and a context K, then H can be replaced by H' (the corresponding right side in S) in that context to form G'. In other words, this means that some $\overline{H}$, isomorphic to H, is a subgraph of G such that the edges and vertices of G not in $\overline{H}$ and the vertices corresponding to sources of H span a subgraph $\overline{K}$ of G, isomorphic to K, (that we shall call the *context* of $\overline{H}$ in G), with the property that the vertices common to $\overline{H}$ and $\overline{K}$ correspond to the sources of H and K in the considered isomorphisms.

PROPOSITION 4.6. *Let S be a graph rewriting system and let k be the smallest number such that every graph occurring among the pairs in S is generated by F_k. For every $k' \geq k$, for all graphs G and G', $G \Rightarrow_S G'$ if and only if $G \rightarrow_{(S, F_{k'})} G'$.*

PROOF. Recall that the contexts involved in the definition of $\rightarrow_{(R, F)}$ are not required to be generated by F. So the necessity is clear. For the sufficiency, we must prove that whenever $G = f[H]$ for a context $f[\,]$, we also have

[2] The first version of the proof of this theorem appeared in 1986.

$G = r_i^*(P_i(H, K))$ for some K. We prove this by induction, for all values of H and i over the depth of the argument place in $f[\,]$. The base case where $f[\,]$ is the context that removes all sources is easy:

$$G = r_i^*(H) = r_i^*(P_i(H, \mathbf{i})).$$

The inductive case has four subcases according to the operator of which H is an argument in the expression:

(i) $G = f[H] = f'[r_{i+1}(H)]$,
(ii) $G = f[H] = f'[l_i^j(H)]$,
(iii) $G = f[H] = f'[P_i(H, K)]$,
(iv) $G = f[H] = f'[S_i(\ldots, H, \ldots)]$.

In each case above, the depth of f' is less than that of f. In case (i), we have by the inductive hypothesis that there is a K' such that $G = r_i^*(P_i(r_{i+1}(H), K'))$. But this can also be written $r_{i+1}^*(P_{i+1}(H, l_i^{i+1}(K')))$ where H occurs as required. Likewise, we have for case (ii) that $G = r_{i+1}^*(P_{i+1}(l_{i+1}^j(H), K')) = r_i^*(P_i(H, K''))$, where K'' is obtained from K' by removal of the jth source (it follows from Corollary 4.9 that any source of a graph can be removed). For case (iii), we have $G = r_i^*(P_i(P_i(H, K), K')) = r_i^*(P_i(H, K''))$, where $K'' = P_i(K, K')$. Finally, the expression of S_i in terms of other operators proves the last case (iv). □

We can take the total number of edges and vertices as the size of a graph. Now the notion of a *graph reduction system* follows from that of a graph rewriting system analogously to the algebraic rewriting systems in Section 3—we only add a size function and require that the left-hand side of every rule has larger size than the corresponding right-hand side. The following is our main characterization result for graph reduction systems.

THEOREM 4.7. *Let L be a recognizable set of graphs of treewidth at most k. Then $L = L(R, K)$ for some graph reduction system R and some finite set $K \subset L$.*

PROOF. It follows from Proposition 4.1 that L is generated by F_k. We consider the results in Section 3 for the F_k-algebra M_k. This algebra has a size function. So it follows from Proposition 3.3 that $L = L(R, K)$ for some finite reduction system R and some finite subset K of L. □

Remark. The sets of graphs L as in Theorem 4.7 are definable by hyperedge replacement (HR) graph grammars. This follows from the closure property of HR sets of graphs with respect to intersection with recognizable sets and from the fact that the set of graphs with tree-width at most k is HR. Some HR sets of graphs are definable by reduction without being recognizable. An example can be constructed from the nonrecognizable context-free language $\{a^n b^n | n \geq 1\}$, which is defined by the reduction system $\{a^2 b^2 \to ab\}$ with the accepting word ab.

PROPOSITION 4.8. *Let $G = f[H]$ where G and H are generated by F_k. Then $G = f'[H]$ for a context $f'[\,]$ generated by $F_{k'}$ and $k' < 2k$.*

PROOF. By the proof of Proposition 4.6 and Corollary 4.4, such a graph G can be written as $r_i^*(P_i(H, K))$ for some i not greater than k. Consider a tree-decomposition of width at most k of G, $(T, \{X_n\}_{n \in N})$. Consider an X_n that contains one source s of H (and thus also of K). Add every source of K except s to every $X_{n'}$, $n' \in N$, and remove all non-source vertices of H from every $X_{n'}$. This results in a tree-decomposition of width at most $k + i - 1 < 2k$

for K. Thus, K is generated by $F_{k'}$ by Corollary 4.4, and so is the context $\lambda x.r_i^*(P_i(x, K))$. □

4.3 Effectiveness of the Main Result. If we are given a recognizable set L of graphs, we can only construct the reduction system for L if we have $\sim_L$ or one of its finite refinements available. It turns out that this is not necessarily the case. As an example, each minor-closed class that excludes a planar graph is recognizable and of bounded treewidth, but an algorithm that constructs representatives of the congruence classes or decides congruence with respect to a given such property is not known.

We can construct the reduction system R and the set K of Theorem 4.7 if L is defined by a known MS-formula φ and if we know an upper bound on the treewidth of graphs in L (but we have no method to decide the existence of such a bound from φ). By the proof of Theorem 4.5 given in [16] one can construct from φ a family of finite sets of formulas $\{\Phi_i\}_{i=0}^k$, with the following property: Let $G \approx G'$ if and only if G and G' are of the same sort g_i, $0 \leq i \leq k$ and for every $\psi \in \Phi_i$ we have $G \models \psi$ if and only if $G' \models \psi$. This equivalence relation is a decidable congruence with respect to F_k, and it is a refinement of $\sim_L$. We can thus construct the (finite) set X of minimum size graphs generated by F_k in each of the (finitely many) congruence classes, since every minimum size graph in a class must be produced by an operator with minimum size arguments. It is now easy to produce R and K using the procedure indicated in the proofs of Theorem 4.7 and Proposition 3.3.

Suppose, on the other hand, that we do not know a bound on the treewidth implied by φ, but we are given L as $\{G | G \models \varphi$, treewidth of G is at most $k\}$. We know by results of Robertson and Seymour [30–32] and Courcelle [17, 19] that the class of graphs of treewidth at most k is MS-definable (by means of a set of forbidden minors) but the corresponding MS-formula, Θ_k, is not known for $k \geq 4$. So L is MS-definable (with the formula $\varphi \wedge \Theta_k$) but we cannot find R and K of Theorem 4.7. It appears difficult to find Θ_k—once it is available one could effectively find the minimal forbidden minors for partial k-trees. It is not enough to know an algorithm deciding the membership in L to be able to construct $\sim_L$. One must also know at least an upper bound on the number of equivalence classes of $\sim_L$, see also Lengauer and Wanke [27]. Fellows and Langston [22] show how to find the minimal forbidden minors from an algorithm deciding $\sim_L$ or one of its finite refinements, for a minor-closed family L and when a tree-width bound is known for these minors.

In the examples that follow, we do not derive the classes automatically. Instead, we start with the nullary operators and generate new values using combinations of old values as arguments to the operators of the algebra. Each time a new value has been obtained, we must decide if it is congruent by $\sim_L$ to one previously obtained, and if so generate a reduction rule. Here, it is fatal to conclude that two graphs are congruent when they are not, but the opposite mistake only results in more classes than strictly necessary. The congruences must ultimately be proved, but often it turns out that there is a small set of congruent pairs that generates all congruences. Only if one infinitely often fails to identify two congruent values as such does this procedure fail to terminate.

