JOURNAL OF ALGORITHMS 12, 308–340 (1991)

# Easy Problems for Tree-Decomposable Graphs*

STEFAN ARNBORG AND JENS LAGERGREN

*Department of Numerical Analysis and Computing Science, The Royal Institute of Technology, S-100 44 Stockholm, Sweden*

AND

DETLEF SEESE†

*Akademie der Wissenschaften der DDR, Karl-Weierstrass-Institut für Mathematik, Mohrenstrasse 39, D-O-1C86 Berlin, Germany*

Received May 10, 1988; revised January 24, 1990

Using a variation of the interpretability concept we show that all graph properties definable in monadic second-order logic (MS properties) with quantification over vertex and edge sets can be decided in linear time for classes of graphs of fixed bounded treewidth given a tree-decomposition. This gives an alternative proof of a recent result by Courcelle. We allow graphs with directed and/or undirected edges, labeled on edges and/or vertices with labels taken from a finite set. We extend MS properties to extended monadic second-order (EMS) problems involving counting or summing evaluations over sets definable in monadic second-order logic. Our technique allows us also to solve some EMS problems in linear time or in polynomial or pseudopolynomial time for classes of graphs of fixed bounded treewidth. Moreover, it is shown that each EMS problem is in NC for graphs of bounded treewidth. Most problems for which linear time algorithms for graphs of bounded treewidth were previously known to exist, and many others, are EMS problems. © 1991 Academic Press, Inc.

## 1. INTRODUCTION

Quite a few NP-complete computational problems on graphs have been shown to have linear-time algorithms on restricted families of graphs. While some instances of such "algorithms" are implicit in applied research

*Research supported by the Swedish Natural Sciences Research Council and the Swedish Board for Technical Development.

†Present address: FB 11/Computer Science, Univerity Dulsburg, Pf 10 15 03, C-W-4100, Dulsburg 1, Germany.



0196-6774/91 $3.00
Copyright © 1991 by Academic Press, Inc.
All rights of reproduction in any form reserved.

as far back as the middle of the nineteenth century (for a survey see [3]), and there are clear analogies with the theory of non-serial dynamic programming as described by Bertele and Brioschi [9], the systematic study of this problem had of course to await the recognition of the NP-completeness concept.

This research area started in the 1970s with a large number of papers describing polynomial time algorithms for NP-complete graph problems that worked when the graph was restricted to be a tree or a series-parallel graph. A fairly general characterization of problems decidable in linear time on series-parallel graphs was presented by Takamizawa, Nishizeki, and Saito [40]. Now the search started for the best generalization of series-parallelness.

It turned out that there are many classes of graphs whose structure is so closely related to trees, that one can use this fact to solve problems for them whose solutions are unknown or difficult for arbitrary graphs. A highlight in this research is the fundamental series of papers of Robertson and Seymour (see, e.g., Johnson [28] for a survey) on graph minors. Robertson and Seymour introduced in [31] the notion of treewidth of a graph. Intuitively, a graph of treewidth $w$ can be expressed as the (nondisjoint) union of graphs of size $w + 1$ arranged as nodes in a tree in such a way that the set of nodes containing a given vertex is a subtree (i.e., a contiguous part) of the tree. The treewidth is thus a parameter that measures how close to a tree a graph is. Its exact definition will be given in Section 3.

The concept of treewidth and similar concepts found deep applications in the design of (at least theoretically) efficient algorithms (see, e.g., Johnson [28]). Many important classes of graphs have a universal bound for the treewidth of their members. As an example, trees and forests have treewidth $\leq 1$, series-parallel graphs and outerplanar graphs $\leq 2$, almost trees with $k$ additional edges have treewidth $\leq k + 1$, Halin graphs $\leq 3$, and members of $k$-terminal recursive graph families[1] have treewidth $\leq k$. More information on these classes can be found in [11, 27].

At the same time, similar graph measures were found and suggested as a generalization of being series-parallel, and also linear time algorithms were found for these graph families and other previously known families of graphs. For a survey of this activity see Johnson [27]. Particularly, the property of being a subgraph of a $k$-tree was suggested in [6] and shown equivalent to having treewidth $\leq k$ by Scheffler [37] and Wimer [43]. A

[1]See [43]; the statement assumes that all the basis graphs in the definition, with edges added between all pairs of terminals, also have treewidth $\leq k$.

$k$-tree, for a natural number $k$, is either

(i) a graph which can be obtained from another $k$-tree by addition of a vertex with $k$ edges connecting it to $k$ pairwise adjacent vertices in the other $k$-tree, or

(ii) a $k$-clique (a graph of $k$ vertices and all possible $\binom{k}{2}$ edges).

Seese suggested the notion of tree-partite graph [39], which is closely related to the treewidth concept.

There are many approaches to finding a "template" for the design of linear time algorithms for various problems on these graphs, from giving a list of examples with informal principles [6, 8] to definition of a language in which the property must be expressed [10, 19, 39] as well as some intermediate approaches [44]. The first systematic attempts at finding such a language were based on the existential locally verifiable (ELV) concept, essentially second-order properties with existential quantification and no complementation, see Seese [39]. Courcelle also considered problems with universal quantification and showed that all properties expressible in monadic second-order logic on graphs of bounded treewidth can be decided in linear time [19] if a parse of the graph is given. A more algebraic approach was used by Bodlaender in [10] to define the two classes ECC and LCC.

Monadic second order logic is a powerful language which contains the propositional logic connectives, $\wedge$, $\neg$, $\vee$, $\rightarrow$, and $\leftrightarrow$ (with the usual meanings: and, not, or, implies, if, and only if, respectively), individual variables $x, y, z, x_1, x_2, \ldots$, existential ($\exists$) and universal ($\forall$) quantifiers, and predicates. Moreover, it contains (and is distinguished from first-order logic by) set variables $X, Y, Z, U, X_1, X_2, \ldots$, the membership symbol $\in$, and it allows existential and universal quantification over set variables. Mathematical logic has traditionally been concerned with the "intrinsic validity" or "satisfiability" of statements in the form of logical formulae. In order to do so, one has introduced structures as representatives of possible worlds, and the satisfaction relation $\models$ between a structure and a formula. A structure is a set (of objects in the universe) and, for every predicate symbol in the logical language under consideration, a table of the tuples of objects satisfying it. In order to show that a formula is "intrinsically valid," a logician must directly or indirectly show that the formula is satisfied by every structure compatible with the logical language, and in order to show that it is satisfiable he must show that it is satisfied by some structure.

Instead, we want to use monadic second-order logic sentences to define problems on finite graphs. Solving a problem for a given graph will then correspond to deciding if a given finite structure (that represents the

graph) satisfies a given sentence (that defines the problem). As an example, the property that a subgraph of $G$ induced by a set $Z$ is connected can be stated as:

$$\begin{aligned}\text{Partition } 2(U,V,Z) &\equiv (Z = U \cup V) \wedge (U \cap V = \varnothing) \\ &\quad \wedge (U \neq Z) \wedge (V \neq Z) \\ \text{Adjacent}(U,V) &\equiv \exists u \exists v (v \in V \wedge u \in U \wedge \text{Adj}(u,v)) \\ \text{Connected}(Z) &\equiv \forall U \forall V \text{ Partition } 2(U,V,Z) \rightarrow \text{Adjacent}(U,V),\end{aligned}$$

where lower case variables range over vertices and upper case variables range over vertex sets, and *Adj* is the adjacency relation of the graph. Here we used set equality and some set operators freely, but they are easy to define in monadic second order logic (e.g., $(Z = U \cup V) \equiv \forall x\ (x \in Z \leftrightarrow x \in U \vee x \in V)$, $(U \cap V = \varnothing) \equiv \neg \exists x\ (x \in U \wedge x \in V)$). As an exercise the reader can easily find monadic second-order formulas (MS formulas) $\Phi(X)$ and $\Psi(X)$ which hold in a graph $G$ for a set of edges $B$, i.e., $G \models \Phi[B]$ or $G \models \Psi[B]$, where $B$ is the interpretation for $X$, if and only if the subgraph in $G$ induced by $B$ is a path or a cycle, respectively. In later applications we consider mixed graphs (see Section 3), whose domains contain vertices as well as edges, and which contain an incidence relation, unary predicates to designate vertices from edges and unary predicates corresponding to vertex and edge labels. The individual variables of the corresponding monadic second-order logic range then over vertices and edges and the set variables range over sets of vertices and edges, i.e., all possible subsets of the domain of the graph considered.

Many NP-complete computational problems on graphs can be formulated as MS properties, i.e., properties expressible by MS formulas. This is interesting for two reasons. The first is that it is in most cases very easy to verify that a property $P$ is a monadic second-order property, just by writing the corresponding definition of $P$ in a formal way. The second is that once a property has been so defined, it can be decided in linear time for graphs of bounded treewidth given with their corresponding tree-decompositions. So it is, e.g., not difficult to see that Courcelle's result [19] subsumes Takamiza, Nishizeki, and Saito's [40] theorem on properties definable by a finite number of forbidden subgraphs. However, sometimes the formulation is not the one that comes first to mind. As an example, the problems *Hamiltonian circuit* and *covering by triangles*[2] can be thought of as asking for a permutation of vertices and a large number of vertex sets, respectively. The most obvious strengthenings of monadic second-order

[2]Deciding if the graph is a non-disjoint union of triangles.

logic that would allow such formulations, such as quantification over functions or sets of sets, would also allow definition of problems that are NP-complete on trees, like bandwidth or subgraph isomorphism. But the problems above can be formulated in monadic second-order logic as the existence of a 2-regular partial graph of the problem graph (in MS terminology, a set of edges with exactly two of them incident to each vertex) which is connected (*Hamiltonian cycle*) and whether each edge is part of a 3-cycle (*covering by triangles*). Other limitations are more serious. In many cases one asks for a set with a certain property (e.g., an independent set with a prescribed size that is part of the problem instance). In the current logic and for tree-decomposable graphs we can only code this requirement into the formula in such a way that we obtain different formulas for different sizes. Thus we must move the prescribed size into the problem definition from the problem instance description. Also in this case, the most obvious generalization allowing us to keep the constants within the problem instance description would allow us to define problems that are not solvable by finite-state tree automata as we will define them in Section 5. However, many problems can be defined in terms of sums of evaluations (vertex and/or edge weight functions denoted by $f_j$ for $j = 1, \ldots, m$) over subsets definable in monadic second-order logic, in the form $\Phi(X_1, \ldots X_l) \wedge \psi(|X_1|_1, \ldots, |X_l|_m)$, where $|X_i|_j$ is to be interpreted as $\Sigma_{x \in X_i} f_j(x)$, $\Phi$ is a monadic second-order formula, and $\psi$ is an evaluation relation. An evaluation relation is a propositional formula whose atoms are sign conditions on polynomials built from the $|X_i|_j$. We can also replace the relation $\psi$ by an objective function and ask for its extremal value: $\max_{\Phi(X_1, \ldots, X_l)} F(|X_1|_1, \ldots, |X_l|_m)$. Problems in these classes will be called EMS problems.

