Theory Comput. Systems **33**, 125–150 (2000)
DOI: 10.1007/s002249910009

Theory of Computing Systems

Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width*

B. Courcelle,[1] J. A. Makowsky,[2] and U. Rotics[2]

[1]Laboratoire d'Informatique,
Université Bordeaux-I,
33405 Talence, France
Bruno.Courcelle@labri.u-bordeaux.fr

[2]Department of Computer Science,
Technion – Israel Institute of Technology,
32000 Haifa, Israel
{janos,rotics}@cs.technion.ac.il

Abstract. Hierarchical decompositions of graphs are interesting for algorithmic purposes. There are several types of hierarchical decompositions. Tree decompositions are the best known ones. On graphs of tree-width at most k, i.e., that have tree decompositions of width at most k, where k is fixed, every decision or optimization problem expressible in monadic second-order logic has a linear algorithm. We prove that this is also the case for graphs of clique-width at most k, where this complexity measure is associated with hierarchical decompositions of another type, and where logical formulas are no longer allowed to use edge set quantifications. We develop applications to several classes of graphs that include cographs and are, like cographs, defined by forbidding subgraphs with "too many" induced paths with four vertices.

1. Introduction

The class of P_4-sparse graphs was introduced by Hoàng in his doctoral dissertation [Hoà] as the class of graphs for which every set of five vertices induces at most one P_4 (by a P_4 we mean a path on four vertices). This class contains the class of P_4-reducible

* J. A. Makowsky was partially supported by a Grant of the Israeli Ministry of Science for French–Israeli Cooperation (1994), by a Grant of the German–Israeli Binational Foundation (1995–1996), and by the Fund for Promotion of Research of the Technion – Israeli Institute of Technology.

graphs introduced by Jamison and Olariu in [JO1], as the class of graphs for which no vertex belongs to more than one induced P_4. These two classes contain the class of cographs, and have been studied intensively in recent years. Such a study is motivated by the practical applications of these classes in areas such as scheduling, clustering, and computational semantics. In [JO1] and [JO3] a unique tree presentation is proposed for the classes of P_4-reducible and P_4-sparse graphs, respectively. These tree presentations are used later in [JO4] and [JO2] to develop $O(|V| + |E|)$ time recognition algorithms for these classes. In [JO5] $O(|V| + |E|)$ time algorithms are proposed for solving five optimization problems on the class of P_4-sparse graphs: maximum size clique, maximum size stable set, minimum coloring, minimum covering by cliques, and minimum fill-in. If the tree presentation of the P_4-sparse graph is also given as input, then the running time of these algorithms is just $O(|V|)$ independently of the number of edges in the graph. Jamison and Olariu conclude their paper with

Problem 1 [JO5]. Find other optimization problems which can be solved in linear time on the class of P_4-sparse graphs.

Giakoumakis and Vanherpe in [GV] took up this line of research. They used the modular decomposition of a graph to obtain $O(|V| + |E|)$ time algorithms for the maximum weight clique and for the maximum weight stable set problems in the case of P_4-sparse graphs, and for the optimal weighted coloring and for the minimum weight clique cover problems in the case of P_4-reducible graphs. If the modular decomposition of the graph is given as input, then the running time of these algorithms is just $O(|V|)$.

Giakoumakis and Vanherpe also introduced in [GV] the classes of extended P_4-sparse and extended P_4-reducible graphs, and showed how to extend their results to these two classes of graphs, with minimal additional work.

Babel and Olariu introduced in [BO] the class of (q, t) graphs. A (q, t) graph is a graph in which no set with at most q vertices is allowed to induce more than t distinct P_4's. Clearly, it is assumed that $q \geq 4$. The class of $(q, q-4)$ graphs extends the class of P_4-sparse graphs. In particular $(5, 1)$ graphs are exactly the P_4-sparse graphs and $(4, 0)$ graphs are exactly the cographs.

Rusu, see [GRT], introduced the class of P_4-tidy graphs which extends the class of extended P_4-sparse graphs. Let G be a graph and let X be an induced P_4. A vertex v outside X is a *partner of* X if X and v together induce two P_4's. A graph is P_4-*tidy* if any induced P_4 has at most one partner.

In Section 3 we show that a wide class of decision and optimization problems on the class of P_4-sparse graphs is solvable in time $O(|V| + |E|)$ or in time $O(|V|)$ assuming that the modular decomposition of the graph is given as input. These problems are characterized by their expressibility in certain variations of Monadic Second-Order Logic, $MSOL(\tau_{1,p})$ (for decision problems) or $LinEMSOL(\tau_{1,p})$ (for optimization problems), the study of which was initiated by Courcelle and others in a sequence of papers [Cou1], [Cou2], [Cou4], [Cou5], [Cou6], [CM], [ALS]. Roughly speaking, $MSOL(\tau_1)$ is Monadic Second-Order Logic with quantification over subsets of vertices, but not of edges; $MSOL(\tau_{1,p})$ is the extension of $MSOL(\tau_1)$ by unary predicates representing labels attached to the vertices. $LinEMSOL(\tau_{1,p})$ is the extension of $MSOL(\tau_{1,p})$ which allows one to search for sets of vertices which are optimal with respect to some linear

evaluation function. The precise definitions are given in Section 2. A typical $MSOL(\tau_1)$ decision problem is k-colorability for fixed k. The maximum weight clique and the maximum weight stable set problems are $LinEMSOL(\tau_{1,p})$ definable. The optimal (weighted) coloring problem is not $LinEMSOL(\tau_{1,p})$ definable, see [Lau2].

A *labeled graph* is a graph with labels associated with its vertices, such that each vertex has exactly one label. A p-graph is a simple undirected loop-free labeled graph with vertex labels in $\{1, 2, \ldots, p\}$. An unlabeled graph is considered as a 1-graph. In Section 3 we show that:

Theorem 2. *Let p be a fixed integer. Every $LinEMSOL(\tau_{1,p})$ problem on the class of P_4-sparse p-graphs can be solved in time $O(|V| + |E|)$ and the corresponding algorithm can be derived constructively from its $LinEMSOL(\tau_{1,p})$ definition. If the modular decomposition of the graph is given as input, then the running time of the algorithm is $O(|V|)$.*

Note that Theorem 2 also holds for any subclass of the class of P_4-sparse graphs, such as the classes of P_4-reducible graphs and cographs.

For example, in the terminology and numbering of [GJ], all the following problems are $LinEMSOL(\tau_{1,p})$ definable. So we have:

Corollary 3. *The following problems can be solved in linear time on the class of P_4-sparse p-graphs* (*and any of its subclasses*): *vertex cover* [*GT*1], *dominating set* [*GT*2], *domatic number for fixed k* [*GT*3], *k-colorability for fixed k* [*GT*4], *partition into cliques for fixed k* [*GT*15], *clique* [*GT*19], *independent set* [*GT*20], *and induced path* [*GT*23].

In Section 3 we prove Theorem 2 using *MSOL*-translation schemes and their induced transductions. The idea is to present a graph G by a tree built over some of its subgraphs (and called its modular decomposition) and to transfer the considered optimization problems on G into optimization problems on its modular decomposition. Since the modular decompositions of P_4-sparse graphs can be formalized as labeled partial 2-trees and can be constructed in linear time, and since *LinEMSOL* optimization problems have linear algorithms on partial 2-trees (see [ALS] and [CM]), we obtain a proof of Theorem 2. The basic tool here is the *MSOL* definable translation scheme, permitting a reduction of the optimization problems from graphs to their modular decompositions, while preserving the *LinEMSOL* expressibility. Using similar arguments Theorem 2 can be extended to the classes of $(q, q-4)$ graphs and P_4-tidy graphs. It is proved in [EHPR] that the so-called uniformly nonprimitive 2-structures which are certain directed graphs with labeled edges, have polynomial decision algorithms for problems expressible in *MSOL* without edge set quantifications. Cographs are isomorphic to a subclass of this class. The proof method is the one we use for Theorem 2.

In Section 4 we extend Theorem 2 to the class of graphs of bounded clique-width, first introduced by Courcelle et al. [CER]. We recall the notions of graph operations and clique-width presented in [CO].

We use three types of graph operations on p-graphs denoted $\oplus$, $\eta_{i,j}$, and $\rho_{i\to j}$. Informally, $G_1 \oplus G_2$ is the disjoint union of the p-graphs G_1 and G_2, $\eta_{i,j}(G)$ is the p-graph obtained by adding to G undirected edges connecting all vertices labeled i to

all the vertices labeled j in G, and $\rho_{i\to j}(G)$ is the p-graph obtained by changing all the i labels to j labels in G. A formal definition of these graph operations is given in Section 4.1.