4.4 An Example. We illustrate the theory with a simple example. Consider the class L_2 of partial 2-trees and the algebra M_2 as defined above. F_2 generates exactly the class L_2 by Proposition 4.1. But since we allow contexts

not generated by F_2 to distinguish values, $\sim_{L_2}$ is a finer congruence than the trivial one with only one class of each sort. By Proposition 4.8 it is sufficient to consider the class L_2 in the algebra M_3. We find the following equivalence classes of $\sim_{L_2}$, given by their representatives. The class of graphs that are not in L_2 for any context is omitted.

g_0: **0**
g_1: **1**
g_2: $\mathbf{2}, e_2, K_4^-$

Here, K_4^- (K_4 minus an edge) is obtained by regarding two vertices of K_4 as sources and deleting the edge between them. There are quite many classes of sort g_3 that we do not list since we will not need them. We also note that it is not necessary to consider congruence classes not generated by F_2, since F_2 generates the whole class L_2. The reason for this is that if a graph is in a class L generated by a subsignature F' of F, then it is always possible to parse it with respect to F' and to reduce it using only rewrite rules whose both members are generated by F'. In our example, the last class of sort g_2 cannot be generated by F_2 and need not be further considered. This class, however, constitutes the context that under parallel composition, distinguishes the first two classes. The procedure given in the proof of Proposition 3.1 leads to the familiar series-parallel reduction system, $R = ((S_2(e_2, e_2), e_2), (P_2(e_2, e_2), e_2), (S_1(\mathbf{1}), \mathbf{1}), (K_1, \mathbf{0}))$, with four rules for degree 2 vertex elimination, parallel edge elimination, pendant edge removal and isolated vertex elimination, respectively. The construction process of Proposition 3.1 generates also a number of rewrite rules that are immediate consequences of those above, like $(S_2(l_2^1(\mathbf{1}), l_2^1(\mathbf{1})), l_2^1(\mathbf{1}))$ (removal of an isolated vertex on condition that there are two more vertices in the graph), and the same rule can be generated several times.

It will often be convenient to consider a graph algebra generating only i-sourced graphs, for some i, and consider such a graph "equivalent" to the corresponding graph with sources removed by the operator r_i^*. We can find representatives (or all minimum size members) of all congruence classes of sort g_0 if we have them for sort g_i: If S_i is a set containing a minimum size representative (or every minimum size member) from each class of sort g_i, then the set S contains a minimum size representative (all minimum size members) from each class of g_0, where S is the union over i of the set $r_i^*(S_i)$ of graphs from S_i with sources removed, and the set of graphs with fewer than i vertices. The latter set is not finite, since there is no limit on the multiplicity of edges. But in every finite congruence, a minimum size member of a class has a finite bound (usually, 1 or 2) on the multiplicity of an edge.

4.5 A Linear-Time Decision Method. We now describe an algorithm for deciding membership in a recognizable set L of graphs with treewidth bounded by some known number k. The algorithm is based on a graph reduction system that is used to successively update a data structure initially representing a given graph. The results of applications of reduction rules to identified subgraphs isomorphic to left-hand side of rewriting rules are recorded in the data structure. (Each such instance is called a *redex* and the resulting graph a *reduct*.) When no redex can be found, the membership of the irreducible reduct graph in the finite set of accepting graphs determines the membership of the original graph in L.

The distinguishing property of a special graph reduction system admitting a linear-time algorithm is that each left-hand side of a rule is a parallel composition of internally connected graphs with identical sources. We say that an i-sourced graph is *internally connected* if the underlying graph is connected, its source vertices do not constitute a separating set, and there are no edges between sources except when the graph itself is an edge between two sources. The internally connected parts of an i-sourced graph are obtained from the components into which the graph is split by the source vertices, together with the original source vertices as sources. Moreover, in a special graph reduction system not too many of its left-hand-side components with the same source sequence can be composed in parallel without constituting a redex. Finally, let us call a reduction system *auto-reduced* if none of its left-hand sides properly contains a redex. The special reduction system is also auto-reduced. To summarize, a reduction system R is *special* if

(i) Every left-hand side is a parallel composition of a number of internally connected i-sourced graphs, for some $i, 0 \leq i \leq k$,
(ii) There is an upper bound on the number of parallelly composed internally connected parts from left-hand sides of R whose parallel composition is irreducible, and
(iii) R is auto-reduced.

LEMMA 4.9. *Every recognizable set of graphs L of bounded treewidth can be defined by a special reduction system.*

PROOF. We show how such a special reduction system can be obtained from a given recognizable class L. For this purpose, assume that the set D originally contains all smallest elements of every congruence class of $\sim_L$. (We can construct D using the method described in the proof of Proposition 3.3.) We shall construct D' and R, by adding elements starting from the empty set. Let c be the size of a largest element in D. Repeat the following process until it results in no more changes of R and D': Construct all sort-compatible expressions $f(d_1, \ldots, d_\beta)$ with operands from $D \cup D'$ and operators from F_k. Consider the value d of such an expression and split d into its internally connected parts. For an index i denoting a subset of the sources of d, let $\{d_{i,j} | 1 \leq j \leq J_i\}$ be the set of all internally connected parts of d having the same sources. The parallel composition d_i of these graphs is congruent to some $\overline{d_i}$ in D. If $|d_i| = |\overline{d_i}|$, then process the next i or d. Otherwise, first find every rule (e, f) in R such that e is reduced by $\{(d_i, \overline{d_i})\}$ to some e'. Remove each such rule from R and if $|e'| > |f|$ and e' is a parallel composition of internally-connected parts, then add (e', f) to R, else add e' to D'. Now add $(d_i, \overline{d_i})$ to R and apply all rewrites in R to obtain a normal form $\overline{\overline{d}}$ of d, and add $\overline{\overline{d}}$ to D'.

This procedure will terminate, since no element of D' will be larger than $c2^k$ (each of the 2^k source subsets can receive a reduct of size at most c), a rule once removed from R will not be added again, and left-hand sides of rules are obtained from expressions of the form $f(d_1, \ldots, d_\beta)$, with the operands from $D \cup D'$. The class L is defined by $L = L(R, (D \cup D') \cap L)$, and the reduction system is of the required type. Moreover, R can be augmented so that no left-hand side of R consists of more than c internally connected parts and so that every set of c internally connected components from lefthand sides of R has a subset among the left-hand sides of R. □

By a *part* in a graph reduction system R, we subsequently mean an internally connected part of a left-hand side of R. By a *partial match* in G, we mean a part and its isomorphic subgraph of G together with the isomorphism. The following lemma shows that we do not have to find and explicitly remove those partial match indicators that have become invalidated (i.e., are no longer applicable after a reduction step), because a vertex matching one of its sources has disappeared and we never find the same separator in the reduced graph.

LEMMA 4.10. *Let S be a set of partial matches of a special reduction system in a graph G. An application of a reduction rule upon discovery of a redex in S invalidates exactly those partial matches in S that have a source vertex that does not appear in the reduct. Thus, the invalid partial matches will not be referred to in subsequent redex searches.*

PROOF. Restating the lemma, let us assume that a vertex v of G matches an internal vertex of a part R_1 of a redex completed by some partial matches from S. Let us assume further that v matches also a vertex of a part R_2, not a part of the redex, in another partial match of S. We have to show that there is a vertex w in G that matches an internal vertex of R_1 and a source of R_2. In the subgraph of G isomorphic to R_1, consider all paths from v to the vertices that match sources of R_1. If none of them contains a vertex matching a source of R_2, then the vertices of the redex are internal for R_2, which contradicts the auto-reduced property of the special reduction system. Since the parts R_1 and R_2 are internally connected, if the vertices matching sources are identical, so are the matching parts. □

Algorithm 4.1: Membership Decision.
Input: Graph G, given by its adjacency list,
Special reduction system R, given by the list of rules
and the set of accepting irreducible graphs.
Output: YES if the irreducible reduct is an accepting graph, NO otherwise.
Data Structures:
Access structures A_t, $1 \le t \le k$,
indexed by t-tuples identifying vertex sequences
matched to sources in partial matches,
Current reduct graph,
List of low degree vertices,
Array of current vertex degrees.
Method:
1. Initialize data structure: G becomes current reduct graph
and all vertices of low degree are put on the List.
2. **while** non-empty List **do**
Let v be a vertex on the List.
2.1 **while** partial match with v not found **do**
Attempt to match v against an internal vertex not already matched against v
of every internally connected component of a left-hand side
of every rule in R.
2.2 **if** a partial match is found,
then record it in the appropriate A_t with the appropriate source index.
if the partial match completes a redex recognition, perform the reduction by
2.2.1 changing the structure of the reduct graph,
2.2.2 initializing the new vertex adjacencies and degrees,
2.2.3 updating the source degrees, and
2.2.4 inserting new vertices of low degree in the List.
2.3 **if** no match is found, remove v from the List.
3. Match the irreducible reduct graph against the accepting graphs.
if there is a match, output YES, otherwise NO.