M. O. Rabin in [29] introduced a notion of model interpretability of theories as a useful tool to prove decidability or undecidability of given theories. An extended variant of interpretability was used by Compton and Henson [17] to investigate the complexity of logical theories. We will introduce a related notion of interpretability to transform effectively a given algorithmic problem $P_1$ on a given class $K_1$ to a different problem $P_2$ on another class $K_2$. Using this notion we will transform a problem on labeled graphs formulated as a MS property to another problem on labeled binary trees. Since we can transform our labeled graph (of bounded treewidth) to a labeled binary tree in linear time, we can decide properties on such graphs in linear time if we can decide MS properties on labeled binary trees in linear time. We give a direct proof of this by showing how to construct, from a given MS formula, a tree automaton (which is a generalized finite automaton accepting labeled binary trees, see Section 5) that decides whether the MS formula is satisfied by a structure representing a labeled binary tree (given as input to the tree automaton).

Problems involving evaluations are solved by the following procedure: a tree automaton deciding an MS property is equipped with maps or counters which are updated according to simple rules, counting extrema or weights of certain subclasses of problems related to the given problem. This technique has been used for hand translation of particular problems (see e.g., Arnborg and Proskurowski [6]), but we will show that the approach is quite general.

The main contributions of this paper are

(i) a construction by which, for a given bounded treewidth, a general MS graph property $P$ is transformed to an MS binary tree property $\tau(P)$, and a general labeled graph $G$ with a suitable tree-decomposition is transformed to a labeled binary tree $T(G)$ in time linear in the number of vertices of $G$ and in such a way that $P$ holds for $G$ if and only if $\tau(P)$ holds for $T(G)$. This allows us, using techniques developed by Doner [20] and Thatcher and Wright [42], to compile a tree automaton which decides the MS-problem $\tau(P)$ on the tree $T(G)$ (and thus also $P$ on the graph $G$) in linear time, and

(ii) a procedure whereby such an automaton for a MS formula with free variables is modified to solve a related EMS problem involving counting.

Our algorithms are explicit and have constructive performance bounds. Moreover, the corresponding algorithms can be obtained automatically if the description of the MS or EMS problem is given in the corresponding monadic second-order language or its extension for EMS problems, respectively. However, they are practically feasible only for moderate values of treewidth and sizes of the involved MS formulae (particularly, the nesting level of quantifiers). A related effort was made by Borie, Tovey, and Parker [15, 16] to define problems easy for recursively defined families of $k$-terminal graphs. They derive many results similar to ours using a direct manipulation of parses for graphs and tree automata.

## 2. Overview of Paper

In Section 3 we fix our technical definitions of the graph structure we consider, mixed graphs. We define treewidth and tree decomposition of this structure. The monadic second-order language we consider is defined and used to define MS and EMS properties. Section 4 defines our interpretability concept and proves that the structures we consider can be linear time interpreted into the class of labeled binary trees. In Section 5 we show how to construct a tree automaton that decides whether a labeled binary tree is a model for a fixed MS formula. We also consider problems

with counting and show classes of problems decidable by counting automata in linear or polynomial time. Section 6 illustrates our approach on reliability computations, a very significant engineering application. Extensions and conclusions are described in Section 7.

## 3. Definitions

We allow a graph to be directed, undirected, or mixed (i.e., having directed as well as undirected edges), and it may contain some prespecified (fixed) sets of distinguished vertices or edges. Thus, generally a graph can be regarded as a relational structure $(A, V, E, D, P_1, \ldots, P_p, R_1, R_2, R_3)$, where $A$ is a nonempty set (set of vertices and edges), $V, E, D, P_1, \ldots, P_p$ are unary predicates, and $R_1, R_2, R_3$ are binary predicates, with the intended meaning that:

$V$ designates the set of vertices,

$E$ designates the set of undirected edges,

$D$ designates the set of directed edges,

$P_1, \ldots, P_p$ are special sets of vertices and edges,

$R_1(a, b)$ holds if and only if $a$ is a vertex incident with the edge $b$

$R_2(a, b)$ holds if and only if $a$ is the tail of the directed edge $b$

$R_3(a, b)$ holds if and only if $a$ is the head of the directed edge $b$.

We shall also allow evaluations of the edges and vertices by non-negative integer or rational numbers. In this case one has to relate the corresponding functions to the structures. When dealing with algorithmic problems of graphs it is natural to connect with a graph $G$ a measure, its size $|G|$. For a graph without evaluations this is the number of vertices and edges. In cases with integer-valued evaluations we will also add all values in these to obtain the size of the problem instance, and for rational-valued evaluations we add the numerators and denominators, when each evaluation is expressed with a common denominator over the domain. For a graph $G$ we denote its number of vertices by $n$. The following definition is due to Robertson and Seymour [31] and is adapted here to the notion of mixed graphs:

Definition 3.1. A *tree decomposition* of a mixed graph $G = (A, V, E, D, P_1, \ldots, P_p, R_1, R_2, R_3)$ is a pair $(T, S)$, where $T$ is a tree and $S$ is a family of sets indexed by the vertices of $T$, such that:

1. $\bigcup_{X_t \in S} X_t = A$.

2. For all $c$ such that $E(c)$ there exists a unique $X_t \in S$ such that $c \in X_t$; and if $a$ satisfies $R_1(a, c)$ then $a \in X_t$.

3. For all $c$ such that $D(c)$ there exists a unique $X_t \in S$ such that $c \in X_t$; and if $a, b$ satisfies $R_2(a, c)$ and $R_3(b, c)$, respectively, then $a, b \in X_t$.

4. For all $a \in A$ the subgraph of $T$ induced by $\{t | a \in X_t\}$ is connected.

The *width* of such a decomposition is $\max_{X_t \in S} |\{a | a \in X_t, V(a)\}| - 1$.

A graph $G$ is of treewidth $w$ if and only if the smallest width of a tree decomposition of $G$ is $w$. We shall need the following:

LEMMA 3.2 (Arnborg, Corneil, and Proskurowski [4]). *For every natural number $w$, a tree decomposition of width $w$ for a graph $G$ can be found in polynomial time, if one exists.*

What is actually proved in [4] is that membership in the class of partial $k$-trees can be decided in time $O(n^{k+2})$ and that a partial $k$-tree can be embedded in a $k$-tree in time $O(n^{k+2})$. But from such an embedding of the underlying simple graph of $G$ it is easy to produce a tree decomposition [37]. This tree decomposition can be found in such a way that the size of the tree is $\leq n - k$.

*Remark.* We should also mention that an $O(n^2)$ approximate embedding algorithm was developed by Robertson and Seymour [32, 33]. Various improvements are possible, see, e.g., Courcelle [20] and Bodlaender [14].

The *monadic second-order language* for a given class of structures (i.e., for a fixed signature) results from the usual first-order language by adding variables for sets of individuals and the membership symbol $\in$ and allowing quantification also over set variables. To complete the definition of monadic second-order logic one has to define the satisfaction relation $G \models \Phi$ for a given structure $G$ of the corresponding signature with domain $A$ and a given formula $\Phi$ whose free individual variables have a given interpretation as elements of $A$ and whose free set variables have a given interpretation as subsets of $A$. This is done as usual by induction on the structure of $\Phi$ where the only new case $\exists X \Phi(X)$ with set variable $X$ is

$G \models \exists X \Phi(X)$ if and only if there is a subset $B$ of $A$ such that

$G \models \Phi[B]$, where $B$ is chosen as interpretation of $X$.

Universal quantification and the usual set operators and relations can now be defined in the standard way.

DEFINITION 3.3. A property $P$ is a monadic second order property (or MS property) over a class $K$ of structures of a given signature if there is a

monadic second-order formula $\Phi$ such that for all $G$ of $K$: $P$ holds for $G$ if and only if $G \models \Phi$.

In the following we shall write $P(G)$ instead of "$P$ holds for $G$." Many properties of graphs are monadic second-order properties for the class of all graphs. But some properties are not, since they ask about the cardinality of a set of edges or vertices with a specific property or about the maximum or minimum of a sum of an evaluation (weight function) over such a set. To handle these kinds of problems also, extended monadic second-order problems (briefly denoted as EMS problems) are introduced.

We will combine an open MS formula containing free set variables $X_1, \ldots, X_l$ with a formula involving the sums of the evaluations $f_1, \ldots, f_m$ over sets for which $X_1, \ldots, X_l$ stand. An *evaluation term* is an expression built from the arithmetic operators $+$, $-$, $\times$, and with atoms being rational numbers, $|X_i|_j$ for $1 \le i \le l$ and $1 \le j \le m$, and $Y_i$ for $1 \le i \le t$. We denote it by, e.g., $F(|X_1|_1, \ldots |X_l|_m, Y_1, \ldots Y_t)$. In case that the evaluations are all in the set of natural numbers, we can also allow mod $p$ for a natural number $p$ in evaluation terms. Most EMS problems have only one evaluation which is identically 1, thus $|X_1|_1$ represents the cardinality of $X_1$.

An *evaluation relation* is a propositional formula with atoms being of the form $F \le 0$ or $F = 0$, where $F$ is an evaluation term.

The intended interpretation of evaluation terms and relations should be obvious. Using them we can define the following problems, collectively named EMS problems:

DEFINITION 3.4. A property $P$ is an extended monadic second-order property, or EMS property over a class $K$ of structures $(G, f_1^G, \ldots, f_m^G)$ if there are constants $l$ and $t$, a monadic second-order formula $\Phi$ with free set variables $X_1, \ldots, X_l$ and an evaluation relation $\psi(Y_1, \ldots, Y_{lm+t})$ such that for each structure $(G, f_1^G, \ldots, f_m^G) \in K$ it holds:

$P(G, f_1^G, \ldots, f_m^G, C_1, \ldots, C_t)$ if and only if there are subsets $A_1, \ldots, A_l$ of the domain $A$ such that $G \models \Phi[A_1, \ldots, A_l]$ and

$$\psi\left[\sum_{a \in A_1} f_1^G(a), \ldots, \sum_{a \in A_l} f_m^G(a), C_1, \ldots, C_t\right].$$

Above, the $C_i$ correspond to numbers given with the problem instance. A problem is an *extended monadic second-order extremum problem* if it can be stated as finding the maximum of an evaluation term $F(\Sigma_{a \in A_1} f_1^G(a), \ldots, \Sigma_{a \in A_l} f_m^G(a), C_1, \ldots, C_t)$ under the constraints $\Phi[A_1, \ldots, A_l]$ and $\psi[\Sigma_{a \in A_1} f_1^G(a), \ldots, \Sigma_{a \in A_l} f_m^G(a), C_1, \ldots, C_t]$ for MS

formula $\Phi$ and evaluation relation $\psi$. Since negation is allowed in evaluation terms, we can express minimization in terms of maximization.