With every p-graph G one can associate an algebraic expression built using operations of the three types mentioned above which defines G. We call such an expression a k-expression defining G, if all the labels in the expression are in $\{1, \ldots, k\}$. Clearly, k is greater than or equal to p. Also, for every p-graph G, there is an n-expression which defines G, where n is the number of vertices of G. Let $\mathcal{C}(k)$ be the class of p-graphs which can be defined by k-expressions. The clique-width of a p-graph G, denoted $cwd(G)$, is defined by $cwd(G) = Min\{k\colon\ G \in \mathcal{C}(k)\}$.

With these definitions we show:

Theorem 4. *Let $\mathcal{C}$ be a class of p-graphs of clique-width at most k (i.e., $\mathcal{C} \subseteq \mathcal{C}(k)$) such that there is a (known) $O(f(|E|, |V|))$ algorithm, which, for each p-graph G in $\mathcal{C}$, constructs a k-expression defining it. Then every $LinEMSOL(\tau_{1,p})$ problem on $\mathcal{C}$ can be solved in time $O(f(|E|, |V|))$. A corresponding algorithm can be effectively constructed from the logical formula describing the problem, and the parsing algorithm for the class.*

In the statement of Theorem 4 we must assume that we know an efficient parsing algorithm, because none is known for $\mathcal{C}(k)$ in general (there exist polynomial algorithms in special cases). Theorem 4 applies to any class of graphs of bounded clique-width for which an efficient parsing algorithm exists. There are many such classes. For example, the cliques, the cographs, and any class of graphs of treewidth at most k. We show that:

Proposition 5. *$(q, q-4)$ graphs and P_4-tidy graphs have clique-width at most q and 4, respectively, and for each $(q, q-4)$ (P_4-tidy) graph G, a q-expression (4-expression) defining it can be constructed in $O(|V| + |E|)$ time.*

From Theorem 4 and Proposition 5, we get a second proof of Theorem 2. This proof is based on graph operations and clique-width, and constructs algorithms for solving $LinEMSOL(\tau_{1,p})$ problems on P_4-sparse graphs, different from those constructed by the first proof of Theorem 2 mentioned above. Although Theorem 4 subsumes Theorem 2, we give a specific proof of Theorem 2 because it is more direct, hopefully usable in other similar situations, and does not use the machinery of the Feferman–Vaught theorem used in the proof of Theorem 4. Theorem 4 is interesting by its generality. Sections 3 and 4 can be read independently.

Courcelle and Mosbah [CM] also considered the logics $MSOL(\tau_2)$ and $EMSOL(\tau_{2,p})$ (which are similar to the logics mentioned above, but with quantifications over subsets of edges allowed). They showed that Theorem 2 can be extended also for all the $EMSOL(\tau_{2,p})$ optimization problems on each class of graphs of tree width at most k. However, our next result shows that this extension cannot be done for P_4-tidy, $(q, q-4)$, P_4-sparse, cographs, and all graph classes which contain the cliques, provided that $\mathbf{P} \neq \mathbf{NP}$.

For (edge-)labeled graphs this is easy to see, since every graph can be presented as a labeled clique with exactly the original edges labeled with a specific label. However, it

is also true for unlabeled graphs, provided that $\mathbf{P} \neq \mathbf{NP}$ on unary languages. We denote by $\mathbf{P}_1$ ($\mathbf{NP}_1$) the class of languages over one letter (also called tally languages), which are in $\mathbf{P}$ ($\mathbf{NP}$). In Section 5 we show that:

Theorem 6. *If $\mathbf{P}_1 \neq \mathbf{NP}_1$, then there is an MSOL$(\tau_2)$ definable decision problem over the class of cliques which is not solvable in polynomial time.*

An extended abstract of this paper was presented in [CMR].

2. Background

2.1. *Graphs as Logical Structures*

In what follows, we use the term *graph* for finite nonempty undirected graphs without self-loops or multiple edges. We use the term *labeled graph* for graphs having labels which are associated with their vertices such that each vertex has exactly one label. A *p-graph* is a labeled graph with (vertex) labels in $\{1, 2, \ldots, p\}$. An unlabeled graph G is considered as a 1-graph such that all the vertices of G are labeled by 1.
The following are the two most common (labeled) graph presentations, for logically oriented work:

Definition 7 (The Vocabularies τ_1 and $\tau_{1,p}$). We denote by τ_1 the vocabulary $\{E\}$ consisting of one binary relation symbol E. For a graph G, we denote by $G(\tau_1)$ the presentation of G as a logical structure $\langle V, E\rangle$, where V is the domain of the logical structure which consists of the set of vertices of G and E is the binary relation corresponding to the adjacency matrix of G.

We denote by $\tau_{1,p}$ the vocabulary $E, U_1, \ldots, U_p$, where p is any fixed integer. For a p-graph G, we denote by $G(\tau_{1,p})$ the presentation of G as a logical structure $\langle V, E, U_1, \ldots, U_p\rangle$, where V and E are the same as above, and $U_1, \ldots, U_p$ are the unary predicates corresponding to the labels of the vertices of G.

Note that we use a vocabulary $\tau_{1,p}$ for expressing properties of labeled graphs in general. Such properties make no reference to labels larger than p that may exist in the considered graph.

Remark. Certain structures of type $\tau_{1,p}$ do not represent p-graphs, either because the predicates U_i do not form a partition of the domain, or because E is not symmetric or both. A first-order formula can express that a given structure actually represents a p-graph. We only consider $\tau_{1,p}$ structures representing p-graphs without any further notice.

Definition 8 (The Vocabularies τ_2 and $\tau_{2,p}$). We denote by τ_2 the vocabulary $\{R, P_E, P_V\}$ consisting of one binary relation symbol R, and two unary relation symbols P_E, P_V. For a graph G, we denote by $G(\tau_2)$ the presentation of G as a logical structure

$\langle V \cup E, R, P_E, P_V \rangle$, where the domain of the logical structure consists of the set of vertices and edges of G, R is a binary relation, such that (e, v) is in R if and only if v is a vertex of G which is incident with the edge e of G, and P_V (resp. P_E) is a unary predicate such that v (resp. e) is in P_V (resp. P_E) if and only if v (resp. e) is a vertex (resp. an edge) of G.

We denote by $\tau_{2,p}$ the vocabulary $\{R, P_E, P_V, U_1, \ldots, U_p\}$, where p is any fixed integer. For an edge- and vertex-labeled graph G, we denote by $G(\tau_{2,p})$ the presentation of G as a logical structure $\langle V \cup E, R, P_E, P_V, U_1, \ldots, U_p \rangle$, where R, P_E, P_V are as above, and $U_1, \ldots, U_p$ are the unary predicates corresponding to the labels of the vertices and edges of G.

2.2. *Monadic Second-Order Logic Decision and Optimization Problems*

We recall that Second-Order Logic (*SOL*) is like first-order logic, but also allows variables and quantification over relation variables of various but fixed arities. Monadic Second-Order Logic (*MSOL*) is the sublogic of *SOL* where relation symbols are restricted to being unary. More details on the definition of *MSOL* can be found in [Cou7], [EF], and [Pap]. For a set variable X and a first-order variable u, we denote by $X(u)$ the atomic formula indicating that $u \in X$.

Graphs are a special case of finite structures. Therefore, before concentrating on graphs, we start with the following definitions and facts concerning finite structures. In what follows we are concerned only with finite structures, therefore whenever we use the term *structure* we mean *finite structure*. Let τ denote any vocabulary consisting of a finite set of relation symbols, and let K be any class of τ-structures. We denote by $Str(\tau)$ the class of all τ-structures.

Definition 9 ($MSOL(\tau)$ Decision Problem over K). We say that a decision problem is an $MSOL(\tau)$ *decision problem over* K if it can be expressed in the following form: given a τ-structure $\mathcal{A} \in K$ does $\mathcal{A} \models \varphi$, where φ is a closed *MSOL* formula over τ, hold? Note that φ and K are not part of the problem instance, which consists just of $\mathcal{A}$. In the case where class K consists of all τ-structures, $K = Str(\tau)$, we say that a problem which can be stated as above is an $MSOL(\tau)$ decision problem.