LEMMA 4.11. *The Algorithm* 4.1 *is correct.*

PROOF. We show that the algorithm maintains an invariant implying the desired result upon the exit from the outer loop (2).

The invariant of the inner loop (2.1) states that only the vertices on the List, together with the partial matches recorded in the A_is, can match a left-hand side of a reduction rule. This follows easily from the finiteness of the reduction system (and thus bounded degree of the internal vertices of the left-hand sides of its rules).

The invariant of the outer loop states that the current reduct graph is congruent with the input graph and that the data structures are correctly maintained. This follows from Lemma 4.10. □

For our decision algorithm to work in time proportional to the number of vertices of the input graph, we shall need an access structure where a p-tuple of vertex indices is used as key to store the necessary information. The operations are *Store*$(\vec{i}, value)$, *Read*$(\vec{i})$, and *Remove*. The size of the item stored is bounded by a constant that depends only on the set L and its description by a reduction system. First, we investigate the performance achievable. The method is known from Aho et al. [1, exercise 2.12]. We recall it here for the readers convenience.

LEMMA 4.12. *A data structure with access operations Store, Read, and Remove can be implemented so that each operation takes* $O(1)$ *time on a RAM with the uniform cost measure.*

PROOF. The data structure consists of an $O(n^p)$ array indexed by p indices, each ranging over $O(n)$ values, and an $O(n)$ size "verification table" containing indices of initialized elements and stored values. An array element indexed by $\vec{i} = (i_1, \ldots, i_p)$ is retrieved by accessing the $\vec{i}$th entry of the array. An initialized entry of the array contains the offset in the verification table where the index of the entry and the stored value is recorded. If this offset is out of range or the pointed index does not agree with the index of the array entry, then this entry consists of uninitialized noise (nothing has been stored). Storing an element consists of first reading, then either changing or adding an element in the verification table. □

LEMMA 4.13. *Initialization of the data structures used in Algorithm* 4.1 *takes a linear amount of time and establishes the invariants of Lemma* 4.11.

PROOF. The first step (1) of the decision algorithm is to store the edges, vertex degrees and adjacency lists of the instance graph as the current reduct graph. During this phase it is not necessary to store all multiple edges explicitly, since there are numbers a and b, $0 \leq a < b \leq c$ (recall that c is an upper bound on the size of a minimum member of a congruence class), such that b parallel edges are congruent to a parallel edges, that is, at most b parallel edges need ever be stored in the [illegible] table. The access structures A_i are initialized in constant time (cf. Lemma 4.12). The vertices of low degree are placed in the List, which becomes non-empty unless the graph is irreducible. The invariants of the outer and the inner loops are thus trivially established. The total time of this phase is $O(n)$. □

LEMMA 4.14. *Deciding whether a vertex v of a given graph G matches an internal vertex w of a part takes only constant amount of time (when the part is considered constant and G varies).*

PROOF. The matching is done by first matching the internal vertices of the part to vertices in G with the appropriate degrees, and then identifying the vertices matching the sources. Thus, it will never be necessary to search the adjacency list of a high degree vertex in G. A symmetrical part will be matched in many, but only constantly many, ways. A vertex can also only be matched in constantly many ways in all (to all parts). One has to explore isomorphisms of a constant number of vertices with degrees bounded by a constant that depends only on the rewriting system. All vertices of the graph that match internal vertices of the part must have degrees less than some constant and other adjacencies of vertices that match sources of the pattern need not be explored. □

LEMMA 4.15. *Deciding whether a match of a part completes a search for a redex of a special reduction system (as stated in Lemma 4.14) takes constant amount of time.*

PROOF. The access structures A_i should maintain, for each initialized entry, that is, a given sequence of sources, an array of partial match counts for each match type with the same sequence of sources. The match types are defined by left-hand sides of the special reduction system and the count concerns graph vertices corresponding by a match type to part vertices. The counts are checked against the required counts of the left-hand sides every time a new partial match is found. By the property (ii) of the special reduction system, the number of these partial matches is bounded by a constant adding only a constant amount of time needed when recording a partial match, and when rediscovering a redex (which requires updating of the counts and lists of partial matches). □

We are now ready to state our main algorithmic result.

THEOREM 4.16. *Membership in every monadic second order definable set of graphs of bounded treewidth can be decided in time linear in the size of the graph on a RAM with the uniform cost measure.*

PROOF. Let w be the treewidth of the graph class and let n be the number of vertices of the graph. Clearly the size of the graph is $O(n)$ since it has fewer than wn edges. We know that every MS-definable set of graphs is recognizable (Theorem 4.5). Hence, if it is also of bounded treewidth, it is recognized by a graph reduction system (Theorem 4.7), and moreover by a special one (Lemma 4.9). By Lemma 4.11, Algorithm 4.1 based on a special graph reduction system R is correct. It remains to show that the algorithm decides class membership of a graph in linear time.

By Lemma 4.13, the initialization step can be performed in linear time. Since the number of reductions made by a reduction system is linear in the size of the graph, it now suffices to show that there is a constant bound on the time required for each reduction step.

By finding the maximum number of nonsource vertices c_1 in a right-hand side of R, we find a bound $n(1 + c_1)$ on the number of original and new vertices used during the reduction process. We introduce a numbering (internal

naming) for the parts of left-hand sides of reduction rules, and introduce an array A_p indexed by p-tuples for every sort g_p of such a part, where we store the number of partial matches and the indices of the matches to its nonsource vertices. One such table has 2 indices and is used to store the edges of the current reduct graph.

Now we use the arrays to store partial matches to internally connected parts in the graph in linear time. We examine those vertices with a degree identical to the degree of some internal vertex of a part. By Lemmas 4.14 and 4.15, this takes a linear amount of time.

As soon as the multi-set of matches with a given set of source vertices contains a left-hand side of a reduction rule, the reduction can be performed regardless of possibly overlapping matches (cf. Lemma 4.15). The reduction is done as follows: The vertices corresponding to internal vertices of the reduction rule's left-hand side are removed from the graph: Edges involving them are removed and the adjacency lists are updated. Note that the removed vertex is of bounded degree so its adjacency list is of constant length. When updating the neighbors' adjacency lists, we know which element to remove, so no search is involved. Next, the right part of the rule is introduced in the current reduct graph by allocation of new vertex indices for its internal vertices and its edges are introduced in the edge array. The vertices matched into the sources must also be considered as having their neighborhood altered and thus must be regarded as new.

Since the initialization time is $O(n)$ for graph size n and constant work suffices to reduce the graph size by a constant amount, the time required to reach a normal form with respect to R is $O(n)$. If this graph is too large, we reject it (as not belonging to L); otherwise, it is compared to the list of graphs $(D \cup D') \cap L$ in constant time and is accepted if and only if it is found there. □

For a given k, the class of partial k-trees is recognizable. This follows, for example, from the fact that it is minor-closed and thus it has a finite set of minimal forbidden minors (by [29]), and thus it is MS-definable and, by Theorem 4.5, recognizable. Hence, the following corollary:

COROLLARY 4.17. *For every k, membership in the class of partial k-trees is decidable in linear time.*

Remark. The corresponding algorithms are known only for $k \leq 3$.

Of course, our linear-time method, which uses polynomial space is perhaps not what would be chosen in every application. Linear space and $O(n \log n)$ time can also be achieved, with some balanced tree structures like AVL or 2-3 trees [1]. Average case linear time and space can be obtained with hashing techniques [1].