For a *linear EMS extremum problem* we require that the associated evaluation term $F$ is linear in the quantities $|X_i|_j$, and that the evaluation relation is missing (i.e., identically true).

The *enumeration problem* for an MS property $\Phi(X_1, \ldots, X_l)$ is that of computing

$$|\{(A_1, \ldots, A_l): G \models \Phi[A_1, \ldots, A_l]\}|.$$

An *EMS packing (partitioning) problem* for an MS property $\Phi$ is the problem of packing (partitioning the vertices of) a given graph $G$ with a maximum (into a minimum or maximum) number of connected subgraphs with property $\Phi$. All EMS packing and partitioning problems can be expressed as linear EMS extremum problems as follows. Let $X$ be the union of edges in the subgraphs asked for. We require that each maximal connected component of the graph induced by $X$ has property $\Phi$ (and, in the case of partitioning, that every vertex of $G$ be incident to some edge of $X$). $Y$ is a set of representatives of the maximal connected components of the induced graph. These conditions on $X$ and $Y$ are easy to express with an MS formula $\Phi'(X, Y)$ and we seek the extremum of $|Y|$ constrained by $\Phi'$. Since we require disjointness we cannot easily express covering problems, and since we require connectedness we cannot phrase the chromatic index problem in this way. This appears to be no coincidence, since only those packing and partitioning problems expressible in this way have been shown to be linear time solvable.

The concept of EMS problems is very powerful since many of the NP-hard problems for graphs are EMS problems. Some are gathered in the following theorems where we use the terminology of Garey and Johnson [22].

The proofs of the theorems are almost trivial. In most cases one has only to perform a straightforward transformation of the definition of the problem into an extended second-order expression. We will give one example from each theorem.

THEOREM 3.5. *The following properties are MS properties: domatic number for fixed K* [*GT*3], *graph K-colorability for fixed K* [*GT*4], *achromatic number for fixed K* [*GT*5], *monochromatic triangle* [*GT*6], *partition into triangles* [*GT*11], *partition into isomorphic subgraphs for fixed connected H* [*GT*12], *partition into Hamiltonian subgraphs* [*GT*13], *partition into forests for fixed K* [*GT*14], *partition into cliques for fixed K* [*GT*15], *partition into perfect matchings for fixed K* [*GT*16], *covering by cliques for fixed K* [*GT*17], *covering by complete bipartite subgraphs for fixed K* [*GT*18], *induced subgraph with property P (for monadic second-order*

*properties P and fixed K)* [*GT*21], *induced connected subgraph with property P (for monadic second-order properties P and fixed K)* [*GT*22], *induced path for fixed K* [*GT*23], *cubic subgraph* [*GT*32], *Hamiltonian completion for fixed K* [*GT*34], *Hamiltonian circuit* [*GT*37], *directed Hamiltonian circuit* [*GT*38], *Hamiltonian path (and directed Hamiltonian path)* [*GT*39], *subgraph isomorphism for fixed H* [*GT*48], *graph contractability for fixed H* [*GT*51], *graph homomorphism for fixed H* [*GT*52], *path with forbidden pairs for fixed n* [*GT*54], *kernel* [*GT*57], *degree constrained spanning tree for fixed K* [*ND*1], *disjoint connecting paths for fixed K* [*ND*40], *chordal graph completion for fixed K, chromatic index for fixed K, spanning tree parity problem, distance d chromatic number for fixed d and k, thickness* $\leq K$ *for fixed K, membership for each class C of graphs which is closed under minor taking (all of these last problems are unnumbered).*

EXAMPLE. Consider the last case of Theorem 3.5. All minors of a given graph are obtained by first taking a subgraph and then *contracting* a set of edges. The contraction operation for an edge replaces it and its two end-vertices with a new vertex, and this vertex will have a neighborhood which is the union of the neighborhoods of the two original vertices (thus, the homoeomorph familiar from Kuratowskis theorem is a minor constrained so that only edges with one endvertex of degree two can be contracted). According to Wagners Conjecture (or Robertson–Seymours theorem [31]), the class $C$ is exactly characterizable by a finite list of forbidden minors. Call these minors $M_1, \ldots, M_q$. For a given graph $H$, let $P_H$ be the graph property "The graph has $H$ as a minor." $P_H$ is easily expressed as an MS-property using the predicates *Adjacent* and *Connected* defined in the introduction. Introduce set variables $X_1, \ldots, X_r$ for the $r$ vertices of $H$ and let $(f_i, t_i)_{i=1}^{s}$ be the list of pairs of adjacent vertices in $H$ (each pair arbitrarily ordered). $P_H$ is now defined by the formula

$$\exists X_1 \ldots \exists X_r$$
$$\mathit{Connected}(X_1) \wedge \ldots \wedge \mathit{Connected}(X_r)$$
$$\wedge \mathit{Adjacent}(X_{f_1}, X_{t_1}) \wedge \ldots \wedge \mathit{Adjacent}(X_{f_s}, X_{t_s})$$
$$\wedge \bigwedge_{i=1}^{r} \bigwedge_{j=i+1}^{r} \neg \exists x\, x \in X_i \wedge x \in X_j,$$

and likewise the property that the graph has none of $M_1, \ldots, M_q$ as minors can be expressed in MS-logic as

$$\neg(P_{M_1} \vee \ldots \vee P_{M_q}).$$

THEOREM 3.6. *The following problems are linear EMS extremum problems such that* $|G|$ *is the number of vertices and edges in* $G$: *vertex cover* [$GT1$], *dominating set* [$GT2$], *feedback vertex set* [$GT7$], *feedback arc set* [$GT8$], *partial feedback edge set for fixed maximum cycle length* $l$ [$GT9$], *minimum maximal matching* [$GT10$], *partition into cliques* [$GT15$], *clique* [$GT19$], *independent set* [$GT20$], *induced subgraph with property* $\Pi$ (*for MS property* $\Pi$) [$GT21$], *induced connected subgraph with property* $\Pi$ (*for MS property* $\Pi$) [$GT22$], *induced path* [$GT23$], *balanced complete bipartite subgraph* [$GT24$], *bipartite subgraph* [$GT25$], *degree-bounded connected subgraph for fixed* $d$ [$GT26$], *planar subgraph* [$GT27$], *transitive subgraph* [$GT29$], *uniconnected subgraph* [$GT30$], *minimum K-connected subgraph for fixed* $K$ [$GT31$], *minimum equivalent digraph* [$GT33$], *Hamiltonian completion* [$GT34$], *multiple choice matching for fixed* $J$ [$GT55$], *k-closure* [$GT58$], *path distinguishers* [$GT60$], *maximum leaf spanning tree* [$ND2$], *maximum length-bounded disjoint paths for fixed* $J$ [$ND41$], *maximum fixed-length disjoint paths for fixed* $J$ [$ND42$], *minimum edge*(*vertex*) *deletion for any MS property* $P$ [40].

As an example, the property that $X_1$ is a dominating set of a graph is expressed by

$$\mathrm{Dom}(X_1) \equiv \forall v, \quad V(v) \rightarrow \exists w, \quad w \in X_1 \wedge \mathrm{Adj}(v, w).$$

With a weight function $f_1$ that is identically 1 on the vertex set, the dominating set problem [GT2] of [22] is directly solved from the solution to the EMS minimisation problem of $|X_1|_1$ subject to $\mathrm{Dom}(X_1)$.

THEOREM 3.7. *These problems are linear EMS extremum problems*: *bounded diameter spanning tree for fixed* $D$ [$ND4$], *multiple choice branching for fixed* $m$ [$ND11$], *Steiner tree in graphs* [$ND12$], *maximum cut* [$ND16$], *longest circuit* [$ND28$], *longest path* [$ND29$].

The difference between the problems in Theorem 3.6 and those in Theorem 3.7 is that the latter contain an integer valued weight function, whereas the former contain only weight functions with a finite set of values (typically only the value 1, so that the cardinality of a set is measured). As an example, the property that $X_v$ and $X_e$ are the vertices and edges of a circuit, $\mathrm{Circuit}(X_v, X_e)$, can be expressed informally as "$X_v$ consists exactly of the end vertices of edges in $X_e$; the subgraph of $G$ with vertices $X_v$ and edges $X_e$ is connected; every vertex in $X_v$ is incident to exactly two edges in $X_e$".

So, [ND28] of [22] is directly solvable from the solution of the maximization problem of $\Sigma_{x \in X_e} w(x)$ subject to $\mathrm{Circuit}(X_v, X_e)$.

THEOREM 3.8. *The following problems are EMS problems: partition into isomorphic subgraphs for fixed H* [*GT*12], *partition into perfect matching* [*GT*16], *Kth best spanning tree for fixed K* [*ND*9], *bounded component spanning forest for fixed K* [*ND*10], *minimum cut into bounded sets* [*ND*17], *shortest weight-constrained path* [*ND*30], *Kth shortest path for fixed K* [*ND*31].

EXAMPLE. Problem [ND30] of [22] can be defined with the formula $\mathrm{Path}(f, t, X_e, X_v) \equiv$ "$X_e$ is the edge set of a path from $f$ to $t$ over vertices $X_v$," similarly defined as the predicate *Circuit* above. We use two weight sets given in the instance, the weights $w$ and the lengths $l$ of edges. The evaluation relation will be

$$\left(\sum_{x \in X_e} w(x) \leq W\right) \wedge \left(\sum_{x \in X_e} l(x) \leq K\right),$$

where $W$ and $K$ are integers given with the instance.

*Remark*. In all the above results (Theorems 3.5 to 3.8) the properties considered are MS, linear EMS or EMS over the class of all graphs. In case the considered properties are restricted to special classes of graphs, then it might not be necessary to fix all of the constants that we have fixed above. For instance domatic number [GT3], chromatic number [GT4] and graph Grundy numbering [GT56] are EMS problems over the class of graphs of treewidth at most $m$, for constant $m$.

## 4. INTERPRETABILITY FOR ALGORITHMIC PROBLEMS

In this section we introduce a notion of interpretability to transform in an effective way a given algorithmic problem $P_1$ on a given class $K_1$ to a different problem $P_2$ on another class $K_2$ (hopefully simpler or more well known).

The method of interpretability is a widely used method in mathematical logic to transform one theory into another one. Especially the method of syntactic interpretation was used by Tarski, Mostowski, and Robinson [41] to deduce decidability or undecidability of theories from other theories. Semantic interpretations were used by Rabin [29] as a variant of the interpretability concept and proved to be a useful tool to show decidability or undecidability of theories (see also [30]). A different variant was introduced by Compton and Henson [17] to prove lower bounds on the complexity of logical theories. The notion of interpretability used here can be viewed as an extension of Rabin's semantic interpretations.