Example 1. The 3-colorability problem is an $MSOL(\tau_1)$ problem, since it can be stated as follows: given a graph G, presented as a logical structure $G(\tau_1)$, does $G(\tau_1) \models \varphi$, where φ is the closed $MSOL(\tau_1)$ formula defined by

$$\varphi \equiv \exists X_1, X_2, X_3 (Partition(X_1, X_2, X_3) \wedge IndSet(X_1) \wedge IndSet(X_2) \wedge IndSet(X_3)),$$

where $Partition(X_1, X_2, X_3)$ is defined by

$$\begin{aligned} Partition(X_1, X_2, X_3) \equiv\ & \forall v (X_1(v) \vee X_2(v) \vee X_3(v)) \\ & \wedge \neg \exists u ((X_1(u) \wedge X_2(u)) \vee (X_1(u) \wedge X_3(u)) \\ & \vee (X_2(u) \wedge X_3(u))), \end{aligned}$$

and $IndSet(X)$ is defined by

$$IndSet(X) \equiv \forall u, v((X(u) \wedge X(v)) \longrightarrow \neg E(u, v)),$$

hold? Let $f_1, f_2, \ldots, f_m$ be m function symbols for some fixed integer m. For a set variable X_i and an assignment z we use $|z(X_i)|_j$ as a short notation for $\sum_{a \in z(X_i)} f_j(a)$. We denote by $|A|$ the cardinality of a finite set A.

Definition 10 ($LinEMSOL(\tau)$ Optimization Problems over K). We say that an optimization problem P is a $LinEMSOL(\tau)$ *optimization problem over* K if it can be expressed in the following form: given a τ-structure $\mathcal{A} \in K$, and m evaluation functions $f_1, \ldots, f_m$ associating integer values to the elements of $\mathcal{A}$, find an assignment z to the free variables in θ such that

$$\sum_{\substack{1 \le i \le l \\ 1 \le j \le m}} a_{ij} |z(X_i)|_j = opt \left\{ \sum_{\substack{1 \le i \le l \\ 1 \le j \le m}} a_{ij} |z'(X_i)|_j \colon \langle \mathcal{A}, z' \rangle \models \theta(X_1, \ldots, X_l) \right\},$$

where θ is an $MSOL(\tau)$ formula having free set variables $X_1, \ldots, X_l$, *opt* is either *Min* or *Max*, and $\{a_{ij} \colon 1 \le i \le l,\ 1 \le j \le m\}$ are any integers. Since the coefficients a_{ij} can be negative we only deal with *Max*: a minimization is obtained from a maximization with negated coefficients. Note that $\theta(X_1, \ldots, X_l)$, K and the constants $\{a_{ij}\}$ are not part of the problem instance, which consists just of $\mathcal{A}$ and the evaluation functions $f_1, \ldots, f_m$.

For any assignment z to the free variables of θ which satisfies the above condition, we say that z *realizes a solution* to the problem P on $\mathcal{A}$ with evaluation functions $f_1, \ldots, f_m$.

In the case where the class K consists of all the τ-structures, $K = Str(\tau)$, we denote a $LinEMSOL(\tau)$ optimization problem over K shortly as a $LinEMSOL(\tau)$ problem. Note that the syntax of every $LinEMSOL(\tau)$ problem is completely defined by τ, $\theta(X_1, \ldots, X_l)$, the constants $\{a_{ij}\}$, and m.

Example 2. The maximum weight clique problem is to find for a given graph G, with weights assigned to its vertices, a clique C of G such that the total weight of the vertices of C is maximum. This problem is a $LinEMSOL(\tau_1)$ problem since it can be expressed as follows: given a graph G presented as a τ_1-structure, $G(\tau_1)$, and one evaluation function f_1 associating integer weight values to the vertices of $G(\tau_1)$, find an assignment z to the free set variable X_1 in θ such that

$$|z(X_1)|_1 = Max\{|z'(X_1)|_1 \colon \langle G(\tau_1), z' \rangle \models \theta(X_1)\},$$

where $\theta(X_1)$ is defined by

$$\theta(X_1) = \forall u, v((X_1(v) \wedge X_1(u) \wedge u \neq v) \longrightarrow E(u, v)).$$

Remark. Every $MSOL(\tau)$ decision problem can be expressed also as a $LinEMSOL(\tau)$) optimization problem. Thus, in what follows we are concerned only with $LinEMSOL(\tau)$ optimization problems.

2.3. *MSOL Translation Schemes and Transductions*

In this section we define the notion of a translation scheme. The idea is to define a new structure over vocabulary σ from a given structure over vocabulary τ by means of a finite set of logical formulas $\varphi, \psi_1, \ldots, \psi_m$ over τ. The formula φ defines the domain of the new structure and each relation R_i of arity k of the new structure is defined by the formula ψ_i with k free variables. This notion is called "interpretation" but we prefer the word "translation" to focus on the syntactic nature of the definition. The classical definition for First-Order Logic (see [EF]) is adapted for *MSOL*.

Definition 11 (Translation scheme Φ). Let τ and σ be two vocabularies, let $\sigma = \{R_1, \ldots, R_m\}$, and let $\rho(R_i)$ be the arity of R_i. Let $\Phi = \langle \varphi, \psi_1, \ldots, \psi_m \rangle$ be *MSOL* formulas over τ. We say that Φ is well formed for σ over τ if φ has one free first-order variable and no free set variables, and, for $1 \leq i \leq m$, each ψ_i has $\rho(R_i)$ free first-order variables and no free set variables. Such a $\Phi = \langle \varphi, \psi_1, \ldots, \psi_m \rangle$ is called a *τ-σ-translation scheme*. If φ is **true** and each ψ_i is quantifier free, Φ is called *quantifier free*.

In the following text we denote a τ-σ-translation scheme shortly as a translation scheme if τ and σ are clear from the context. With a translation scheme Φ one can naturally associate a (partial) function Φ^* from τ-structures to σ-structures. This function is called a transduction from τ-structures to σ-structures. For more general cases see [Cou3], [Cou7], [Mak], and [EO].

Definition 12 (The Induced Transduction Φ^*). Let $\mathcal{A}$ be a τ-structure, let Φ be a τ-σ-translation scheme, and let A be the domain of $\mathcal{A}$. The structure $\mathcal{A}_\Phi$ is defined as follows:

(i) The domain of $\mathcal{A}_\Phi$ is the set $A_\Phi = \{a \in A\colon \mathcal{A} \models \varphi(a)\}$.
(ii) The interpretation of R_i in $\mathcal{A}_\Phi$ is the set

$$\mathcal{A}_\Phi(R_i) = \{\bar{a} \in A_\Phi^{\rho(R_i)}\colon \mathcal{A} \models \psi_i(\bar{a})\}.$$

Note that $\mathcal{A}_\Phi$ is a σ-structure of cardinality at most $|A|$.
(iii) The partial function $\Phi^*\colon Str(\tau) \longrightarrow Str(\sigma)$ is defined by $\Phi^*(\mathcal{A}) = \mathcal{A}_\Phi$. Note that $\Phi^*(\mathcal{A})$ is defined if and only if $\mathcal{A} \models \exists x \varphi(x)$. In particular, if Φ is quantifier free, then Φ^* is a total function.

With a translation scheme Φ we can also naturally associate a function $\Phi^\sharp$ from $MSOL(\sigma)$-formulas to $MSOL(\tau)$-formulas. This function is called the backwards translation associated with Φ.

Definition 13 (The Backwards Translation $\Phi^\sharp$). Let θ be an $MSOL(\sigma)$-formula and let Φ be a τ-σ-translation scheme. The formula θ_Φ is defined inductively as follows:

(i) For θ of the form $x_1 = x_2$, θ_Φ is defined as $x_1 = x_2 \wedge \varphi(x_1) \wedge \varphi(x_2)$.
(ii) For $R_i \in \sigma$ and θ of the form $R_i(x_1, \ldots, x_m)$, θ_Φ is defined as $\psi_i(x_1, \ldots, x_m) \wedge \bigwedge_i \varphi(x_i)$.
(iii) For a set variable U and a first-order variable y, if θ is $U(y)$, then θ_Φ is $U(y) \wedge \varphi(y)$. Note that in our notation $U(y)$ is the same as $y \in U$.
(iv) For the boolean connectives, the translation distributes, i.e., if θ is of the form $(\theta_1 \vee \theta_2)$, then θ_Φ is defined as $(\theta_{1_\Phi} \vee \theta_{2_\Phi})$ and if θ is $\neg\theta_1$, then θ_Φ is $\neg\theta_{1_\Phi}$, and similarly for $\wedge$.
(v) For the existential quantifier of first-order variables, we use relativization, i.e., if θ is of the form $\exists y\theta_1$, then θ_Φ is defined as $\exists y(\varphi(y) \wedge \theta_{1_\Phi})$.
(vi) For the existential quantifier of a set variable U, the translation distributes, i.e., if θ is of the form $\exists U\theta_1$, then θ_Φ is defined as $\exists U(\theta_{1_\Phi})$.
(vii) The function $\Phi^\sharp$: $MSOL(\sigma) \longrightarrow MSOL(\tau)$ is defined by $\Phi^\sharp(\theta) = \theta_\Phi \wedge \theta_\varphi$, where θ_φ is the relativization of the free set variables in θ, say $X_1, \ldots, X_l$, defined by

$$\theta_\varphi = \bigwedge_{1 \le i \le l} \forall y(X_i(y) \to \varphi(y)).$$

If Φ is quantifier free, then (since θ_φ is logically equivalent to **true**) $\Phi^\sharp(\theta) = \theta_\Phi$.