5. *Applications*

5.1 TREES. In the introduction, we have mentioned the well-known process of deciding membership in the class of trees *via* leaf pruning. From our point of view, this process is a rewriting (actually, a reduction) with the single rule contracting a pendant edge. The normal form of any tree is the trivial graph consisting of an isolated vertex; any other graph has a nonreducible graph as a

normal form. The same reduction system can be obtained by applying our formalism to the algebra M_1 with operators $\mathbf{0}$, K_1, P_0, r_1, e_1, $\mathbf{1}$, S_1, and P_1. Building the operation table, we find class representatives $\mathbf{0}$, $v_0 = r_1(\mathbf{1})$, $P_0(v_0, v_0) = d_0$, $\mathbf{1}$, $l_1^1(v_0) = d_1$. The equivalence $e_1 \approx \mathbf{1}$ gives the pendant edge rule, and $R = \{(e_1, \mathbf{1})\}$. The classes d_0 and d_1 do not generate a tree in any context, so we do not have to add the corresponding rules $(P_0(d_0, v_0), d_0)$ and $(P_1(d_1, d_1), d_1)$ to the rewriting system. Thus, $((R, F_1), \{v_0\})$ defines the class of trees.

5.2 OUTERPLANAR GRAPHS. Let us follow up with another example, to our knowledge not considered in the literature. Here, the class of interest is that of *biconnected* (i.e., *nonseparable*) *outerplanar graphs*, P. A graph is outerplanar if and only if it has an embedding in the plane such that all the vertices lie on the border of the unbounded region of the plane ("the outer mesh"). These graphs are isomorphic to convex n-gons with nonintersecting chords. All known algorithms for recognition of these structures (see, for instance [10] and [33]) use the fact that these graphs are Hamiltonian. Below, we present a reduction system based on an algebra generating a superclass of biconnected outerplanar graphs.

Consider the class of 2-sourced graphs $\mathbf{G}_2$, the nullary operation e_2 evaluating to a single 2-sourced edge, and two operations P_2, S_2: $\mathbf{G}_2 \times \mathbf{G}_2 \to \mathbf{G}_2$, denoting respectively the parallel and the series operation with the usual interpretation. We also have the operation r_2^*, source removal, which can only be the outermost operator since no operator takes a 0-sourced graph as argument. For this reason, we can restrict our attention to the congruence classes of sort g_2 and derive those in g_0 by projection. Let us take the declared constant graph e_2 as the representative of the equivalence class 1. Applied to arguments from equivalence class 1 (see Figure 1), the two operations result in, respectively, the two-edge, two-vertex graph, which belongs also to class 1, and the path of length 2, the representative of class 2. $P_2(1,2)$ and $P_2(2,2)$ define the classes 3 and 4, respectively. We take $S_2(1,4)$ as representative of the class 5 that contains every graph that can not be a subgraph of a graph in P. The complete tables for these two operations follow. The P_2 operator is commutative, so we give only half the table. The S_2 operation is not commutative in the graph algebra (the sources should be regarded as an ordered set), but it is so in the quotient algebra as can be seen from the operation table. Script entries correspond to new class representatives (e.g., $P_2(1,2)$ is the representative of class 3) nonscript entries generate reduction rules, by the construction of Proposition 3.2.

P_2	1	2	3	4	5
1	1	*3*	3	4	5
2		*4*	4	5	5
3			4	5	5
4				5	5
5					5

S_2	1	2	3	4	5
1	*2*	2	2	*5*	5
2	2	2	2	5	5
3	2	2	2	5	5
4	5	5	5	5	5
5	5	5	5	5	5

The major task in every such example is to show the equivalence of the result of some operation with some earlier encountered graph. It is clear that

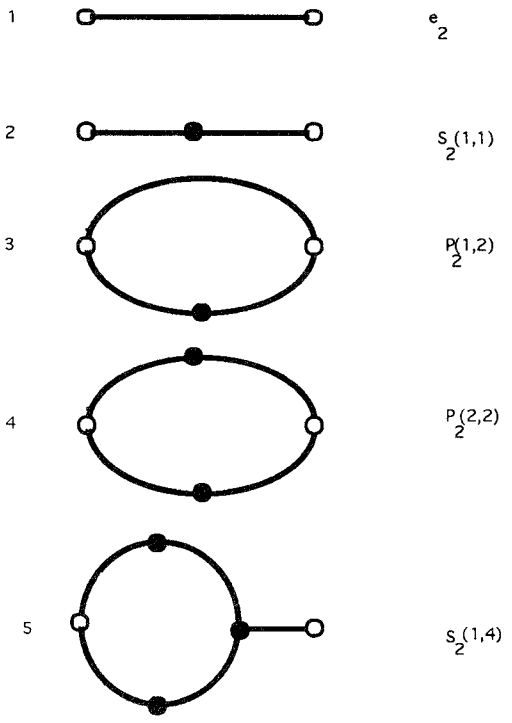


FIG. 1. Equivalence class representatives for biconnected outerplanar graphs.

an edge can be added or removed parallel to an existing one without violating outerplanarity or biconnectedness. Similarly, one can add an edge between vertices with a common degree-2 neighbor, if the local context shows that the added edge cannot be essential for the outer mesh of the graph. Since any two-path has to be included in the outer mesh, such a path can be extended by another degree 2 vertex. We give the five representatives of the equivalence classes of $\sim_P$ of sort g_2 in Figure 1. In this and the following figures, we indicate source 1 by the leftmost unfilled vertex and source 2 by the rightmost such vertex. The pairs of the reduction system, referring to the operations giving rise to new reduction rules (according to Proposition 3.2) is shown in Figure 2. Note that we do not need rules with both sides in class 5, and that some redundant rules are not included. As an example, the rule $P_2(3,3) \to 4$ is redundant because we have

$$
\begin{aligned}
P_2(3,3) &= P_2(P_2(1,2), P_2(1,2)) && \text{(definitions)}\\
&= P_2(P_2(1,1), P_2(2,2)) && \text{(since } P_2 \text{ is associative and commutative)}\\
&\to P_2(1, P_2(2,2)) && \text{(by the first rule of Figure 2)}\\
&\to P_2(2,2) = 4 && \text{(by the second rule of Figure 2)}
\end{aligned}
$$

K is the set of 0-sourced graphs obtainable as a normal form of a minimum size member of one of the accepted classes, with sources removed, and the graphs K_0 and K_1. In this case, 1, 3, and 4 are the accepted classes, and they

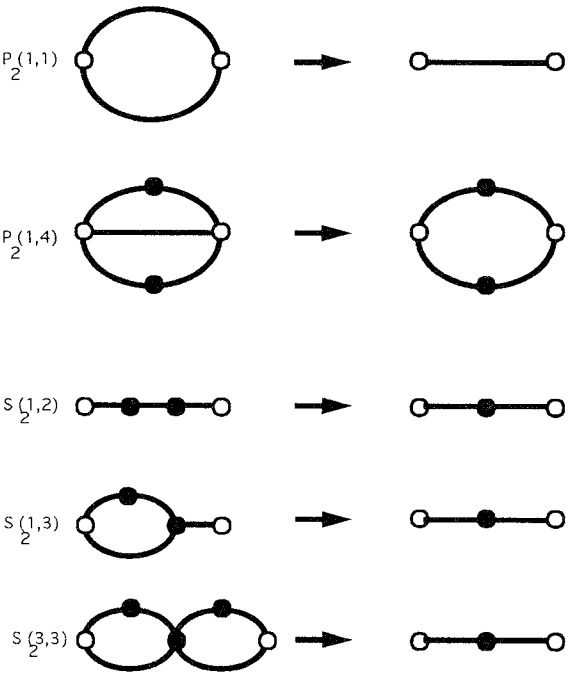


FIG. 2. Reduction rules for biconnected outerplanar graphs.

all have unique minimum-size elements. But the representative of class 4 with sources removed is the cycle of length 4, C_4, which is rewritten by the third rule to C_3 that is the representative of class 3. Thus, in this case K will be $\{C_3, K_2, K_1, K_0\}$; the last two being the only graphs in the class not generated by the operators considered.