Let $K_1$ and $K_2$ be two classes of relational structures and let $L_1$ and $L_2$ be corresponding monadic second-order languages. Assume that there are $L_2$-formulas $\varepsilon(x_1, x_2)$, $\alpha(x)$, and $\gamma_{R_i}(x_1, \ldots, x_{l_i})$ (for each relational symbol $R_i$ of $L_1$ of arity $l_i$, where $R_1, \ldots, R_k$ are assumed to be all relational symbols of $L_1$), which have only the indicated variables as free variables. For each structure $G$ of $K_2$ and each $L_2$ formula $\delta(x_1, \ldots, x_s)$ define $G(\delta)$ to be the $s$-ary relation which is defined by $\delta$ in $G$, i.e., $\{(a_1, \ldots, a_s) | a_i$ is in the domain of $G$ for all $i$: $1 \leq i \leq s$ and $G \models \delta[a_1, \ldots, a_s]\}$.

We also require that $G(\varepsilon)$ is an equivalence relation for every $G \in K_2$. For simplicity denote the sequence $(\alpha, \gamma_{R_1}, \ldots, \gamma_{R_k}, \varepsilon)$ of $L_2$ formulas as interpretation $I$ and define $G^I$ for each structure $G \in K_2$, by

$$G^I := \big(G(\alpha), G(\gamma_{R_1}), \ldots, G(\gamma_{R_k})\big)/G(\varepsilon),$$

where $/G(\varepsilon)$ denotes the factorisation by the equivalence relation $G(\varepsilon)$. $G^I$ is obviously a structure for $L_1$, but not necessarily a structure in $K_1$. A simple example of interpretation is worked out in the Appendix.

DEFINITION 4.1. $K_1$ is said to be linear (polynomial) time interpretable into $K_2$ with respect to $L_1$, $L_2$ if there is an $I$ as above and an algorithm $\mathscr{A}$ which constructs for each $G \in K_1$ a structure $\mathscr{A}(G) \in K_2$ in time linear (polynomial) in $|G|$ such that:

$$G \cong (\mathscr{A}(G))^I$$

We need one more definition to cover the case where there also are evaluations $f_1^G, \ldots, f_m^G$ for each structure $G \in K_1$

DEFINITION 4.2. $K_1$ is said to be evaluation preserving linear (polynomial) time interpretable into $K_2$ with respect to $L_1$, $L_2$ if there is an $I$ as above and an algorithm $\mathscr{A}$ which constructs for each $G \in K_1$ a structure $\mathscr{A}(G) \in K_2$ in time linear (polynomial) in $|G|$ such that there exists an isomorphism $h$: $(\mathscr{A}(G))^I \to G$ satisfying

$$f_i^G(h([a]G(\varepsilon))) = \sum_{a' \in [a]G(\varepsilon)} f_i^{\mathscr{A}(G)}(a')$$

for every $a \in G(\alpha)$ (here $[a]G(\varepsilon)$ denotes the equivalence class of $G(\varepsilon)$ generated by $a$).

Each interpretation gives a canonical way to transform EMS problems for $K_1$ into EMS problems for $K_2$. To define this transformation we define first a transformation from $L_1$ formulas into $L_2$ formulas. This can

be done by induction on the structure of the formulas in the following way:

$$(x \in X)^I := \exists y(\varepsilon(y, x) \wedge y \in X)$$

$$(x_1 = x_2)^I := \varepsilon(x_1, x_2)$$

$$R(x_1, \ldots, x_{l_R})^I := \exists y_1, \ldots, y_{l_R}\Big(\bigwedge_{1 \le i \le l_R} \varepsilon(x_i, y_i) \wedge \gamma_R(y_1, \ldots, y_{l_R})\Big)$$

for each relational symbol $R$ from $L_1$

$$(\neg \varphi)^I := \neg(\varphi^I)$$

$$(\varphi \wedge \psi)^I := \varphi^I \wedge \psi^I$$

$$(\exists x \varphi)^I := \exists X(\varphi^I \wedge \alpha(x))$$

$$(\exists X \varphi)^I := \exists X(\varphi^I).$$

Evaluation terms and relations do not have to be changed. Denote the following transformation of an EMS-problem $P$ by $P^I$. If $P$ is defined using MS formula $\Phi(X_1, \ldots, X_l)$, then $P^I$ will be the EMS problem which is $P$ with the only difference that it uses instead the MS formula

$$\Phi^I(X_1, \ldots, X_l) \wedge \bigwedge_{i=1}^{l} \forall z_1, z_2(z_1 \in X_i \wedge \varepsilon(z_1, z_2)) \to z_2 \in X_i.$$

The added conjunction will ensure one-to-one correspondence of solutions and weight sets of the two problems over the two domains. The following result will be a helpful tool to prove linear (polynomial) complexity for EMS problems.

THEOREM 4.3. *Assume that $P$ is an EMS-problem on a class $K_1$ and let $I$ be an evaluation preserving linear (polynomial) time interpretation of $K_1$ into a class $K_2$. If $P^I$ can be solved over $K_2$ in linear (polynomial) time, then $P$ can also be solved in linear (polynomial) time over $K_1$.*

*Proof.* To prove this theorem one has to show that $P(G)$ is equivalent to $P^I(\mathscr{A}(G))$ for all $G \in K_1$. But this follows easily from the definitions, by induction over the structure of the MS formulas. Specifically, we must show first (i) and then (ii) below for an arbitrary $G_2 \in K_2$:

(i) For all $b_i \in G_2(\alpha)$ and all $B_j \subset G_2(\alpha)$ such that $(\ldots, b_i, \ldots, B_j, \ldots) \in G_2(\varphi^I)$, and all $b_i'$ such that $\varepsilon(b_i, b_i')$ and $B_j'$ such that, for all $b_j \in B_j$ there is a $b_j' \in B_j'$ with $\varepsilon(b_j, b_j')$ and for all $b_j' \in B_j'$ there is a $b_j \in B_j$ with $\varepsilon(b_j, b_j')$, it holds that $(\ldots, b_i', \ldots, B_j', \ldots) \in G_2(\varphi^I)$

(ii) for all $(\ldots, b_i, \ldots, B_j, \ldots) \in G_2(\varphi^I)$ we also have $(\ldots, [b_i], \ldots, [B_j], \ldots) \in G_2^I(\varphi)$, where [ ] refers to equivalence classes generated by $G_2(\varepsilon)$.

From (i) and (ii) follows that $G_2 \models \varphi^I$ if and only if $G_2^I \models \varphi$. □

The following technical lemma is useful in the applications of the theorem.

LEMMA 4.4. *Evaluation preserving linear (polynomial) time interpretability is transitive.*

*Proof.* The proof of this lemma is straightforward; if we have the pairs of structure class and language $(K_1, L_1)$, $(K_2, L_2)$, and $(K_3, L_3)$, and a linear (polynomial) time interpretation is given by $(I_{12}, \mathscr{A}_{12})$ from $(K_1, L_1)$ to $(K_2, L_2)$ and by $(I_{23}, \mathscr{A}_{23})$ from $(K_2, L_2)$ to $(K_3, L_3)$, then the pair of composed functions $(I_{12} \circ I_{23}, \mathscr{A}_{12} \circ \mathscr{A}_{23})$ yields a linear (polynomial) time interpretation from $(K_1, L_1)$ to $(K_3, L_3)$. □

*Remark.* The restriction of $K_2$ to be a class of relational structures is not necessary. The interpretability notion can easily be extended to arbitrary classes of structures. Moreover, other languages than monadic second-order logic can be considered.

We will now show how MS and EMS problems for our type of graphs (vertex and edge labeled mixed graphs) of bounded treewidth are interpreted into the class of EMS problems for labeled binary trees, where binary trees are represented by relational structures where the nodes of the trees are objects in the universe of the structure and the tree structure is described by the left and right son relations $L$ and $R$. First a tree decomposition of $G$ is used to produce a labeled tree. Then the tree is transformed to a labeled binary tree.

THEOREM 4.5. *For each natural number $m$ every class $K_1$ of graphs whose treewidth is bounded by $m$ is evaluation preserving polynomial time interpretable into the class of binary trees. If the graphs from $K_1$ are given together with a tree decomposition then the corresponding transformations can be performed in linear time.*

*Proof.* As a first step in the proof of this theorem we shall prove that there is an evaluation preserving polynomial interpretation of $K_1$ into the class of trees with a bounded number of designated unary predicates. Assume that $G = (A, V, E, D, P_1, \ldots, P_p, R_1, R_2, R_3) \in K_1$ is given. At first one constructs a tree decomposition $(T, S)$, where $S = \{X_t | t$ is a vertex in $T\}$, using Lemma 3.2. Then one takes the tree $T$ and constructs from it a new tree $T'$ with some designated unary predicates as follows. For each vertex $t$ of $T$ and each $a \in A$ such that $a \in X_t$ perform the following operation: add a new vertex, $a_t$, to $T'$ and connect $t$ and $a_t$ with a new edge in $T'$.

Now define predicates $P_V$, $P_E$, and $P_D$ representing the following sets:

$$P_V := \{a_t | t \text{ is a vertex in } T,\ a \in X_t \text{ and } V(a)\}$$

$$P_E := \{a_t | t \text{ is a vertex in } T,\ a \in X_t \text{ and } E(a)\}$$

$$P_D := \{a_t | t \text{ is a vertex in } T,\ a \in X_t \text{ and } D(a)\}.$$

These predicates enable us to recognize the vertices, edges, and directed edges of $G$ in $T'$. Now enumerate for each $t$ the unordered pairs $\{a, b\}$ of vertices in $X_t$. There are at most $\binom{m+1}{2}$ pairs for each $t$ since the treewidth of $G$ is bounded from $m$. For each $i$: $1 \le i \le \binom{m+1}{2}$ define a predicate $P^i_{R_1}$ as follows:

> $P^i_{R_1}$ represents the set of all $a_t$, $b_t$, and $e_t$ such that $a, b, e \in X_t$ and $\{a, b\}$ is an unordered pair of vertices of $G$ with number $i$ at node $t$ and $e$ is an undirected edge of $G$ incident with $a$ and $b$.

Similarly, one enumerates all ordered pairs $(a, b)$ of vertices in $X_t$ (for each $t$) and defines a predicate $P^i_{R_2}$ (for $i$: $1 \le i \le (m+1)^2$) as follows:

> $P^i_{R_2}$ represents the set of all $a_t$, $d_t$ such that there is a vertex $b$ of $G$ such that $(a, b)$ has number $i$ at node $t$ and $d$ is a directed edge of $G$ with head $b$ and tail $a$.