From Definition 13 it follows that:

Observation 14. *For each translation scheme Φ, $\Phi^\sharp(\theta) \in MSOL$ provided $\theta \in MSOL$. If Φ is quantifier free, then $\Phi^\sharp(\theta)$ has the same quantifier depth as θ.*

The following fundamental property of translation schemes follows from the above definitions.

Theorem 15 [EF]. *Let $\Phi = \langle \varphi, \psi_1, \ldots, \psi_m \rangle$ be a τ-σ-translation scheme, let $\mathcal{A}$ be a τ-structure such that $\Phi^*(\mathcal{A})$ is defined, and let $\theta(v_1, \ldots, v_n, X_1, \ldots, X_l)$ be an $MSOL(\sigma)$-formula having n free first-order variables $v_1, \ldots, v_n$ and l free set variables $X_1, \ldots, X_l$. Then for every assignment z to the free variables of θ such that for every element $a = z(v_i)$, $\mathcal{A} \models \varphi(a)$, $1 \le i \le n$, and for every element $b \in z(X_j)$, $\mathcal{A} \models \varphi(b)$, $1 \le j \le l$, we have that*

$$\langle \mathcal{A}, z \rangle \models \Phi^\sharp(\theta)(v_1, \ldots, v_n, X_1, \ldots, X_l) \iff \langle \Phi^*(\mathcal{A}), z \rangle \models \theta(v_1, \ldots, v_n, X_1, \ldots, X_l).$$

With a translation scheme Φ we can also naturally associate a function $\Phi^\triangle$ from $LinEMSOL(\sigma)$ problems to $LinEMSOL(\tau)$ problems.

Definition 16 (The Backwards Translation $\Phi^\triangle$). Let P be a $LinEMSOL(\sigma)$ optimization problem given by σ, the $MSOL(\sigma)$ formula $\theta(X_1, \ldots, X_l)$ having free set variables $X_1, \ldots, X_l$, the (possibly negative) constants $\{a_{ij}\}$, and m (the number of evaluation

functions). Let Φ be a τ-σ-translation scheme.

(i) The optimization problem P_Φ is defined by τ, the $MSOL(\tau)$ formula $\Phi^\sharp(\theta(X_1, \ldots, X_l))$ having free set variables $X_1, \ldots, X_l$ (the same as of θ), the constants $\{a_{ij}\}$, and m (the number of evaluation functions).
(ii) The function $\Phi^\triangle$: $LinEMSOL(\sigma) \longrightarrow LinEMSOL(\tau)$ is defined by $\Phi^\triangle(P) = P_\Phi$.

Theorem 17. *Let P be a $LinEMSOL(\sigma)$ optimization problem, let $\Phi = \langle \varphi, \psi_1, \ldots, \psi_m \rangle$ be a τ-σ-translation scheme, and let $\mathcal{A}$ be a τ-structure such that $\Phi^*(\mathcal{A})$ is defined. Then an assignment z realizes a solution to the problem $\Phi^\triangle(P)$ on $\mathcal{A}$ with evaluation functions $f_1, \ldots, f_m$ if and only if z realizes a solution to the problem P on $\Phi^*(\mathcal{A})$ with evaluation functions $f_1, \ldots, f_m$ restricted to the domain of $\Phi^*(\mathcal{A})$.*

Proof. Let $\theta(X_1, \ldots, X_l)$ be the $MSOL(\sigma)$ formula used in the definition of P, and let z be an assignment which realizes a solution to the problem $\Phi^\triangle(P)$ on $\mathcal{A}$ with evaluation functions $f_1, \ldots, f_m$. Then the following condition holds:

$$\sum_{\substack{1 \le i \le l \\ 1 \le j \le m}} a_{ij} |z(X_i)|_j = Max \left\{ \sum_{\substack{1 \le i \le l \\ 1 \le j \le m}} a_{ij} |z'(X_i)|_j \colon \langle \mathcal{A}, z' \rangle \models \Phi^\sharp(\theta)(X_1, \ldots, X_l) \right\}.$$

By the above condition it follows that, for every element $a \in z(X_i), 1 \le i \le l, \mathcal{A} \models \varphi(a)$. Thus, from Theorem 15:

$$Max \left\{ \sum_{\substack{1 \le i \le l \\ 1 \le j \le m}} a_{ij} |z'(X_i)|_j \colon \langle \mathcal{A}, z' \rangle \models \Phi^\sharp(\theta)(X_1, \ldots, X_l) \right\}$$

$$= Max \left\{ \sum_{\substack{1 \le i \le l \\ 1 \le j \le m}} a_{ij} |z''(X_i)|_j \colon \langle \Phi^*(\mathcal{A}), z'' \rangle \models \theta(X_1, \ldots, X_l) \right\}.$$

Hence, z realizes a solution to the problem P on $\Phi^*(\mathcal{A})$ with evaluation functions $f_1, \ldots, f_m$ restricted to the domain of $\Phi^*(\mathcal{A})$. The other direction follows by a similar argument. □

2.4. *The Modular Decomposition of P_4-Sparse Graphs*

A set M of vertices of a graph G is called a *module* of G if every vertex outside M is either adjacent to all vertices in M or to none of them. A module M is called *strong* if for any module M_1 either $M \cap M_1 = \emptyset$ or one module contains the other. The *modular decomposition* of a graph G is based on a tree denoted by $T(G)$. The nodes of $T(G)$ are (in one-to-one correspondence with) the strong modules of G and a module M is a descendant of module M' in $T(G)$ iff $M \subseteq M'$. Consequently the leaves of $T(G)$ are the vertices of G and the strong module corresponding to a node of $T(G)$ consists of all leaves

of $T(G)$ that are descendants of that node. Each internal node is labeled by P, S, or N, as explained in Proposition 18. It can be shown that $T(G)$ is unique up to isomorphism. More details on how the tree $T(G)$ is constructed can be found in [GV], [BM], and [CH].

Let h be an internal node of $T(G)$. We denote by $M(h)$ the module corresponding to h which consists of the set of vertices of G of the subtree of $T(G)$ rooted at h. Let $\{h_1, \ldots, h_r\}$ be the set of sons of h in $T(G)$. We denote by $G(h) = \langle V(h), E(h)\rangle$ the *representative graph* of the module $M(h)$ defined by $V(h) = \{h_1, \ldots, h_r\}$ and

$$E(h) = \{(h_i, h_j) \mid \exists u, v(u \in M(h_i) \wedge v \in M(h_j) \wedge (u, v) \in E)\}.$$

Note that by the definition of a module, if a vertex of $M(h_i)$ is adjacent to a vertex of $M(h_j)$, then every vertex of $M(h_i)$ will be adjacent to every vertex of $M(h_j)$.

The modular decomposition $M(G)$ of G is the pair consisting of $T(G)$ and the mapping that associates with each node h of $T(G)$ the graph $G(h)$ (which is actually isomorphic to a subgraph of G).

It is clear that G can be reconstructed from $M(G)$. In particular, the vertices of G are the leaves of $T(G)$ and there is an edge between x and y iff x and y have ancestors h and h' which are sons of a same node k, and such that h and h' are linked by an edge in $E(k)$. This definition is expressible by a translation scheme taking as input $T(G)$ augmented with the edges of $E(h)$ for each internal node h.

From the construction of $T(G)$ it follows that:

Proposition 18. *Let G be any graph and let h be an internal node of $T(G)$. If $G(h)$ is a complete graph, then h is labeled S, if $G(h)$ is edgeless, then h is labeled P, otherwise h is labeled N.*

Recall that the neighborhood $Neigh(v)$ of a vertex v of G is defined as the set of vertices of G adjacent to v, i.e., $Neigh(v) = \{u|(u, v) \in E\}$.

Definition 19 (Prime Spider). A graph G is a *prime spider* if the vertex set of G can be partitioned into sets D, K, and R such that:

(i) D is a stable set (i.e., no two vertices in D are adjacent), K is a clique and $|D| = |K| \geq 2$.
(ii) R contains at most one vertex, i.e., $|R| \leq 1$, and if R contains one vertex say r, then r is adjacent to all the vertices in K and is not adjacent to any of the vertices in D.
(iii) There exists a bijection f between D and K such that either $Neigh(x) = \{f(x)\}$ for all vertices x in D or else $Neigh(x) = K - \{f(x)\}$ for all vertices x in D.

The triple (D, K, R) is called the *spider partition* of G.