Outerplanar graphs that are *not necessarily biconnected* require a slightly different treatment. An algebra generating a superclass of these graphs (excluding the single vertex graph) has, besides the series and parallel operations, P_2 and S_2, two nullary operators **2** and e_2. These graphs are the representatives of classes 0 and 1. Class 2 is defined by $S_2(1,1)$, class 3 by $P_2(2,2)$, and class 4 by $S_2(1,3)$. Class 5 contains all generated graphs that are not outerplanar (for a minor-closed class of graphs, we need only one such congruence class), and is represented by $P_2(1,4)$ (see Figure 3). Below, we give the tables of operations P_2 and S_2. As in the previous example, S_2 is commutative in the quotient algebra.

P_2	0	1	2	3	4	5
0	0	1	2	3	4	5
1		1	2	3	5	5
2			3	5	5	5
3				5	5	5
4					5	5
5						5

S_2	0	1	2	3	4	5
0	0	0	0	0	0	5
1	0	2	2	4	4	5
2	0	2	2	4	4	5
3	0	4	4	4	4	5
4	0	4	4	4	4	5
5	5	5	5	5	5	5

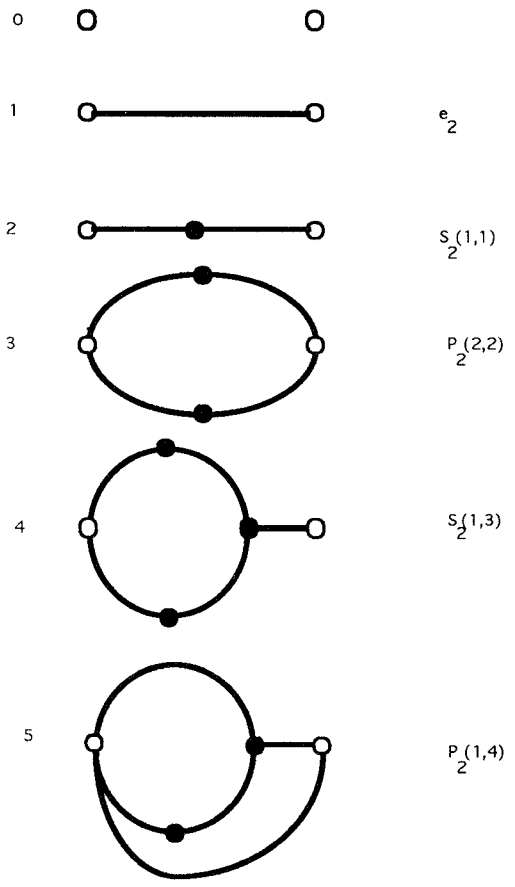


FIG. 3. Equivalence class representatives for outerplanar graphs.

The corresponding classes and reduction rules are given below. As in the previous example, some redundant rules are omitted:

$$
\begin{aligned}
S_2(0,3) &= S_2(0, P_2(1, S_2(1,2))) && \text{(definitions)}\\
&\to S_2(0, P_2(1,2)) && \text{(rule 3 of Figure 4)}\\
&\to S_2(0,2) && \text{(rule 2 of Figure 4)}\\
&= S_2(S_2(1,0),1) && \text{(by inspection)}\\
&\to S_2(0,1) && \text{(rule 7 of Figure 4)}\\
&\to 0 && \text{(rule 7 of Figure 4)}
\end{aligned}
$$

The set K is obtained as in the preceding case. Class 4 has two minimum size representatives, but application of the r_2^* operator makes them equal. There are only two representatives in g_0 of the accepted classes 1, 2, 3, 4: K_2 and its complement. The former represents connected outerplanar graphs, the second disconnected outerplanar graphs.

5.3 PARTIAL 3-TREES. We describe the reduction system presented in [5] and [24] in a new perspective. By a *minor of an i-sourced graph*, we mean the obvious generalization of a minor of an ordinary graph, with the restriction that it is not allowed to contract an edge between two sources. Let $m' \preccurlyeq_L m$ if and only if, for every context $f[\,], f[m] \in L$ implies $f[m'] \in L$. $\preccurlyeq_L$ induces a

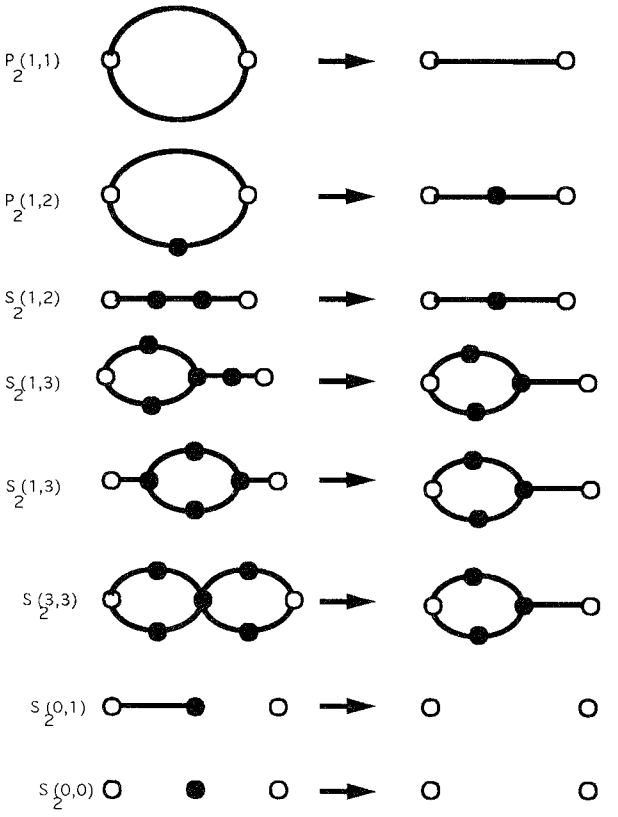


FIG. 4. Reduction rules for outerplanar graphs.

partial order on the congruence classes of $\sim_L$, that is, $m \preccurlyeq_L m'$ and $m' \preccurlyeq_L m$ implies $m \sim_L m'$. Obviously, for a minor-closed class L, $m' \preccurlyeq_L m$ if m' is a minor of m. Let (s_i, κ_i) be a graph rewrite rule where s_i is an i-sourced graph having one nonsource vertex connected with an edge to each of the sources, and κ_i is K_i where each vertex is a source. We say that m' is a *k-star-clique reduct* of m if m' can be produced from m by repeated application of the rule (s_i, κ_i), each with some $i \leq k$. It is easy to see (and proved in [5]) that in such case, when L is the class of partial k-trees, $m \preccurlyeq_L m'$.

We first present an algebra generating all *partial 3-trees* differing slightly from M_3, described previously. Our algebra will have the 3-sourced edgless graph **3** (representing class 0) and the 3-sourced one edge graph $l_3^1(e_2)$ (representing class 1) as values of two nullary operations. In addition to the parallel and series operations, we also define the operation C_3 resulting in cyclic shift of source names ($C_3(C_3(C_3(G))) = G$). In this case, we start by considering the rewrite rule set, given in Figure 5 (in Figures 5 and 6, the top unfilled vertex is source 1, the lower left is source 2, and the lower right is source 3). Clearly, the parallel rule does not change any adjacencies and thus not membership in the class of partial 3-trees either. It is also not difficult to see that for each of the other rules, the right side is both a minor and a 3-star-clique reduct of the left side. Hence, for each rule (l, r), $r \preccurlyeq_L l$ and $l \preccurlyeq_L r$; hence, $l \sim_L r$, where L is now the class of partial 3-trees. In order to see that the system of Figure 5 is complete, one has to produce, from the