Now define for each $i$: $1 \le i \le (m+1)^2$ a predicate $P^i_{R_3}$ as follows:

> $P^i_{R_3}$ represents the set of all $b_t$, $d_t$ such that there is a vertex $a$ of $G$ such that $(a, b)$ has number $i$ at node $t$ and $d$ is a directed edge of $G$ with head $b$ and tail $a$.

A remaining problem is that there are different vertices in $T'$ which represent one and the same vertex of $G$. To solve this problem, observe that it is possible to color the vertices of $G$ with $2m + 2$ colors such that for all adjacent vertices $t, t'$ of $T$ and each pair of vertices $a, b$ from $G$ with $a \neq b$, $a$ and $b$ are colored with different colors if they are both contained in $X_t \cup X_{t'}$. This follows easily from the definition of a tree decomposition and the fact that the width of $G$ is bounded by $m$. Moreover, such a coloring can be constructed in linear time if the tree decomposition is given. Hence assume that the vertices of $G$ are colored with colors $C_1, \ldots, C_{2m+2}$ in this way. For each $i$: $1 \le i \le 2m + 2$ define a new predicate $P_{C_i}$ so that it represents the set of all vertices $a_t \in T'$ such that $a$ is colored in $G$ with $C_i$. Moreover, define $P'_i$ (for each $i$: $1 \le i \le p$) to be the unary predicate which represents the set of all $a_t$ for which $P_i(a)$ holds in $G$. Let $f^G_1, \ldots, f^G_m$ be the evaluations for $G$ and let $g$ be a function from $A$ to the vertex set of $T$ such that $a \in X_{g(a)}$ for each $a \in A$. It is obvious that such a function exists and can be constructed in linear time from a given tree decomposition. For each $G$, $(T, S)$ as above

we define $\mathscr{A}(G) := T'$. It remains to define the new evaluations $f_i^{\mathscr{A}(G)}$. This is done by

$$f_i^{\mathscr{A}(G)}(v) := \begin{cases} f_i^G(a) & \text{if } v = a_{g(a)} \\ 0 & \text{otherwise} \end{cases}$$

for all $i$: $1 \leq i \leq m$. Now the formulas of the interpretation can be chosen as follows. First define

$$\varepsilon_1(x, y) := \bigvee_{1 \leq i \leq 2m+2} \big(P_{C_i}(x) \wedge P_{C_i}(y) \wedge \text{“there exists an } x, y \text{ path whose internal vertices all have a neighbour satisfying } P_{C_i}\text{”}\big).$$

The parts of this formula enclosed in “ ” can easily be formalized in the corresponding monadic second-order language. The definition of $\varepsilon(x, y)$ is now easy:

$$\varepsilon(x, y) := x = y \vee \big(P_V(x) \wedge P_V(y) \wedge \varepsilon_1(x, y)\big).$$

Choose as $\alpha(x)$ a formula in a language for $T'$ expressing $x$ has degree 1; then define:

$$\gamma_V(x) := P_V(x)$$

$$\gamma_E(x) := P_E(x)$$

$$\gamma_D(x) := P_D(x)$$

$$\gamma_{R_1}(x, y) := P_V(x) \wedge P_E(y) \wedge \bigvee_{1 \leq i \leq \binom{m+1}{2}} P^i_{R_1}(x) \wedge P^i_{R_1}(y) \wedge \text{“there is a vertex } u \text{ which is adjacent to } x \text{ and } y\text{”}$$

$$\gamma_{R_2}(x, y) := P_V(x) \wedge P_D(y) \wedge \bigvee_{1 \leq i \leq (m+1)^2} P^i_{R_2}(x) \wedge P^i_{R_2}(y) \wedge \text{“there is a vertex } u \text{ which is adjacent to } x \text{ and } y\text{”}$$

$$\gamma_{R_3}(x, y) := P_V(x) \wedge P_D(y) \wedge \bigvee_{1 \leq i \leq (m+1)^2} P^i_{R_3}(x) \wedge P^i_{R_3}(y) \wedge \text{“there is a vertex } u \text{ which is adjacent to } x \text{ and } y\text{”}$$

$$\gamma_{P_i}(x) := P'_i(x) \qquad \text{for all } i\text{: } 1 \leq i \leq s.$$

For the proof that this indeed gives an evaluation preserving polynomial time interpretation one has only to keep in mind all the above definitions.

Note that the tree $T'$ has no vertex of degree 2 (unless the tree decomposition has redundant nodes in the tree $T$, which is easy to avoid).

To end the proof of Theorem 4.5 via transitivity we have to go from the class of labeled trees without vertices of degree 2 to the class of labeled binary trees.

Let $T'$ be a tree without vertices of degree 2. From this tree build a new tree $T''$ which has a vertex $t_0$ of degree 2 and, besides $t_0$, only has vertices of degree 3 and 1. This can be done as follows. Introduce a new predicate $P_c$ such that adjacent vertices will have different values for $P_c$, which is possible since trees are two-colorable. Recursively split the vertices of degree $d \geq 4$ into two adjacent vertices, one with degree $d - 1$ and one with degree 3, both having the same value for $P_c$ as the old vertex. Finally, when all the vertices have degree 3 or 1, one vertex which was of degree 1 in $T'$ is split into one vertex of degree 1 and one, $t_0$, of degree 2.

Using the ideas of this construction it is not difficult to find a corresponding interpretation. When a vertex is split, one of the new vertices obtains the evaluations and unary predicates of the old vertex and the other obtains the evaluation 0 and false predicates.

An evaluation preserving linear time interpretation of the class of labeled trees with one vertex of degree 2 and the rest of the vertices of degree 1 or 3 into the class of labeled binary trees is easily found. Simply take the vertex of degree 2 as the root and choose an arbitrary ordering. Then the old unoriented tree can be recognized, since two vertices in the old tree are adjacent if and only if one of the corresponding nodes in the binary tree is a left or a right son of the other. □

An example of the preceding construction is given in the appendix.

## 5. Obtaining the Algorithm

Most of the techniques we need to decide MS properties and solve EMS problems were developed during the late sixties in studies of decision problems in logic, particularly the weak monadic second-order theory of two successors, which is very similar to the monadic second-order theory of binary trees with left and right sons indicated by predicates $L$ and $R$. The most relevant papers (for this application) are Doner [20] and Thatcher and Wright [42]. Since those applications were somewhat different from ours, we prefer to give a repackaged exposition using contemporary graph- and automata-theoretic terminology. We follow [42] except that we omit their discussion of regular sets.

DEFINITION 5.1. A tree automaton $M = (S, \Sigma, \delta, s_0, A)$ is a quintuple where:

$S$ is a finite set of states

$\Sigma$ is a finite set, the alphabet, disjoint from $S$

$\delta$ is a transition function $S \times S \times \Sigma \to S$

$s_0$ is the initial state, $s_0 \in S$

$A$ is the set of accepting states, $A \subset S$.

A tree automaton $M = (S, \Sigma, \delta, s_0, A)$ *executes* a binary tree whose nodes are labeled with elements from the alphabet $\Sigma$ by assigning states to its nodes. The empty tree has state $s_0$. If the sons of node $v$ were assigned states $s_l$ and $s_r$, respectively, and node $v$ is labeled $a$, $a \in \Sigma$, then $v$ will be assigned state $\delta(s_l, s_r, a)$ (e.g., a leaf with label $a$ will be assigned state $\delta(s_0, s_0, a)$). $M$ *accepts* a tree if it assigns a state in $A$ to its root. Obviously, a tree automaton can execute a tree in time linear in the size of the tree.

In this application we want to recognize binary trees satisfying an MS formula $\Phi$. The first step is to replace every individual variable $x$ in $\Phi$ by a set variable $X$. This can be done by a transformation $I$ of the formula according to the rules

$$
\begin{aligned}
(x \in Y)^I &:= X \subset Y \wedge \mathrm{Sing}(X) \\
(x = y)^I &:= X \subset Y \wedge \mathrm{Sing}(X) \wedge \mathrm{Sing}(Y) \\
L(x, y)^I &:= \tilde{L}(X, Y) \wedge \mathrm{Sing}(X) \wedge \mathrm{Sing}(Y) \\
R(x, y)^I &:= R(X, Y) \wedge \mathrm{Sing}(X) \wedge \mathrm{Sing}(Y) \\
P_i(x)^I &:= \tilde{P}_i(X) \wedge \mathrm{Sing}(X) \\
(\exists x F)^I &:= \exists X F^I \\
(\neg \varphi)^I &:= \neg(\varphi^I) \\
(\varphi \wedge \psi)^I &:= \varphi^I \wedge \psi^I,
\end{aligned}
$$

where the predicate *Sing* has the interpretation that $\mathrm{Sing}(X)$ is true if and only if $X$ is a singleton set, $\tilde{L}(X, Y)$ is true if and only if for every $x \in X$ and every $y \in Y$, $L(x, y)$ is true, $\tilde{R}$ is similar to $\tilde{L}$ and $\tilde{P}_i(X)$ is true if every member of $X$ satisfies $P_i$. We will now encode unary predicates and free variables of MS formulae into the node labels of the binary tree. Thus $\Sigma = \{0, 1\}^{p+m}$, with one bit for every of the $p$ unary predicates and one bit for every of the $m$ free variables in $\Phi$.

$\tilde{P}_i(X)$

| $l$ | $r$ | $P_i$ | $X$ | $\delta$ |
|---|---|---|---|---|
| $s_1$ | – | – | – | $s_1$ |
| – | $s_1$ | – | – | $s_1$ |
| – | – | 0 | 1 | $s_1$ |
| – | – | – | – | $s_0$ |
| $A = \{s_0\}$ | | | | |

$\tilde{L}(X,Y)$

| $l$ | $r$ | $X$ | $Y$ | $\delta$ |
|---|---|---|---|---|
| $s_0$ | $s_0$ | 1 | 0 | $s_2$ |
| $s_2$ | $s_0$ | 0 | 1 | $s_3$ |
| $s_3$ | $s_0$ | 0 | 0 | $s_3$ |
| $s_0$ | $s_3$ | 0 | 0 | $s_3$ |
| $s_0$ | $s_0$ | 0 | 0 | $s_0$ |
| – | – | – | – | $s_1$ |
| $A = \{s_3\}$ | | | | |

*Sing*$(X)$

| $l$ | $r$ | $X$ | $\delta$ |
|---|---|---|---|
| $s_0$ | $s_0$ | 0 | $s_0$ |
| $s_0$ | $s_0$ | 1 | $s_1$ |
| $s_1$ | $s_0$ | 0 | $s_1$ |
| $s_0$ | $s_1$ | 0 | $s_1$ |
| – | – | – | $s_2$ |
| $A = \{s_1\}$ | | | |

$X \subset Y$

| $l$ | $r$ | $X$ | $Y$ | $\delta$ |
|---|---|---|---|---|
| $s_1$ | – | – | – | $s_1$ |
| – | $s_1$ | – | – | $s_1$ |
| – | – | 1 | 0 | $s_1$ |
| – | – | – | – | $s_0$ |
| $A = \{s_0\}$ | | | | |

FIG. 1. Automata recognizing atoms of MS-formulae.