Note that the edge-complement of a prime spider is also a prime spider. The following proposition is from [GV] based on [JO3]:

Proposition 20. *Let G be a P_4-sparse graph and let h be an internal N-node of $T(G)$. Then $G(h)$ is isomorphic to a prime spider.*

The following proposition is from [GRT]:

Proposition 21 [GRT]. *Let G be a P_4-tidy graph and let h be an internal N-node of $T(G)$. Then $G(h)$ is either isomorphic to a prime spider, to a cycle of five vertices C_5, to a path of five vertices P_5, or to the edge-complement of a path of five vertices $\overline{P_5}$.*

The following proposition is from [BO]:

Proposition 22 [BO]. *Let G be a $(q, q-4)$ graph and let h be an internal N-node of $T(G)$. Then $G(h)$ is either isomorphic to a prime spider or to a graph with at most q vertices.*

3. Linear Algorithms for Optimization Problems on P_4-Sparse Graphs

Our concern in this section is to reduce an optimization problem on a P_4-sparse graph G to one (of the same logical structure) on $M(G)$, efficiently solvable. We thus need an efficient presentation of modular decompositions of P_4-sparse graphs. A first obvious presentation, is to take $T(G)$ and to add the edges of the sets $E(h)$ (perhaps with a special marking to distinguish them from those of $T(G)$). However, these graphs will have too many edges. Our objective is to obtain graphs with "few edges", namely, partial k-trees. For the notion of partial k-tree see, e.g., [Bod2].

If a node h of $T(G)$ is an S-node, we mark it as such, and we omit the edges linking its sons. The marking will indicate the existence of the missing edges, and will be used by a translation scheme which translates $M(G)$ into G. If $G(h)$ is a prime spider, we present it by some colors and very few edges as indicated in the next definition. We consider P_4-sparse p-graphs, i.e., P_4-sparse graphs with vertices labeled in $1, \ldots, p$.

Definition 23 (The 2-Tree Modular Decomposition of G: 2-tree(G)). Let G be a P_4-sparse p-graph. We denote by 2-tree(G) the 2-*tree modular decomposition of* G constructed from $T(G)$ by adding more edges and labels to $T(G)$ according to the following rule:

— Let h be an N-node of G, let $G(h)$ be the representative graph of h which is isomorphic to a prime spider by Proposition 20, and let (D, K, R) be the spider partition of $G(h)$. Then:
- For every vertex x in D add to $T(G)$ the edge $(x, f(x))$, where f is the bijection from D to K defined in Definition 19.
- If $Neigh(x) = \{f(x)\}$ for all vertices x in D mark the N-node h of $T(G)$ as a red N-node. Otherwise, mark h as a black N-node.
- For every vertex x in D add a yellow label to x. For every vertex y in K add a blue label to y. For the one vertex r in R (if it exists) add a white label to r.

It is easy to see that:

Fact 24. *For every P_4-sparse p-graph G, 2-tree(G) is a partial 2-tree.*

Let G be a p-graph. Recall that the vocabulary $\tau_{1,p}$ consists of a binary relation symbol E and a finite set of unary predicate symbols $U_1, \ldots, U_p$, used to label the vertices of the p-graph. In order to present the graph 2-tree(G) as a logical structure we use the vocabulary $\tau_{1,p+10}$ which has $p+10$ unary predicate symbols $U_1, \ldots, U_{p+10}$ such that $U_1, \ldots, U_p$ are used to label the leaves of $T(G)$ in the same way as the vertices of the p-graph G, and $U_{p+1}, \ldots, U_{p+10}$, are denoted by P_{root}, P_{leaf}, P_S, P_P, P_N, P_{red}, P_{black}, P_{blue}, P_{yellow}, and P_{white}, respectively.

The meaning of the last ten unary predicates mentioned above is as follows:

— $P_{\text{root}}(x)$ is true if and only if x is the root of 2-tree(G). Note that using this predicate we can express that u is an ancestor of v in $T(G)$ or vice versa although $T(G)$ is presented as an undirected graph over the vocabulary $\tau_{1,p+10}$.
— $P_{\text{leaf}}(x)$ is true if and only if x is a leaf of the tree $T(G)$.
— $P_S(x)$ (resp. $P_P(x)$, $P_N(x)$) is true if and only if x is an S-node (resp. P-node, N-node) of the tree $T(G)$.
— $P_{\text{red}}(x)$ (resp. $P_{\text{black}}(x)$, $P_{\text{blue}}(x)$, $P_{\text{yellow}}(x)$, $P_{\text{white}}(x)$) is true if and only if x is marked red (resp. black, blue, yellow, white) in 2-tree(G).

Remark. Some vertices may satisfy more than one of the ten unary predicates defined above. Hence, a graph presented over $\tau_{1,p+10}$ may have vertices with more than one label. Since we require that labeled graphs have at most one label for each vertex, we can easily extend $\tau_{1,p+10}$ by adding more unary predicates, such that each vertex will have at most one label. For simplicity we do not specify this extension of $\tau_{1,p+10}$.

Theorem 25. *Let p be any integer. There exists a translation scheme Φ_1 such that for every P_4-sparse p-graph G we have $\Phi_1^*(\text{2-tree}(G)(\tau_{1,p+10})) \cong G(\tau_{1,p})$.*

Note that $\cong$ denotes isomorphism of logical structures. Theorem 25 states that there exists an *MSOL* translation scheme which reconstructs the original P_4-sparse graph G from its partial 2-tree presentation. The proof follows immediately from the definition of 2-tree(G).

Proposition 26. *Let $G = \langle V, E \rangle$ be any P_4-sparse p-graph. Then* 2-tree(G) *can be constructed in $O(|V| + |E|)$ time.*

Proof. Let G be a P_4-sparse p-graph. In [GV] it is shown how to construct $T(G)$ in $O(|V|+|E|)$ time. From Definition 23 it is easy to see that 2-tree(G) can be constructed from $T(G)$ in time linear in the number of nodes of $T(G)$. However, since the number of nodes of $T(G)$ is $O(|V|)$ (as proved in [Spi]), we get that the total construction of 2-tree(G) takes $O(|V| + |E|)$ time. □

The following theorem is from [Cou1], [CM], and [ALS] using the linear time algorithm (see [Bod1]) for constructing tree-decompositions of partial k-trees.

Theorem 27. *Let p and k be fixed integers. Every LinEMSOL$(\tau_{1,p})$ optimization problem on the class of partial k-trees can be solved in $O(|V|)$ time and the corresponding algorithm can be derived constructively from its LinEMSOL$(\tau_{1,p})$ definition.*

Theorem 27 holds also for the richer logical languages based on τ_2. Note that Theorem 27 has two different proofs, one of [Cou1] and [CM] and the other of [ALS], which construct different algorithms for solving *LinEMSOL*$(\tau_{1,p})$ (and also *LinEMSOL*$(\tau_{2,p})$) problems on the class of partial k-trees. We will show that:

Theorem 2. *Let p be a fixed integer. Every LinEMSOL$(\tau_{1,p})$ problem on the class of P_4-sparse p-graphs can be solved in time $O(|V| + |E|)$ and the corresponding algorithm can be derived constructively from its LinEMSOL$(\tau_{1,p})$ definition. If the modular decomposition of the graph is given as input, then the running time of the algorithm is $O(|V|)$.*

Proof. Let P be a *LinEMSOL*$(\tau_{1,p})$ optimization problem on the class of P_4-sparse p-graphs which is expressed as follows: given a P_4-sparse p-graph G presented over $\tau_{1,p}$, and m evaluation functions $f_1, \ldots, f_m$, find an assignment z to the free variables in θ such that

$$\sum_{\substack{1\le i\le l\\ 1\le j\le m}} a_{ij}|z(X_i)|_j = Max\left\{\sum_{\substack{1\le i\le l\\ 1\le j\le m}} a_{ij}|z'(X_i)|_j\colon \langle G(\tau_{1,p}), z'\rangle \models \theta(X_1, \ldots, X_l)\right\},$$

where θ is an *MSOL*$(\tau_{1,p})$ formula having free set variables $X_1, \ldots, X_l$, and $\{a_{ij}\colon 1 \le i \le l,\ 1 \le j \le m\}$ are (possibly negative) integers. Recall that for an assignment z as above we say that it realizes a solution to the problem P on G with evaluation functions $f_1, \ldots, f_m$.

We solve the problem P in $O(|V| + |E|)$ time by the following algorithm:

(i) Check whether the input p-graph G is a P_4-sparse graph using the algorithm of [GV]. If G is not a P_4-sparse graph stop with a "not legal input" answer.
(ii) Construct 2-tree(G) and present it over $\tau_{1,p+10}$.
(iii) Use the algorithm of [CM] or the algorithm of [ALS] (Theorem 27) to find an assignment z to the free variables in $\Phi_1^\sharp(\theta)$ which realizes a solution to the problem $\Phi_1^\triangle(P)$ on 2-tree(G) with evaluation functions $f_1, \ldots, f_m$. By Theorem 25 $\Phi_1^*(\text{2-tree}(G)(\tau_{1,p+10})) \cong G(\tau_{1,p})$. Hence, from Theorem 17 it follows that z also realizes a solution to the problem P on G with evaluation functions $f_1, \ldots, f_m$.