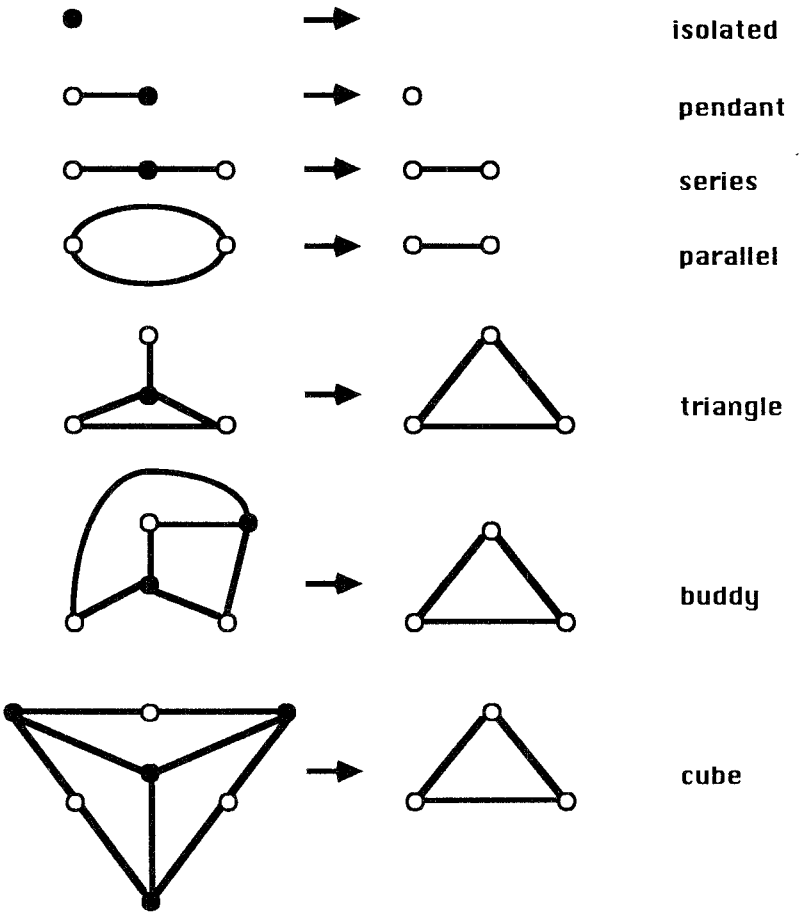


FIG. 5. Reduction rules for partial 3-trees.

nullary operators, all combinations of operators and operands and apply the reduction system to the result. This gives, although with considerable effort, the finite operation tables of the quotient algebra. The representatives of congruence classes are shown in Figure 6. For the set of partial 3-trees, there are thus 9 equivalence classes of 3-sourced graphs generated by the signature. Since the signature generates exactly the class we are interested in, we obtain the class of accepted graphs, K, by taking normal forms of the graphs obtained by removing sources from the representatives of these classes. But applying the reduction rules to these graphs reduces them all to the empty graph **0**.

The representatives for the equivalence classes are the 8 subgraphs of the triangle between three sources ($\mathbf{3}$, $c_1 = l_3^1(e_2)$, $c_2 = l_3^2(e_2)$, $c_3 = l_3^3(e_2)$, $p_1 = P_3(c_2, c_3)$, $p_2 = P_3(c_1, c_3)$, $p_3 = P_3(c_1, c_2)$, $\Delta = P_3(c_1, p_1)$), and $s = S_3(c_1, c_2, c_1)$, the vertex of degree three adjacent to three sources.

The parallel composition of s with itself leads to the buddy reduction rule (cf. [5]). The series composition of three instances of s (recall that S_3: $G_3 \times G_3 \times G_3 \to G_3$) leads to the cube rule. The parallel composition of s with the one-edge graph leads to the triangle rule and the isolated vertex, pendant edge, parallel and series rules are obtained from nontrivial series compositions of the smaller graphs.

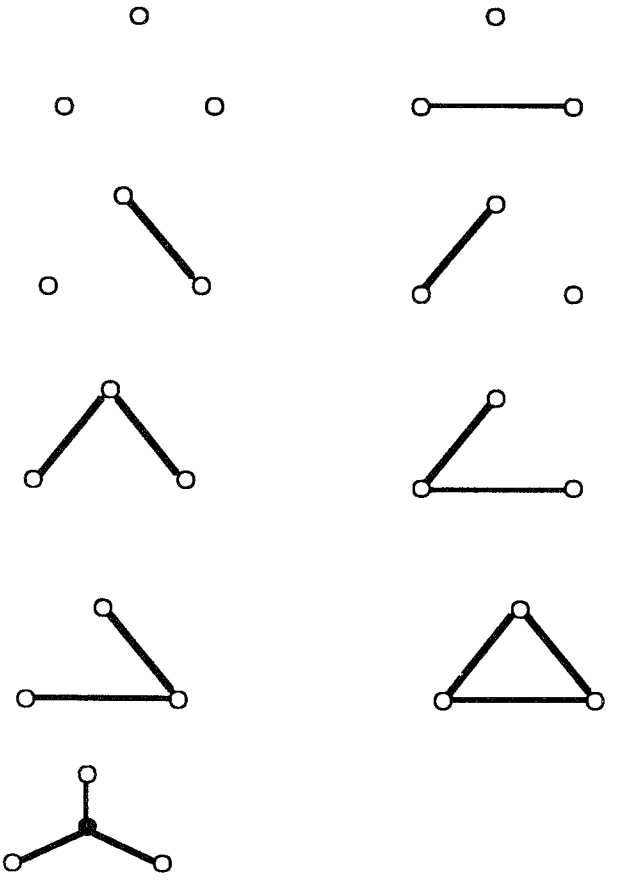


FIG. 6. Representatives for congruence classes for partial 3-trees.

5.4 PLANAR PARTIAL 3-TREES. The signature F_3 generates every partial 3-tree and thus also every planar partial 3-tree. It is not very difficult, although somewhat tedious, to see that all nine graphs of sort g_3 in Figure 6 as well as the 16 graphs of sort g_3 shown in Figure 7 are incongruent with respect to the property under consideration. Most of them restrict the placement of sources in planar embeddings in different ways. The last graph of figure 7 is congruent with the triangle with respect to having treewidth at most k, and with the "star" with respect to planarity. With respect to the intersection of properties, it is congruent to neither graph and yields a new class. The operation table for S_3 would thus consist of at least 25^3 entries, and it appears feasible but unattractive to produce the rewriting system manually in this signature. Somewhat surprisingly, we can get much fewer congruence classes by considering a more complex signature, generating precisely the class of interest.

Let the i-star be the i-sourced graph with one internal vertex connected by an edge to every source (the 3-star is denoted s). Let an *external-planar* graph of sort g_i be a graph of sort g_i such that it is planar and has an embedding in the plane where all sources are on the boundary of a common region (or, equivalently, where all sources are on the outer mesh). Obviously, a graph of sort g_0 or g_1 is planar *iff* it is external-planar. In general, an i-sourced graph is

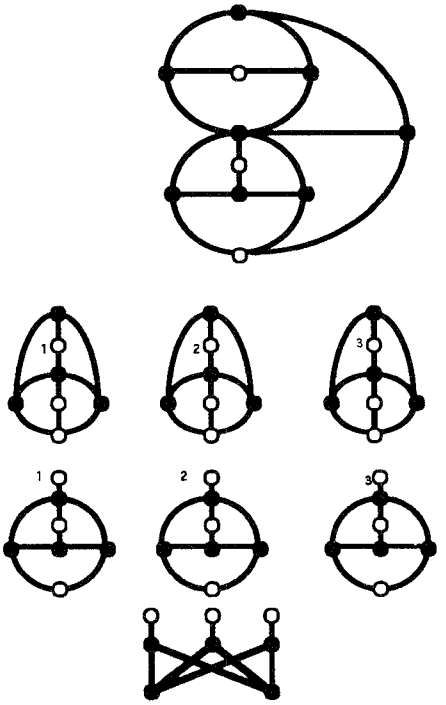
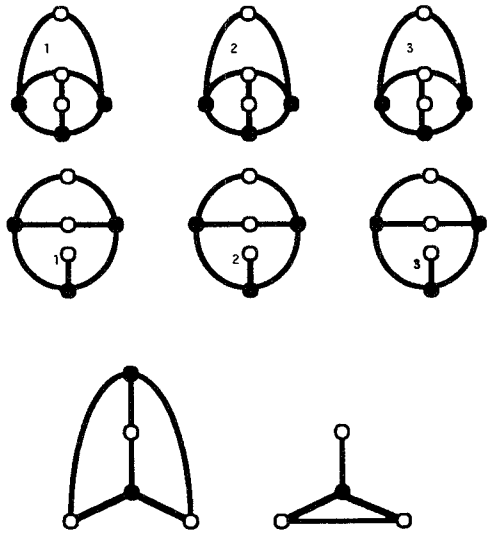


FIG. 7. More congruence class representatives for planar partial 3-trees.



external-planar if and only if its parallel composition with the i-star is planar. A graph of sort g_t is *strongly connected* if it is connected and has at least one internally connected part with i sources. So a K_t of sort g_t is not internally connected or strongly connected if $i > 2$, but an i-star is. We are interested in the family of external-planar and strongly connected graphs generated by F_3. Consider now the signature F_3', consisting of the F_3 operators r_1, r_2, r_3, $\mathbf{1}$, e_2, S_2, S_3 besides the derived operators in Table I.