The automaton deciding $\Phi$ is built recursively from automata recognizing subformulae of $\Phi$. The atoms $\tilde{L}(X,Y)$, $\tilde{P}_i(X)$, $X \subset Y$, and Sing$(X)$ are recognized by the automata in Fig. 1, and the automaton for $\tilde{R}(X,Y)$ is a simple modification of that for $\tilde{L}(X,Y)$. The transition functions in Fig. 1 are specified with the sign – for "don't care" conditions, and the value of $\delta$ should be taken from the first row that matches (the dependency of $\delta$ on its third argument is just a bit test—the bit or bits tested are indicated in the column headers). The initial state is $s_0$ for the four automata of Fig. 1. Actually, *Sing* can be defined in terms of $\subset$, but this does not seem to be a great advantage in this case [42].

The logical constants in MS formulae can be reduced to $\neg$, $\wedge$, and $\exists$. If $\varphi$ is recognized by automaton $M = (S, \Sigma, \delta, s_0, A)$, then $\neg\varphi$ is recognized by $M_\neg = (S, \Sigma, \delta, s_0, S - A)$. If $\varphi_1$ and $\varphi_2$ are recognized by $M_1 = (S_1, \Sigma, \delta_1, s_{01}, A_1)$ and $M_2 = (S_2, \Sigma, \delta_2, s_{02}, A_2)$, respectively, then $\varphi_1 \wedge \varphi_2$ is recognized by $M_\wedge = (S_1 \times S_2, \Sigma, \delta_\wedge, (s_{01}, s_{02}), A_1 \times A_2)$, with $\delta_\wedge((s_l', s_l''), (s_r', s_r''), a) = (\delta_1(s_l', s_r', a), \delta_2(s_l'', s_r'', a))$.

The treatment of $\exists$ is analogous to the conversion of an ordinary nondeterministic automaton to an equivalent deterministic one. Let $\varphi(X)$ be recognized by $M = (S, \Sigma, \delta, s_0, A)$. Then $\exists X_\varphi(X)$ is recognized by

$M = (2^S, \Sigma, \delta_\exists, \{s_0\}, A_\exists)$, where

$$\delta_\exists(S_l, S_r, a) = \{\delta(s_l, s_r, b) | b \in \Sigma, s_l \in S_l, s_r \in S_r, \text{ and } b_{[X=0]} = a_{[X=0]}\}.$$

Above, $a_{[X=0]}$ is obtained from $a$ by replacing the bit representing variable $X$ by 0, and

$$A_\exists = \{S' | S' \cap A \neq \emptyset\}.$$

The preceding discussion easily gives:

THEOREM 5.2. *For each MS-property $P$ and each class $K$ of labeled binary trees, $P(G)$ for $G \in K$ can be decided in time linear in $n$.*

Combining this with Theorem 4.5 we have

COROLLARY 5.3 (Courcelle [19, 20]). *For each MS-property $P$, and for each class $K$ of graphs of universally bounded treewidth, the problem $P(G)$ for $G \in K$ can be decided in linear time, if $G$ is given together with a tree decomposition.*

*Remark.* Courcelle actually proved something stronger. He proved that the *counting monadic second-order logic* is decidable in linear time for classes of graphs with bounded treewidth. But the only thing that differs between this logic and the usual monadic second-order logic is that its language contains atomic formulas of the form $\text{Card}_{p,q}(X)$, which has the interpretation that $|X| \equiv q \pmod{p}$. This extension can be covered also using our approach, but we do not need it in our main applications.

### 5.1. *EMS Problems*

EMS problems are frequently described and solved for tree-like graph families, but the solutions are usually obtained using intuition and hard work. Specifically, a key problem is to identify certain equivalence or congruence classes [6, 19] or homomorphisms [8]. We shall see that this information is actually produced by the tree automaton compilation process and that linear time algorithms are possible under fairly general conditions. In order to see how this is done we consider the tree automaton produced by the open formula $\Phi(X_1, \ldots X_l)$, $M = (S, \Sigma \times B, \delta, s_0, A)$ which works on labeled binary trees, with labels taken from the set of $p + l$-bit vectors, $\Sigma = \{0, 1\}^p$ (one bit for each unary predicate $P_1, \ldots, P_p$, acting as a membership flag) and $B = \{0, 1\}^l$ (one bit for each free set variable, acting as a characteristic function for the sets $X_1, \ldots X_l$). The transition function $\delta$ is of type $S \times S \times \Sigma \times B \to S$. In an ordinary MS problem, existential quantifiers are added, $M$ is converted to a nondeterministic automaton which guesses the labels (i.e., the memberships of

the current node in the $X_i$) and then the standard powerset construction is used to make it deterministic. For EMS problems this process is only slightly changed. Our counting automaton has state set $S' = 2^S$, and in every state $s \subset S$ a map $m_e$ is added for every element $e \in s$. This map keeps track of possible values for the sums of the evaluations over the $X_i$ restricted to the subtree below a node. The state transition function $\delta'$ is

$$\delta'(s_1, s_2, \sigma) = \{\delta(e_1, e_2, \sigma, b) | e_1 \in s_1 \wedge e_2 \in s_2 \wedge b \in B\}.$$

The maps are sets of $l$ by $m$ matrices. The map for the initial state is just a singleton set with a zero matrix. The set of maps for a non-terminal node $v$ labeled $\sigma$ and with sons $v_1$ and $v_2$ having states $s(v_1)$ and $s(v_2)$ and maps $m_e(v_1)$ and $m_e(v_2)$ is

$$m_e(v) \leftarrow \bigcup_{\substack{e_1 \in s(v_1), e_2 \in s(v_2) \\ b:\, \delta(e_1, e_2, \sigma, b) = e}} \bigcup_{\substack{i_1 \in m_{e_1}(v_1) \\ i_2 \in m_{e_2}(v_2)}} \{b^T f(v) + i_1 + i_2\},$$

where $f(v)$ is the vector $(f_1(v), \ldots, f_m(v))$. The maps of those substates of the root falling in $A$ are precisely the possible value combinations for the $|X_i|_j$, so an EMS property or MS extremum problem can be solved by evaluating the corresponding evaluation relation $\psi$ or evaluation function $F$ for these value combinations. We have defined $|G|$ in Section 3. This definition corresponds to unary encoding of natural numbers and the cost will be polynomial in $|G|$. This is because all sums are less than the size of the problem instance encoding and the number of different values in the matrices of the maps is $O(|G|)$ so there are only $O(|G|^{lm})$ different matrices in the maps and the problem can be solved in time $O(|G|^{2lm})$, with the uniform arithmetic cost measure. This type of algorithm is called pseudopolynomial by Garey and Johnson [22].

THEOREM 5.4. *For each EMS property $P$, and for each class $K$ of graphs of universally bounded treewidth, the problem $P(G, f_1, \ldots, f_m, K_1, \ldots, K_t)$ for $G \in K$ can be decided in time polynomial in $|G|$.*

THEOREM 5.5. *For each class $K$ of graphs of universally bounded treewidth, every EMS extremum problem $P$ can be solved in time polynomial in $|G|$ for $G \in K$.*

For a linear extremum problem with objective function $\Sigma_{ij} a_{ij} |X_i|_j$, the maps can be replaced by counters keeping track of the extremum of the objective function over all values for the $X_i$ within each equivalence class defined by a state of $M$. The linearity of the objective function ensures that an extended principle of optimality will hold: every optimal solution on a subtree $T$ with root $n$, state $s$, and descendants $n_1$ and $n_2$ will be

optimal solutions also over subtrees rooted at $n_1$ and $n_2$ for some states $s_1$ and $s_2$ combined with an optimal choice (of memberships in the $X_i$) for $n$ leading to state $s$. All counters will be zero in the initial state, and they are updated as follows for node $n$ with sons $n_1$ and $n_2$:

$$c_e(n) \leftarrow \max_{\substack{e_1 \in s(n_1),\, e_2 \in s(n_2) \\ b:\, \delta(e_1, e_2, \sigma(n), b) = e}} \left( \sum_{ij} f_j(n) b_i a_{ij} + c_{e_1}(n_1) + c_{e_2}(n_2) \right).$$

The states in $S$ define a partitioning of the set of finite binary trees labeled with elements from $B$. The counter value for state $e$ of a node $n$ is the maximum, over all labelings of the subtree of $T$ rooted at $n$ falling in class $e$, of the objective function. This observation easily gives an induction proof over the depth of $T$ that the answer to the optimization problem is found from the root state $s(r)$ and its counters by

$$\max_{e \in s(r) \cap A} c_e(r).$$

This method gives linear time algorithms under the uniform cost measure as defined by Aho *et al.* [1], where an arithmetic operation or comparison is charged constant cost. Under the logarithmic cost measure the problem is solved in $O(n \log n)$.

THEOREM 5.6. *For each class $K$ of graphs of universally bounded treewidth, every EMS linear extremum problem $P$ can be solved in linear time with the uniform cost measure, for $G \in K$, if $G$ is given with a tree decomposition.*

For enumeration problems, the transition function is the same as before while counters are initialized to 1 and we will have the following update rule:

$$c_e(n) \leftarrow \sum_{\substack{e_1 \in s(n_1),\, e_2 \in s(n_2) \\ b:\, \delta(e_1, e_2, b) = e}} c_{e_1}(n_1) c_{e_2}(n_2).$$

The answer to the problem is obtained from the counters of the root:

$$\sum_{e \in s(r) \cap A} c_e(r).$$

THEOREM 5.7. *For each MS property $P$, and for each class $K$ of graphs of universally bounded treewidth, the enumeration problem of $P(G)$ for $G \in K$ can be solved in linear time in n with the uniform cost measure and if $G$ is given with a tree decomposition.*

COROLLARY 5.8. *For classes of graphs with universally bounded treewidth the following holds: All problems mentioned in Theorems* 3.5, 3.6, *and* 3.7 *can be decided in time linear in n, if the graphs are given together with a tree decomposition. All problems mentioned in Theorem* 3.8 *can be decided in time polynomial in* $|G|$.