Step (i) can be done in $O(|V| + |E|)$ time as established in [GV], and by Proposition 26 step (ii) can be done in $O(|V| + |E|)$ time. By Fact 24 and Theorem 27 step (iii) can be done in $O(|V|)$ time, since the number of nodes and edges in 2-tree(G) is $O(|V|)$. Hence the running time of the algorithm is $O(|V| + |E|)$. If the modular decomposition $T(G)$ of G is given as an input, then the running time of the algorithm is $O(|V|)$, since step (i) is given as input and step (ii) can be done in $O(|V|)$ time. □

4. Linear Algorithms for Optimization Problems on Graphs of Bounded Clique-Width

4.1. *Graph Operations and Clique-Width*

For p-graphs G, H such that $G = \langle V, E, V_1, \ldots, V_p\rangle$ and $H = \langle V', E', V'_1, \ldots, V'_p\rangle$ and $V \cap V' = \emptyset$ (if this is not the case then replace H with a disjoint copy of H), we denote by $G \oplus H$ the disjoint union of G and H such that

$$G \oplus H = \langle V \cup V', E \cup E', V_1 \cup V'_1, \ldots, V_p \cup V'_p\rangle.$$

For a p-graph G as above we denote by $\eta_{i,j}(G)$, where $i \neq j$, the p-graph obtained by connecting all the vertices labeled i to all the vertices labeled j in G. Formally:

$$\eta_{i,j}(G) = \langle V, E', V_1, \ldots, V_p\rangle, \qquad \text{where} \quad E' = E \cup \{(u, v)\colon\ u \in V_i,\ v \in V_j\}.$$

For a p-graph G as above we denote by $\rho_{i\to j}(G)$ the renaming of i into j in G, $i \neq j$, such that

$$\rho_{i\to j}(G) = \langle V, E, V'_1, \ldots, V'_p\rangle,$$
$$\text{where} \quad V'_i = \emptyset, \quad V'_j = V_j \cup V_i, \quad \text{and} \quad V'_q = V_q \qquad \text{for} \quad q \neq i, j.$$

These graph operations have been introduced in [CER] for characterizing graph grammars. For every vertex v of a graph G and $i \in \{1, \ldots, p\}$, we denote by $i(v)$ the p-graph consisting of one vertex v labeled by i.

Example 3. A clique with four vertices u, v, w, x can be expressed as

$$\rho_{2\to 1}(\eta_{1,2}(2(u) \oplus \rho_{2\to 1}(\eta_{1,2}(2(v) \oplus \rho_{2\to 1}(\eta_{1,2}(1(w) \oplus 2(x)))))))).$$

Note the "temporary use" of the label 2.

With every p-graph G one can associate an algebraic expression built using operations of the three types mentioned above which defines G. We call such an expression a k-expression defining G, if all the labels in the expression are in $\{1, \ldots, k\}$. Clearly $k \geq p$. Also, for every p-graph G, there is an n-expression which defines G, where n is the number of vertices of G.

Definition 28 (Clique-Width). Let $\mathcal{C}(k)$ be the class of p-graphs which can be defined by k-expressions. The *clique-width* of a p-graph G, denoted $cwd(G)$, is defined by $cwd(G) = Min\{k\colon\ G \in \mathcal{C}(k)\}$.

The clique-width is a complexity measure on graphs somewhat similar to treewidth, which yields efficient graph algorithms provided the graph is given with its k-expression (for fixed k). A related notion has been introduced by Wanke [Wan] in connection with

graph grammars. $\mathcal{C}(1)$ is the class of edgeless graphs. The graphs in $\mathcal{C}(2)$ are exactly the cographs, see [CO]. They are definable from isolated vertices by $\oplus$, and the product $\otimes$ defined as

$$G \otimes H = \rho_{2\to 1}(\eta_{1,2}(G \oplus \rho_{1\to 2}(H))).$$

Trees have clique-width at most 3 (see [CO]).

Problem 29. Find a characterization of graphs of clique-width at most k, $k \geq 3$. Do there exist polynomial time algorithms for recognizing the classes $\mathcal{C}(k),\ k \geq 4$?

A polynomial time algorithm for recognizing the class $\mathcal{C}(3)$ is presented in [CHL+].

Lemma 30. *A p-graph with an underlying unlabeled graph of clique-width at most k has clique-width at most $p * k$.*

Proof. (Sketch). Let t be a k-expression for the underlying unlabeled graph. Let $c(v)$ denote the label in $\{1, \ldots, p\}$ of vertex v. A label i used in the subexpression $i(v)$ of t is replaced by $(i, c(v))$. Of course pairs (i, j) can be coded as integer labels between 1 and $p * k$ in such a way that labels $1, \ldots, p$ correspond to pairs $(1, 1), \ldots, (1, p)$. The additional information $c(v)$ can be maintained in the edge creations (i.e., η operations) and label renamings (i.e., ρ operations). Thus an edge creation will be replaced by $p * p$ edge creations, in order to handle the additional labels. □

4.2. *P_4-Tidy Graphs are of cwd ≤ 4 and $(q, q-4)$ Graphs are of cwd $\leq q$*

Let G and H be two disjoint graphs and let v be a vertex of G. We denote by $G[H/v]$ the graph K obtained by the substitution in G of H for v. Formally, $V(K) = V(G) \cup V(H) - \{v\}$ and

$$E(K) = E(H) \cup \{e\colon\ e \in E(G) \text{ and } e \text{ is not incident with } v\} \cup \{(u, w)\colon\ u \in V(H),\ w \in V(G) \text{ and } w \text{ is adjacent to } v \text{ in } G\}.$$

Proposition 31. *For all disjoint graphs G, H, and for every vertex v of G, $cwd(G[H/v]) = Max\{cwd(G), cwd(H)\}$.*

Proof. Let $q = Max\{cwd(G), cwd(H)\}$ and let h and g be q-expressions defining H and G, respectively. Since H is an unlabeled graph, it can be considered as a 1-graph such that all vertices of H are labeled by 1. Hence the q-expression h finally renames all labels into 1. The q-expression g must contain a unique subexpression of the form $i(v)$ corresponding to the initial label of v in the construction of G. By induction on

the structure of g, it can be shown that the q-expression obtained by replacing in g the subexpression $i(v)$ by the q-expression $\rho_{1\to i}(h)$ defines $G[H/v]$. We have shown that $cwd(G[H/v]) \leq q$.

If $cwd(G[H/v]) < q$, then there is a q_1-expression f defining $G[H/v]$, where $q_1 < q$. From f we can extract a q_1-expression for G by taking all the vertices of $V(G) - \{v\}$ in f and taking one vertex of f corresponding to a vertex of H (chosen arbitrarily) and omitting all the other vertices occurring in f. Here by omitting a vertex u from an expression t we mean: replace $i(u)$ in t with $\emptyset$ then replace every $\rho(\emptyset)$ and every $\eta(\emptyset)$ subexpression of t with $\emptyset$, and finally replace a subexpression of t of the form $\emptyset \oplus t_1$ or $t_1 \oplus \emptyset$ with t_1.

Likewise we can extract from f a q_1-expression for H by taking only the vertices of f corresponding to vertices of H and omitting all the other vertices. It follows that $Max\{cwd(G), cwd(H)\} = q_1 < q$, a contradiction. □

Recall that for any graph G we denote by $T(G)$ the tree obtained by the modular decomposition of G and for each internal node h of $T(G)$ we denote by $G(h)$ the representative graph of h defined in Section 2.4.