TABLE I

F_3'	F_3
$//_3^*(x, y)$	$r_3(P_3(x, y))$
$//_1(x, y)$	$P_1(x, y)$
$//_2(x, y)$	$P_2(x, y)$
$//_{2,1}(x, y)$	$P_2(x, l_2^2(y))$
$S_{2,2,2}(x, y, z)$	$S_3(l_3^2(x), l_3^1(y), l_3^2(z))$
$//_{3,1}(x, y)$	$P_3(x, l_3^1(l_2^1(y)))$
$//_{3,2}(x, y)$	$P_3(x, l_3^1(y))$
$S_{3,3}(x, y)$	$S_3(x, y, \mathbf{3})$
$S_{3,2}(x, y)$	$S_{3,3}(x, l_3^2(y))$

An intuitive picture of the operators of F_3' is given by Figure 8. This picture easily leads us to a proof of:

PROPOSITION 5.1. *Every graph generated by F_3' is strongly-connected and externally planar.*

It is more difficult to prove:

THEOREM 5.2. *Every strongly connected external-planar graph generated by F_3 is also generated by F_3'.*

PROOF. Assume the contrary, that is, that there is a smallest i-sourced graph G with largest possible number of sources, that is external-planar, strongly connected and generated by an expression t over F_3 but not by F_3'. If the outermost operator of t is in F_3' (i.e., it is neither l_j^i or P_3), then it has an argument that is not generated by F_3' and is thus not both external-planar and internally connected. Our proof will be by case of the outermost operator of t, and, for each, we show it either impossible or replaceable by an operator later in the list. The case $t = r_3(G')$ is the most complex and is treated separately. Let in this proof *epsc* stand for *external-planar and strongly connected.* Thus, the outermost operator of t is not:

r_1 or r_2: Its argument would be a preferred counterexample;
$\mathbf{1}$ or e_2: These are generated by F_3';
S_3: The arguments must be external-planar. Thus, some of them are not strongly connected. Those arguments can be expressed as $//_2(x, S_2(y, z))$, and t could be expressed with $//_{3,2}$ instead of S_3;
$//_1$: Both arguments must be planar and connected and thus generated by F_3';
$//_2$: Both arguments must be external-planar. If one is not connected, we can generate G using $//_{2,1}$ outermost.
$//_{2,1}$: Both arguments would have to be epsc and thus generated by F_3';
P_3: If both arguments were strongly connected, then G would not be external-planar. So one argument is not strongly connected and we can produce G with $//_{3,2}$ and/or $//_{3,1}$;
$//_{3,2}$: Both arguments must be external-planar. If the second is not connected, then $//_{3,1}$ could be used. If the first is not internally connected, then G would not be so either.
$//_{3,1}$: Both arguments must be epsc;
$S_{3,3}$: Both arguments must be external-planar. If one argument is not strongly connected, then G can be written with S_2 instead; otherwise, both arguments and thus also G would be generated by F_3';

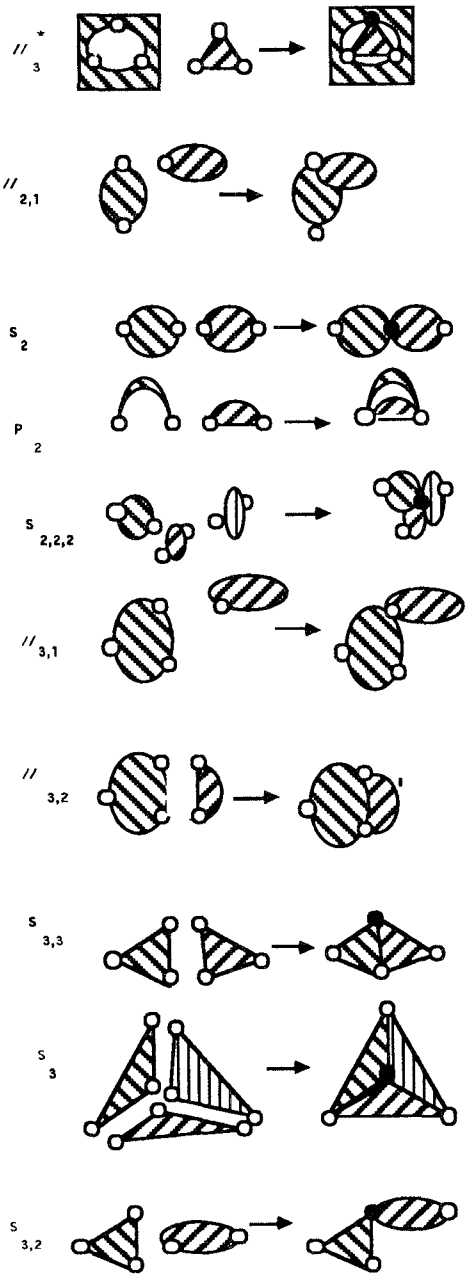


FIG. 8. Operators of algebra F_3'.

S_2: If an argument were not strongly connected, G would be disconnected; If an argument were not external-planar, nor would G be;

$S_{3,2}$: Both arguments must be external-planar. If the first is not strongly connected, then it can be written with $S_{2,2,2}$;

l_1': G would not be connected;

$S_{2,2,2}$: Clearly, all three arguments have to be epsc and thus generated by F_3';

So now the case r_3 is all that remains. Let $G = r_3(G')$. G' is not generated by F_3', since then it would be a preferred counterexample. G is epsc. If G' is

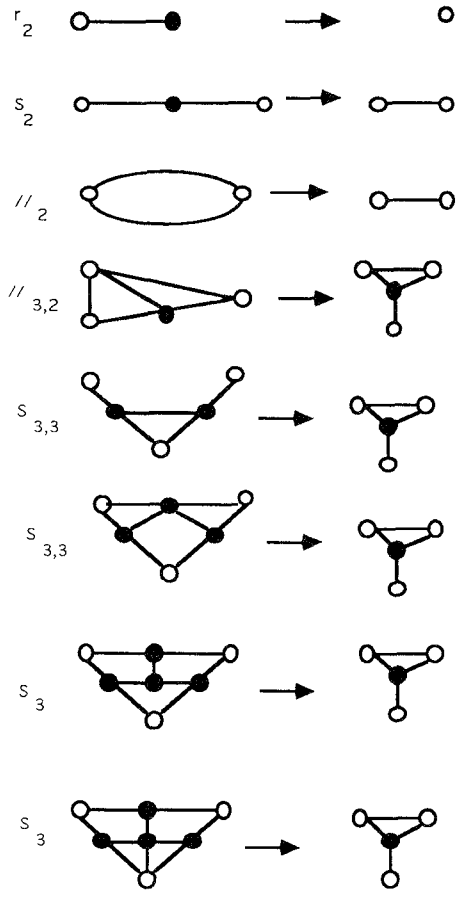


FIG. 9. Rewrite rules for planar partial 3-trees.

not strongly connected, it can be written using either $//_{2,1}$ or S_2 outermost, and these cases are discarded above. Thus, G' is not external-planar, but it is planar and a planar embedding can be constructed so that its first two sources are on the outer mesh. So, $P_3(s, G')$ is not planar (recall s is the vertex of degree 3 adjacent to three sources), but $P_3(\Delta, G')$ is. Thus, the first of these graphs has a K_5 or $K_{3,3}$ minor, but not the second. But since K_5 is 4-regular, it is easy to see that both or neither of these graphs has a K_5 minor, thus the first has a $K_{3,3}$ minor where one vertex and its incidence edges correspond to the s in $P_3(s, G')$, and the sources to the neighbors of the first vertex in the $K_{3,3}$. The rest of the $K_{3,3}$ shows that G' has $P_3(s, s)$ as a minor. Moreover, each s in this expression corresponds to an internally-connected part of G', since any edge between these parts would create a forbidden minor, K_5, of partial 3-trees in $r^*(P_3(s, P_3(s, s)))$ [7]. Therefore, $G = r_3(P_3(G_1, G_2))$, where G_1 and G_2 are 3-sourced epsc graphs smaller than G and thus generated by F_3'. But then $g = //_3^*(G_1, G_2)$; thus, G is generated by F_3'—a contradiction. □

Since F_3' generates only external-planar graphs, only one of the graphs in Figure 7 now remains to consider, the last one. It is now easy to produce the operation tables of the quotient algebra. A complete set of autoreduced rules is shown in Figure 9.