Bodlaender [12] has shown that the problem of deciding treewidth $\leq k$ and, for a positive answer, finding a balanced (depth $O(\log n)$) tree decomposition with treewidth $\leq 3k + 2$ is in NC. For a balanced tree, our main algorithms are obviously in NC:

COROLLARY 5.9. *All MS and EMS problems are in NC for graphs of bounded treewidth, even when no tree decomposition is given with the graph.*

## 6. APPLICATION: RELIABILITY CALCULATIONS

The engineering literature on reliability analysis abounds with suggestions how to compute the probability of an event defined with a graph or a logical expression from a set of probabilities of independent primary events. In one family of problems, a communication system is analyzed. The set of primary events represents the status (up or down) of links and sites of the system, modelled as edges and vertices of a graph. The system event under study is then a communication ability such as the connectedness of the partial graph spanned by links and sites that are up, existence of a path between designated vertices involving only links and sites that are up, etc. Another family of problems is systems reliability calculations. Here the primary events can be any conditions arising, by internal malfunctioning or external influence, in an arbitrary technical system. The system event is expressed with a "fault tree," essentially a propositional logic function represented by a Boolean expression where the atoms are the primary events and the operators are boolean operators such as OR, AND, NOT, etc. With identification of common subexpressions, the expression is representable as a labeled directed acyclic graph (a *dag*). Both the problem families are easy when defined on structures whose underlying graphs have bounded treewidth. In this section we will consider the first family of problems. The second family is in certain respects both more difficult and more important (because of its relation to probabilistic logic and artificial intelligence, see, e.g., Genesereth and Nilsson [23]), and will be considered in future work.

The communication system is modeled by a graph with the set $X$ denoting a subset of those components (sites and links, represented by vertices and edges) whose influence on system behavior is under analysis. $X$ actually denotes the "down" subset of this set and we will call the whole

set the set of primary events. We thus identify, for simplicity but unnecessarily, events and components. In a real application one typically considers several failure modes of components. We require that the system event under study can be defined as a set of subsets of the primary event set, using an MS formula $\Phi(X)$ (and the set so defined is $\{A\colon G \models \Phi[A]\}$). It is easy to see from the examples in previous sections, on definitions of connectedness and existence of certain paths, that the system events typically of interest in communications reliability analysis can be defined in this way. We want to compute the probability of the system event, given probabilities of the primary events which are assumed to be independent. Again, this is a simplification but when the primary events are dependent they can often be modeled on combinations of some other events that are independent.

The problem we set out to solve is a weighted enumeration problem. The probability we seek is the sum over all subsets of primary events, $A$ such that $G \models \Phi[A]$, of the probability that $A$ is the set of "down" components. The interpretation of the graph (of bounded treewidth) into a binary tree will put nodes corresponding to primary events at the leaves. Higher up in the tree we will have nodes that are assigned states dependent on the assignment of primary event nodes to $X$. We can think of the state of a node as an event defined in terms of the primary events, so there are many events of the type "node $v$ is in state $s$." Although many nodes in the binary tree $T(G)$ may be derived from one primary event edge or vertex in $G$, only one will have its weight and the state transition function will ensure that the others will share its membership in $X$. Thus, the events (primary and defined) described in two disjoint subtrees of $T(G)$ will always be independent in those cases where they combine into a state that may lead to an accepting state. We now assign counters to the automaton for $\Phi(X)$ so that the counters of a node $v$ labeled $\sigma(n)$ will be assigned the probability distribution of its state. Denote the transition function $\delta\colon S \times S \times \Sigma \times \{0,1\} \to S$, where the last argument flags the nodes membership in $X$. For a primary event node, the weight $f_1$ will be its given probability, and for other terminal nodes $f_1$ is zero. The rule for terminal node $v$ and state $e \in S$ will be

$$c_e(n) \leftarrow \begin{cases} f_1(n) & \text{if } e = \delta(s_0, s_0, \sigma(n), 1) \\ 1 - f_1(n) & \text{if } e = \delta(s_0, s_0, \sigma(n), 0) \\ 0 & \text{otherwise,} \end{cases}$$

and, for a non-terminal node $v$ with sons $v_1$ and $v_2$, we have

$$c_e(v) \leftarrow \sum_{\substack{e_1 \in s(v_1),\, e_2 \in s(v_2) \\ \delta(e_1, e_2, \sigma(v), 0) = e}} c_{e_1}(v_1) c_{e_2}(v_2).$$

The probability of $\Phi(X)$ is obtained analogously to the answer to the enumeration problem as the sum of counts associated with accepting states of the root node.

## 7. Conclusions and Extensions

We have seen that large and easily described families of problems admit efficient algorithms on families of graphs of bounded treewidth. The results presented here imply, e.g., all the complexity results presented by Takamizawa, Nishizeki, and Saito [40], Arnborg and Proskurowski [6], Bern, Lawler, and Wong [8].

We do not expect that our results can be significantly strengthened, i.e., that there are many natural properties not definable as MS or EMS properties that can be decided in linear time, for graphs of bounded treewidth. But some strengthenings should be possible: e.g., besides the sums $\Sigma_{a \in A_i} f_j(a)$ one should consider $\max_{a \in A_i} f_j(a)$ in the evaluation term. It would also be nice to find a logic which could unify the monadic second-order part and the evaluation relation in one formula.

Our interpretability concept can be used also for other graph-like structures than those described above. Hypergraphs are an example of structures which in certain cases (e.g., when the clique representation is of bounded treewidth) can be interpreted on binary trees. The branch width [33] can be used instead of treewidth as a decomposition tool, and a similar coding as in Section 4 takes us to binary trees. It is also possible to interpret graphs of bounded treewidth into trees obtained from parse trees of the given graphs, as defined in [15, 19, 43].

Co-graphs [18] and complement $k$-decomposable [3] graphs can be interpreted into binary trees, but in these cases it is not possible to allow edge sets or evaluations on edges in the MS formulae (and thus the incidence relations are only indirectly accessible via the (directed or undirected) adjacency relation). So we have shown in the examples following Theorems 3.5 to 3.8 that the membership problem for a property closed under minor-taking and dominating set can be solved in linear time on such families. But it is unlikely that we can solve any of the problems in Theorems 3.7 and 3.8 involving edge weights on such families efficiently.

## Appendix A: Examples of Interpretability

We consider the following example: Let $K_1$ be the class of all $3 \times n$-grids for $n \geq 1$, and let $K_2$ be the class of all trees with three unary predicates. Assume for simplicity that the graphs in $K_1$ and $K_2$ are relational

structures with one binary irreflexive symmetric relation *Adj* (the adjacency relation of vertices). Then the formulas of an interpretation of $K_1$ into $K_2$ are

$$\varepsilon(x_1, x_2) := x_1 = x_2,$$

$$\alpha(x_1) := \exists x_2(x_1 \neq x_2 \wedge \mathrm{Adj}(x_1, x_2) \wedge \neg\exists x_3(x_1 \neq x_3 \wedge x_2 \neq x_3 \wedge \mathrm{Adj}(x_1, x_3)))$$

and

$$\begin{aligned}\lambda_{\mathrm{Adj}}(x, y) := \exists z(&\mathrm{Adj}(x, z) \\ &\wedge \mathrm{Adj}(y, z) \wedge ((P_1(x) \wedge P_2(y)) \vee (P_3(x) \wedge P_3(y)) \\ &\wedge (P_3(x) \wedge P_1(y)) \vee (P_3(x) \wedge P_2(y)))) \\ &\vee \exists z_1 \exists z_2(\mathrm{Adj}(z_1, z_2) \\ &\wedge \mathrm{Adj}(x, z_1) \wedge \mathrm{Adj}(y, z_2) \wedge ((P_1(x) \wedge P_1(y)) \\ &\vee (P_2(x) \wedge P_2(y)) \\ &\vee (P_3(x) \wedge P_3(y)))).\end{aligned}$$

Let $G \in K_1$ be the grid of Fig. 2. Then $\mathscr{A}(G)$ is the tree of Fig. 3.

For the interpretation $I = (\alpha, \lambda_{\mathrm{Adj}}, \varepsilon)$, it is now easy to see that it holds:

$$(\mathscr{A}(G))^I \approx G.$$

Moreover, it is not difficult to see that for each $3 \times n$-grid $G' \in K_1$ there is a tree $\mathscr{A}(G') \in K_2$ such that

$$G' \approx (\mathscr{A}(G'))^I.$$

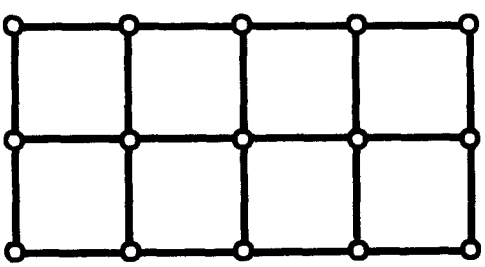


FIG. 2. Graph $G$ in $K_1$.

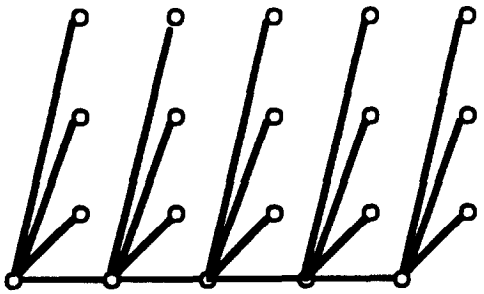


FIG. 3. $\mathscr{A}(G)$ in $K_2$.

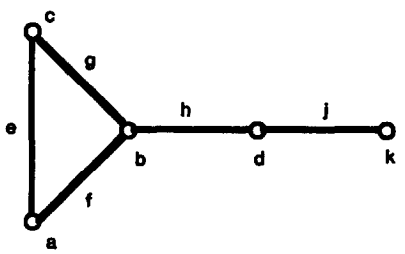



FIG. 4. $G$, a partial 2-tree.

We will now give an example of our main interpretability application, from partial 2-trees to binary trees. Consider the graph $G$ in Fig. 4.

A tree-decomposition of $G$ can be found in Fig. 5, where the sets $X_t$ for the vertices 1, 2, and 3 of the tree are illustrated as the content of the rectangles representing the vertices. The tree $T'$ representing the first transformation is shown in Fig. 6.

The predicates $P_V$ and $P_E$ represent the following vertices: $\{a_1, b_1, c_1, b_2, d_2, d_3, k_3\}$ and $\{e_1, f_1, g_1, h_2, j_3\}$. $P^1_{R_1}$, $P^2_{R_1}$, and $P^3_{R_1}$ could be

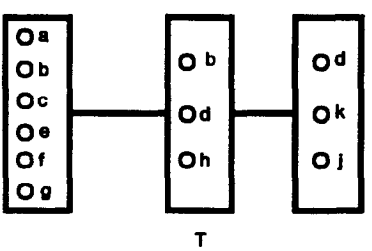



FIG. 5. Tree-decomposition of $G$.