Proposition 32. *For every graph G, $cwd(G) = Max\{cwd(H)$: H is a representative graph of an internal node h in the modular decomposition of $G\}$.*

Proof. Using vertex substitutions we can build an expression which defines G, by the following procedure. Let r be the root of $T(G)$ and let R denote the singleton having one vertex r. Start by the expression $R[G(r)/r]$, substituting the representative graph $G(r)$ for the single vertex r of R. Then scan $T(G)$ in pre-order and whenever an internal node h is reached substitute $K[G(h)/h]$, i.e., substitute $G(h)$ for h, where K is the graph defined by the sequence of substitutions made so far. From the definitions of modular decomposition and representative graphs, it follows that the expression constructed by the above procedure defines the graph G, as a sequence of substitutions starting from the singleton R. The claim follows from Proposition 31, since $cwd(R) = 1$ and all the graphs substituted in the expression constructed above are representative graphs of internal nodes appearing in $T(G)$. □

Proposition 33. *For every prime spider G, $cwd(G) \leq 4$.*

Proof. Let G be a prime spider and let (D, K, R) be the spider partition of G. Let $D = \{d_1, \ldots, d_m\}$, let $K = \{k_1, \ldots, k_m\}$ and let $R = \{r\}$. By the definition of a prime spider either $Neigh(d_i) = k_i$ or $Neigh(d_i) = K - \{k_i\}$, for $1 \leq i \leq m$. In what follows we assume that $Neigh(d_i) = K - \{k_i\}$, for $1 \leq i \leq m$ (the other case can be handled similarly). For $1 \leq i \leq m$, let t_i be the expression defined by the following inductive definition:

(i) $t_1 = 2(k_1) \oplus 1(d_1)$,
(ii) $t_i = \rho_{3\to 1}(\rho_{4\to 2}(\eta_{2,4}(\eta_{1,4}(\eta_{2,3}(3(d_i) \oplus 4(k_i) \oplus t_{i-1})))))$.

Let $2 \leq i \leq m$, let $D_i = \{d_1, \ldots, d_i\}$, and let $K_i = \{k_1, \ldots, k_i\}$. We show by induction on i that the expression t_i defines the 2-graph which is the subgraph of G induced by $D_i \cup K_i$, such that all the vertices in D_i are labeled by 1 and all the vertices in K_i are labeled by 2. The claim trivially holds for $i = 2$. Assume that the claim holds for $i = j-1$, t_j is constructed from t_{j-1} by adding the two vertices d_j and k_j, labeling them by 3 and 4, respectively, and then adding edges as follows:

— Add edges between all the vertices labeled 3 to all the vertices labeled 2. This will add edges connecting d_j to all the vertices in K_{j-1}, which by the inductive hypothesis all have label 2.
— Add edges between all the vertices labeled 4 to all the vertices labeled 1. This will add edges connecting k_j to all the vertices in D_{j-1}, which by the inductive hypothesis all have label 1.
— Add edges between all the vertices labeled 4 to all the vertices labeled 2. This will add edges connecting k_j to all the vertices in K_{j-1}, which by the inductive hypothesis all have label 2.

Then as a last step all the vertices labeled by 4 (i.e., k_j) are relabeled with 2 and all the vertices labeled by 3 (i.e., d_j) are relabeled with 1. Clearly, all the vertices of D_j are labeled with 1 and all the vertices of K_j are labeled with 2. By the inductive hypothesis t_{j-1} defines the subgraph of G induced by $D_{j-1} \cup K_{j-1}$. Since the subgraph of G induced by $D_j \cup K_j$ can be obtained from the subgraph of G induced by $D_{j-1} \cup K_{j-1}$, by adding edges according to the above rules, we conclude that the claim holds also for $i = j$. Hence the expression t_m defines the subgraph of G induced by $D \cup K$. G can be obtained from its subgraph induced by $D \cup K$ by adding the vertex r and connecting it to all the vertices in K. This can be done by the following expression g:

$$g = \rho_{2\to 1}(\rho_{3\to 1}(\eta_{2,3}(3(r) \oplus t_m))).$$

The claim of the proposition follows since g is a 4-expression which defines G. □

Proposition 5. *$(q, q-4)$ graphs and P_4-tidy graphs have clique-width at most q and 4, respectively, and for each $(q, q-4)$ (P_4-tidy) graph G, a q-expression (4-expression) defining it can be constructed in $O(|V| + |E|)$ time.*

Proof. We prove the proposition for P_4-tidy graphs. The proof for $(q, q-4)$ graphs is along the same lines using Proposition 22 instead of Proposition 21. Let G be a P_4-tidy graph and let $T(G)$ be the tree obtained by the modular decomposition of G. By Proposition 32, in order to show that $cwd(G) \leq 4$ it suffices to show that, for each internal node h of $T(G)$, $cwd(G(h)) \leq 4$, where $G(h)$ is the representative graph of h in $T(G)$. If h is a P-node (S-node), then $G(h)$ is an edgeless graph (a clique), and has clique-width equal to 1 (2). If h is an N-node, then by Proposition 21 $G(h)$ is either a prime spider, a cycle of five vertices C_5, a path of five vertices P_5, or its complement $\overline{P_5}$. Since C_5, P_5, and $\overline{P_5}$ have $cwd \leq 4$, and prime spiders have $cwd \leq 4$ by Proposition 33, we have shown that $cwd(G) \leq 4$. A 4-expression defining G can be constructed in linear

time as follows:

(i) Construct the modular decomposition of G, $T(G)$ in time $O(|V| + |E|)$ as shown in [GV].
(ii) From the modular decomposition $T(G)$ construct an expression consisting of a sequence of vertex substitutions which defines G, as follows from the proof of Proposition 32. Since the number of vertices in $T(G)$ is $O(|V|)$ (as proved in [Spi]), this step can be done in time $O(|V| + |E|)$.
(iii) Convert the expression of vertex substitutions obtained in the previous step, to a 4-expression for G as follows from the proof of Proposition 31. This step can be done in time $O(|V| + |E|)$, since each graph H used in the substitutions is either an edgeless graph, a clique, a C_5 cycle, a P_5 path, its complement $\overline{P_5}$, or a prime spider for which a 4-expression can be constructed in time $O(|V(H)| + |E(H)|)$ as can be shown easily for the first five cases and as shown in the proof of Proposition 33 for the case of prime spiders. □

4.3. *The Feferman–Vaught Theorem*

In the proof of Theorem 4 we use a version of the Feferman–Vaught theorem [FV] adapted to *MSOL*. It is not clear who observed first that this adaptation to *MSOL* is true, but it is already in [Läu1] and [She] and follows from [Fef] and [Ehr]. For a good exposition, see [Gur1] and [Gur2].

We review some notation from [CM].

Definition 34. Let $\mathcal{A}$ be a τ-structure, let A be the domain of $\mathcal{A}$, and let φ be an *MSOL*(τ)-formula with free set variables $X_1, \ldots, X_n$. We denote by $sat(\mathcal{A}, \varphi)$ the set of n-tuples of subsets of A for which φ holds in $\mathcal{A}$. Formally:

$$sat(\mathcal{A}, \varphi) = \{(D_1, \ldots, D_n)\colon\ D_i \subseteq A, (\mathcal{A}, D_1, \ldots, D_n) \models \varphi(X_1, \ldots, X_n)\}.$$

The following is a special case of a classical result, for example, see [EF].

Lemma 35. *Let p, h, and n be fixed nonnegative integers. Then there are finitely many MSOL$(\tau_{1,p})$-formulas with free variables in $\{X_1, \ldots, X_n\}$ of quantifier depth $\leq h$ in the language expressing properties of p-graphs, up to tautological equivalence.*

Lemma 36. *For each p, each operation $f \in \{\rho_{i\to j}, \eta_{i,j}\colon\ i, j \in \{1, \ldots, p\}, i \neq j\}$ over p-graphs can be expressed by a quantifier free translation scheme Φ, i.e., $\Phi^* = f$. Hence, for every MSOL$(\tau_{1,p})$ formula θ, and for every p-graph G presented over $\tau_{1,p}$,*

$$sat(f(G), \theta) = sat(G, \Phi^\sharp(\theta)).$$

Proof. Immediate from the definitions of $\rho_{i\to j}$, and $\eta_{i,j}$ and Theorem 15. □

For any set D we denote by $\mathcal{P}(D)$ the power set of D, i.e., the set of all subsets of D. Let E, F be two subsets of D such that $E \cap F = \emptyset$, let $A \subseteq \mathcal{P}(E)^n$, and let $B \subseteq \mathcal{P}(F)^n$

(we call such A and B *separated*). We define $A \boxtimes B$ by

$$A \boxtimes B = \{(D_1 \cup D'_1, \ldots, D_n \cup D'_n)\colon (D_1, \ldots, D_n) \in A,\ (D'_1, \ldots, D'_n) \in B)\}.$$

Theorem 37 (Feferman–Vaught for *MSOL*). *For each p and for every $MSOL(\tau_{1,p})$ formula θ with free variables $X_1, \ldots, X_n$, two lists of $MSOL(\tau_{1,p})$ formulas $\varphi_1, \ldots, \varphi_m$ and $\psi_1, \ldots, \psi_m$ can be constructed such that all the formulas have the same free variables as θ and have quantifier depth no larger than the quantifier depth of θ, and, for every two p-graphs G and H presented over $\tau_{1,p}$ such that $V(G) \cap V(H) = \emptyset$,*

$$sat(G \oplus H, \theta) = \bigcup_{1 \le i \le m} sat(G, \varphi_i) \boxtimes sat(H, \psi_i).$$

Proof. Immediate reformulation of the result by Feferman–Vaught as discussed in [Gur2]. The result can also be proved directly using pebble games for *MSOL*. □

A more sophisticated construction where the union is disjoint can be derived as in Lemma 2.4 of [CM] but is not needed here.