REFERENCES

NOTE: Reference [15] is not cited in text.

1. AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. Design and analysis of computer algorithms. Addison-Wesley, Reading, Mass., 1974.
2. ARNBORG, S. Efficient algorithms for combinatorial problems on graphs with bounded decomposability—A survey. *BIT 25* (1985), 2–33.
3. ARNBORG, S., CORNEIL, D. G., AND PROSKUROWSKI, A. Complexity of Finding Embeddings in a k-tree. *SIAM J. Alg. Discr. Meth. 8* (1987), 277–287.
4. ARNBORG, S., LAGERGREN, J., AND SEESE, D. Easy problems for tree-decomposable graphs. *J. Algorithms 12* (1991), 308–340.
5. ARNBORG, S., AND PROSKUROWSKI, A. Characterization and recognition of partial 3-trees. *SIAM J. Alg. Discr. Meth. 7* (1986), 305–314.
6. ARNBORG, S., AND PROSKUROWSKI, A. Linear time algorithms for NP-hard problems on graphs embedded in k-trees. *Discr. Appl. Math. 23* (1989), 11–24.
7. ARNBORG, S., PROSKUROWSKI, A., AND CORNEIL, D. G. Forbidden minors characterization of partial 3-trees. *Discr. Math. 80* (1990), 1–19.
8. BAUDERON, M., AND COURCELLE, B. Graph expressions and graph rewritings. *Math. Syst. Theory 20* (1987), 83–127.
9. BERN, J. A., LAWLER, E., AND WONG, A. Linear time computation of optimal-subgraphs of decomposable graphs. *J. Algorithms 8* (1987), 216–235.
10. BEYER, T., JONES, W., AND MITCHELL, S. Linear algorithms for isomorphism of maximal outerplanar graphs. *JACM 26*, 4 (Oct. 1979), 603–610.
11. BODLAENDER, H. L. Dynamic programming on graphs with bounded treewidth, In *Proceedings of the 15th ICALP*. Lecture Notes in Computer Science, vol. 317. Springer-Verlag, New York, 1988, pp. 105–118.
12. BODLAENDER, H. L. Improved self-reduction algorithms for graphs with bounded treewidth. *Ann. Discr. Math.*, to appear.
13. BODLAENDER, H. L., AND KLOKS, T. Better algorithms for the path width and tree width of graphs. In *Proceedings of the 18th ICALP*. Lecture Notes in Computer Science, vol. 510. Springer-Verlag, New York, 1991, pp. 544–555.
14. BONDY, J. A., AND MURTY, U. S. R. *Graph Theory with Applications*. North Holland, Amsterdam, The Netherlands, 1976.
15. COURCELLE, B. Equivalence and transformation of regular systems. Applications to recursive program schemes and grammars. *Theoret. Comput. Sci. 42* (1986), 1–22.
16. COURCELLE, B. The monadic second order logic of graphs. I: Recognizable sets of finite graphs. *Inf. Comput. 85* (1990), 12–75.
17. COURCELLE, B. The monadic second order logic of graphs. III: Tree-decompositions, minors, and complexity issues. *Informatique Théorique et Applications 26*, (1992), 257–286.
18. COURCELLE, B. Graph rewriting: an algebraic and logical approach. In *Handbook of Theoretical Computer Science, vol. B*. J. B. van Leeuwen, ed. Elsevier, New York, pp. 194–242.
19. COURCELLE, B. The monadic second-order logic of graphs: Definable sets of finite graphs. In *Lecture Notes in Computer Science*, vol. 344. Springer-Verlag, New York, 1989, pp. 30–53.
20. COURCELLE, B. Graphs as relational structures; an algebraic and logical approach. In *Proceedings of the 4th International Conference on Graph Grammars and their Application to Computer Science*. Lecture Notes in Computer Science, vol. 532. Springer-Verlag, New York, 1991, pp. 238–252.
21. EHRIG, H., NAGL, M., ROZENBERG, G., AND ROSENFELD, A. EDS. *Proceedings of the 3rd International Workshop on Graph Grammars and Their Application to Computer Science*. Lecture Notes in Computer Science, vol. 241. Springer-Verlag, New York, 1987.
22. FELLOWS, M., AND LANGSTON, M. An analogue of the Myhill–Nerode theorem and its use in computing finite basis characterizations. In *Proceedings of 30th Symposium on Foundations of Computer Science*. IEEE, New York, 1989, pp. 520–525.
23. HECHT, M., AND ULLMANN, J. Flow graph reducibility. *SIAM J. Comput.* 1 (1972), 188–202.
24. KAJITANI, Y., ISHIZUKA, A., AND UENO, S. Characterization of partial 3-trees in terms of 3 structures. *Graphs Combin. 2* (1986), 233–246.
25. LAGERGREN, J. Efficient parallel algorithms for tree-decomposition and related problems. In *Proceedings of 31st Symposium on Foundations of Computer Science*. IEEE, New York, 1990, pp. 173–182.

26. LAGERGREN, J. On the non-existence of reduction rules finding an embedding in a k-tree for $k > 3$. *Ann. Discr. Math* , to appear.
27. LENGAUER, T., AND WANKE, E. Efficient analysis of graph properties on context-free graph languages. In *Proceedings of ICALP '88*. Lecture Notes in Computer Science, vol. 317. Springer-Verlag, New York, 1988, pp. 379–393.
28. MATOUŠEK, J., AND THOMAS, R. Algorithms finding tree-decompositions of graphs. *J. Algorithms 12* (1991), 1–22.
29. ROBERTSON, N., AND SEYMOUR, P. D. Some new results on the well-quasi ordering of graphs. *Ann. Discr. Math. 23* (1987), 343–354
30. ROBERTSON, N., AND SEYMOUR, P. D. Graph Minors X, Obstruction to tree-decomposition. *J. Combin. Theory, Ser. B. 52* (1991), 153–190.
31. ROBERTSON, N., AND SEYMOUR, P. D. Graph Minors XIII, The Disjoint Path Problem. Preprint.
32. ROBERTSON, N., AND SEYMOUR, P. D. Graph Minors XIV, Wagners conjecture. Preprint.
33. SYSLO, M. M. Linear time algorithm for coding outerplanar graphs. In *Proceedings of the International Colloquium on Graph Theory and Its Applications* (Apr. 1977). Ilmenau, GDR, 1978, pp. 259–269.
34. WALD, A., AND COLBOURN, C. J. Steiner trees, partial 2-trees, and minimum IFI networks. *Networks 13* (1983), 159–167.
35. WIMER, T. V. Linear algorithms on k-terminal graphs, Ph.D. dissertation Clemson Univ., Aug. 1987.
36. WIMER, T. V., HEDETNIEMI, S. T., AND LASKAR, R. A methodology for constructing linear graph algorithms, DCS. Clemson University, Clemson, S.C., Sept. 1985.

RECEIVED FEBRUARY 1990; REVISED JUNE 1991; ACCEPTED JULY 1992