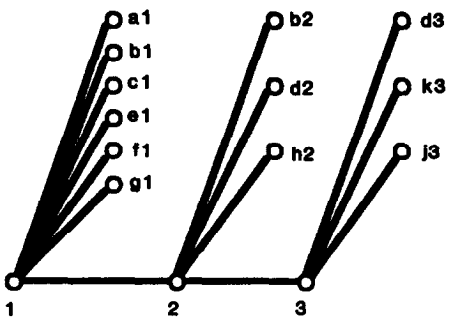
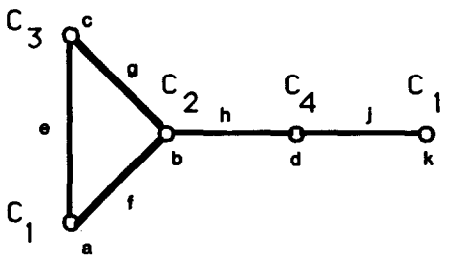



FIG. 6. $G$ transformed into $T'$.



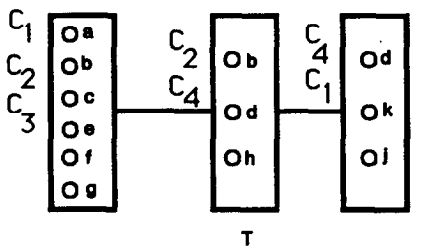




FIG. 7. Coloring for identifying equivalent vertices.

defined as $\{a_1, b_1, f_1, b_2, d_2, h_2, d_3, k_3, j_3\}$, $\{a_1, c_1, e_1\}$, and $\{b_1, c_1, g_1\}$, respectively. Now it is not difficult to define the unary and binary predicates of $G$ using the corresponding predicates of $T'$. The remaining problem is that there are different vertices in $T'$ which represent one and the same vertex of $G$, e.g., $b_1$ and $b_2$ represent both $b$. The coloring used to identify vertices is shown in Fig. 7. In logic language, $P_{C_1}$, $P_{C_2}$, $P_{C_3}$, and $P_{C_4}$ represent the sets $\{a_1, k_3\}$, $\{b_1, b_2\}$, $\{c_1\}$, and $\{d_2, d_3\}$, respectively. We can

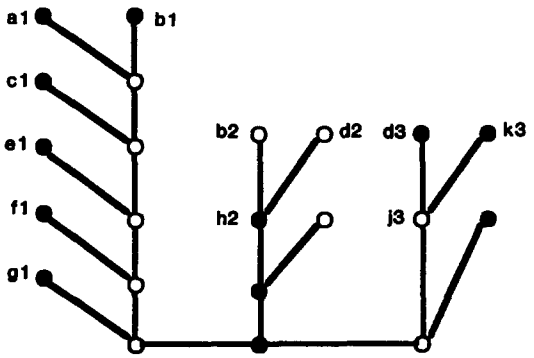



FIG. 8

see that the vertices in $G$ corresponding to $a_1$ and $k_3$ in $T$ are not the same, because the path between vertices 1 and 3 also passes 2 which does not contain a vertex colored with the common color of $a_1$ and $k_1$.

In Fig. 8, finally, we have made the tree $T''$ with vertices of degree 1 and 3 only. The vertices which are filled indicate those that have the predicate $P_c''$ set. So the connected regions of same color represent only one vertex each in $T'$.

## ACKNOWLEDGMENTS

We thank Steve Hedetniemi, Wojciech Janczewski, Lars Langemyr, Andrzej Proskurowski, Petra Scheffler, and Tom Wimer for stimulating discussions on the topic and Bruno Courcelle for pointing out an error in the manuscript.

## REFERENCES

1. AHO, HOPCROFT, AND ULLMAN, "Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, MA, 1972.
2. S. ARNBORG, Reduced state enumeration—Another algorithm for reliability evaluation, *IEEE Trans. Reliability* **R-27** (1978), 101–105.
3. S. ARNBORG, Efficient algorithms for combinatorial problems on graphs with bounded decomposability—A survey, *BIT* **25** (1985), 2–33.
4. S. ARNBORG, D. G. CORNEIL, AND A. PROSKUROWSKI, Complexity of finding embeddings in a $k$-tree, *SIAM J. Algebra Discrete Methods* **8** (1987), 277–284.
5. S. ARNBORG AND A. PROSKUROWSKI, Characterization and recognition of partial 3-trees, *SIAM J. Algebra Discrete Methods* **7** (1986), 305–314.
6. S. ARNBORG AND A. PROSKUROWSKI, Linear Time algorithms for NP-hard problems on graphs embedded in $k$-trees, *Discrete Appl. Math.* **23** (1989), 11–24.
7. J. BARWISE (Ed.), "Handbook of Mathematical Logic," North-Holland, Amsterdam, 1977.

8. M. W. BERN, E. L. LAWLER, AND A. L. WONG, Linear time computation of optimal subgraphs of decomposable graphs, *J. Algorithms* **8** (1987), 216–235.
9. U. BERTELE AND F. BRIOSCHI, "Nonserial Dynamic Programming," Academic Press, New York, 1972.
10. H. L. BODLAENDER, "Dynamic Programming on Graphs with Bounded Tree-width," MIT/LCS/TR-394, MIT, 1987.
11. H. L. BODLAENDER, "Classes of Graphs with Bounded Tree-width," RUU-CS-86-22, University of Utrecht, 1986.
12. H. L. BODLAENDER, "NC-Algorithms for Graphs with Bounded Tree-width," RUU-CS-88-4, University of Utrecht, 1988.
13. H. L. BODLAENDER, "Planar Graphs with Bounded Treewidth," RUU-CS-88-14, University of Utrecht, 1988.
14. H. L. BODLAENDER, "Improved Self-reduction Algorithms from Graphs with Bounded Treewidth," RUU-CS-88-29, University of Utrecht, 1988.
15. R. B. BORIE, R. G. PARKER, AND C. A. TOVEY, Automatic generation of linear algorithms from predicate calculus descriptions of problems on recursively constructed graph families, manuscript, July 21, 1988.
16. R. B. BORIE, "Recursively Constructed Graph Families: Membership and Linear Algorithms," Ph.D. thesis, Georgia Institute of Technology, December 1988.
17. K. COMPTON AND C. W. HENSON, A new method for proving lower bounds on the computational complexity of first-order theories, manuscript, preprint, 1987.
18. D. G. CORNEIL, H. LERCHS, AND L. STEWART BURLINGHAM, Complement reducible graphs, *Discrete Appl. Math.* **3** (1981) 163–174.
19. B. COURCELLE, Recognizability and second-order definability for sets of finite graphs, preprint, Universite de Bordeaux, I-8634, Jan. 1987. See also *Information and Comput.* **85** (1990) 12–75.
20. B. COURCELLE, "The Monadic Second Order Logic of Graphs. III. Tree-width, Forbidden Minors, and Complexity Issues, Report I-8852, Bordeaux-1 University, 1988.
21. J. E. DONER, Decidability of the weak second-order theory of two successors, Abstract 65T-468, *Notices Amer. Math. Soc.* **12** (1965), 819; (1966), 513.
22. R. FAGIN, Generalized first-order spectra and polynomial-time recognizable sets, *in* "Complexity and Computation," (R. Karp, Ed.) SIAM-AMS Proceedings, Vol. 7, Amer. Math. Soc., Providence, RI.
23. M. R. GAREY AND D. S. JOHNSON, "Computers and Intractability," Freeman, San Francisco, 1979.
24. M. R. GENESERETH AND N. NILSSON, "Logical Foundations of Artificial Intelligence," Morgan Kaufmann, Los Altos, CA, 1987.
25. Y. GUREVICH, Monadic second-order theories, in "Model-Theoretic Logics," Chap. XIII, pp. 479–506, (Barwise and Feferman, Eds.), Springer-Verlag, New York, 1985.
26. S. T. HEDETNIEMI, Open problems concerning the theory of algorithms on partial $k$-trees, working paper, 1987.
27. H. IMMERMAN, Languages which capture complexity classes, *SIAM J. Comput.* **4** (1987), 760–778.
28. D. S. JOHNSON, The NP-completeness column: An ongoing guide, *J. Algorithms* **6** (1984), 434–451.
29. D. S. JOHNSON, The NP-completeness column: An ongoing guide, *J. Algorithms* **8** (1987), 285–303.
30. M. O. RABIN, A simple method of undecidability proofs and some applications, *in* "Logic Methods Philos. Sci. Proc." Jerusalem, 1964, pp. 58–68.
31. M. O. RABIN, Decidable theories, *in* "Handbook of Mathematical Logic," (K. J. Barwise, Ed.), pp. 595–629, North-Holland, Amsterdam, 1977.

32. N. ROBERTSON AND P. D. SEYMOUR, Graph minors. II. Algorithmic aspects of tree width, *J. Algorithms* **7** (1986), 309–322.
33. N. ROBERTSON AND P. D. SEYMOUR, Graph minors X, preprint.
34. N. ROBERTSON AND P. D. SEYMOUR, Graph minors XIII; The disjoint path problem," preprint, Sept. 1986.
35. A. ROSENTHAL, Computing the reliability of a complex network, *SIAM J. Appl. Math.* **32** (1977), 384–393.
36. J. R. SHOENFIELD, "Mathematical Logic," Addison–Wesley, Reading, MA, 1967.
37. P. SCHEFFLER, Linear-time algorithms for NP-complete problems restricted to partial $k$-trees, preprint AdW d DDR, R-Math-03/87, Karl-Weierstrass-Inst. für Math., Berlin, 1987.
38. P. SCHEFFLER, Die Baumweite von Graphen als ein Mass für die Kompliziertheit algorithmischer Probleme, preprint AdW d DDR, R-Math-04/89, Karl-Weierstrass-Inst. für Math., Berlin, 1989.
39. P. SCHEFFLER, What graphs has bounded treewidth? *in* "Proceedings, 7th Fischland-Colloquium Discrete Mathematics and Applications, Wustrow, 1988; *Rostock Math. Kolloq.*, to appear.
40. D. SEESE, Tree-partite graphs and the complexity of algorithms, Preprint P-Math 08/86, Akademie der Wissenschaften der DDR, Karl Weierstrass Institut für Mathematik, 1986.
41. K. TAKAMIZAWA, T. NISHIZEKI, AND N. SAITO, Linear-time computability of combinatorial problems on series-parallel graphs, *J. Assoc. Comput. Math.* **29** (1982), 623–641.
42. A. TARSKI, A. MOSTOWSKI, R. M. ROBINSON, "Undecidable Theories," North-Holland, Amsterdam, 1953.
43. J. W. THATCHER AND J. B. WRIGHT, Generalized finite automata theory with an application to a decision problem in second-order logic, *Math. Systems Theory* **2** (1968), 57–81.
44. T. V. WIMER, "Linear algorithms on $k$-terminal graphs," Ph.D. thesis URI-030, Clemson University.
45. T. V. WIMER, S. T. HEDETNIEMI, AND R. LASKAR, A methodology for constructing linear graph algorithms, DCS, Clemson University, September 1985.