4.4. *The Linear Time Algorithms*

The main ideas for proving Theorem 4 are as follows:

(i) If G is a graph defined by a k-expression g, then the set $sat(G, \varphi)$ can be computed by induction on the structure of g, with the help of auxiliary sets $sat(G', \psi)$, for finitely many formulas ψ, and finitely many graphs G' where the graphs G' are defined by subexpressions of g. Here we use the Feferman–Vaught theorem (see Theorem 37) and Lemma 36.

(ii) A value $h(sat(G, \varphi))$ can be computed by the same induction on g, where h is a homomorphism (in some sense as defined below).

(iii) $LinEMSOL(\tau_{1,p})$ problems fall in the framework of computing $h(sat(G, \varphi))$ for well-chosen functions h.

Let G be a graph, let $f_1, \ldots, f_m$ be m evaluation functions associating integer values to the vertices of G, let $D_1, \ldots, D_l \subseteq V(G)$, and let

$$h(D_1, \ldots, D_l) = \sum_{\substack{1 \le i \le l \\ 1 \le j \le m}} a_{ij} |D_i|_j,$$

where $\{a_{ij}\colon 1 \le i \le l,\ 1 \le j \le m\}$ are any integers, and $|D_i|_j$ (see Section 2.2) is a short notation for $\sum_{a \in D_i} f_j(a)$. For $A \subseteq \mathcal{P}(V(G))^l$, let

$$Max\_h(A) = Max\{h(D_1, \ldots, D_l)\colon (D_1, \ldots, D_l) \in A\}.$$

It is clear that, for separated A and B,

$$Max\_h(A \boxtimes B) = Max\_h(A) + Max\_h(B) \tag{1}$$

and, for general A and B,

$$Max\_h(A \cup B) = Max\{Max\_h(A),\ Max\_h(B)\}. \tag{2}$$

From Definition 10 it follows that a $LinEMSOL(\tau_{1,p})$ optimization problem over a class of graphs K can be formulated as the computation of $Max\_h(sat(G,\theta))$ for a given graph $G \in K$ presented over $\tau_{1,p}$, for fixed p, where θ is a fixed $MSOL(\tau_{1,p})$ formula.

For each k-expression g we denote by $Tree(g)$ the labeled tree corresponding to g. The leaves of $Tree(g)$ are the singletons in g (the basic graphs) labeled by their initial label from $\{1,\ldots,k\}$, and the internal nodes of $Tree(g)$ correspond to the operations appearing in g. For each internal node x of $Tree(g)$, we denote by $Graph(x)$ the k-graph defined by the k-expression corresponding to the subtree of $Tree(g)$ rooted at x.

We are now ready to prove Theorem 4, which we restate for convenience.

Theorem 4. *Let $\mathcal{C}$ be a class of p-graphs of clique-width at most k, $\mathcal{C} \subseteq \mathcal{C}(k)$, such that there is a (known) $O(f(|E|,|V|))$ algorithm, which, for each p-graph G in $\mathcal{C}$, constructs a k-expression defining it. Then every $LinEMSOL(\tau_{1,p})$ problem on $\mathcal{C}$ can be solved in time $O(f(|E|,|V|))$. A corresponding algorithm can be effectively constructed from the logical formula describing the problem, and the parsing algorithm for the class.*

Proof. Let P be a $LinEMSOL(\tau_{1,p})$ optimization problem over a class of p-graphs $\mathcal{C} \subseteq \mathcal{C}(k)$. As mentioned above P can be formulated as the computation of $Max\_h(sat(G,\theta))$ for a given p-graph $G \in \mathcal{C}$ presented over $\tau_{1,p}$. Since $G \in \mathcal{C}$ there is a k-expression g which defines G. By Lemma 36 and Theorem 37, the computation of $Max\_h(sat(G,\theta))$ can be done as follows:

(i) Traverse $Tree(g)$ from top to bottom starting from the root assigning formulas to the internal nodes of the tree according to the following rules:
  (a) Assign to the root the formula θ.
  (b) Let $\varphi_1,\ldots,\varphi_l$ be the formulas assigned to an internal node x by this process. If x corresponds to a unary operation of the form $\rho_{i\to j}$ or $\eta_{i,j}$, then use Lemma 36 to obtain formulas $\varphi'_1,\ldots,\varphi'_l$, such that, for $1 \le i \le l$,

$$sat(Graph(x),\varphi_i) = sat(Graph(y),\varphi'_i),$$

  where y is the son of x in $Tree(g)$. Assign all these formulas to y.

  Otherwise x corresponds to the binary operation $\oplus$. In this case use Theorem 37 to obtain $2l$ lists of formulas $\varphi'_{i,1},\ldots,\varphi'_{i,m_i}$, and $\psi'_{i,1},\ldots,\psi'_{i,m_i}$, for $1 \le i \le l$, such that

$$sat(Graph(x),\varphi_i) = \bigcup_{1\le j\le m_i} sat(Graph(u),\varphi'_{i,j}) \boxtimes sat(Graph(v),\psi'_{i,j}), \tag{3}$$

  where u and v are the two sons of x in $Tree(g)$. Assign all the $\varphi'_{i,j}$ formulas to u and all the $\psi'_{i,j}$ formulas to v.

(ii) Traverse $Tree(g)$ from bottom to top and, at each node x and for each formula φ assigned to x by the previous step, compute $Max\_h(sat(Graph(x), \varphi))$ as follows:
— If x is a leaf compute $Max\_h(sat(Graph(x), \varphi))$ directly.
— If x corresponds to a unary operation, set $Max\_h(sat(Graph(x), \varphi)) = Max\_h(sat(Graph(y), \varphi'))$, where y is the son of x, and φ' is the formula assigned to y by the previous step.
— If x corresponds to the binary operation $\oplus$ then using (1)–(3) compute $Max\_h(sat(Graph(x), \varphi))$ from the two lists of values: $Max\_h(sat(Graph(u), \varphi'_j))$, $Max\_h(sat(Graph(v), \psi'_j))$, for $1 \leq j \leq m$, where u and v are the sons of x in $Tree(g)$ and φ'_j and ψ'_j are the lists of formulas assigned to u and v by the previous step, respectively.
Also at each node x and each formula φ assigned to x keep one tuple of $sat(Graph(x), \varphi)$ having the value $Max\_h(sat(Graph(x), \varphi))$.

The correctness of the above procedure follows from Lemma 36 and Theorem 37.

For the complexity, the total time for handling the input graph G is $O(f(|V|, |E|))$ for constructing the k-expression g plus the total time for applying the above procedure. First note that the size of the tree $Tree(g)$ is $O(f(|V|, |E|))$. In step (i) of the above process the number of formulas assigned to each node is bounded by a constant (which does not depend on the size of the input graph G) since by Lemma 36, Theorem 37, and Observation 14 all these formulas are of quantifier depth no larger than the quantifier depth of θ, and by Lemma 35 the number of such formulas is bounded (up to tautological equivalence) by a constant which depends just on the size of θ and p. Hence, in step (ii) the computation done at each node by the above procedure is bounded by a constant (with the uniform cost measure), and the total time of the above procedure is bounded by $O(f(|V|, |E|))$. Note that if x is a leaf, then $Graph(x)$ is a singleton, which implies that $Max\_h(sat(Graph(x), \varphi))$ can be computed in a time that does not depend on the size of the input graph G, i.e., in constant time. Therefore the total complexity of handling the input graph G is $O(f(|V|, |E|)) + O(f(|V|, |E|)) = O(f(|V|, |E|))$. □

Remark. Every k-expression for a graph $G = \langle V, E \rangle$ can be transformed into a k-expression defining G of size $O(|V|)$. This transformation can be done in linear time by a tree transducer. Typically it will remove some redundancies or useless operation symbols (like a renaming $\rho_{i \to j}$ operation when there is no vertex labeled i). (In these complexity considerations, k is fixed.) Thus we could assume in the above proof that the size of $Tree(g)$ is $O(|V|)$.

5. Results That Do Not Extend to $MSOL(\tau_2)$

In this section we show that Theorems 2 and 4 do not hold when $LinEMSOL(\tau_{1,p})$ is replaced by $MSOL(\tau_2)$. Clearly, if these theorems do not hold for $MSOL(\tau_2)$, then they do not hold either for its extensions: $MSOL(\tau_{2,p})$, $LinEMSOL(\tau_2)$, and $LinEMSOL(\tau_{2,p})$. We prove Theorem 6 but to do that we need the following definitions and theorem due to [Fag]. We denote by $\tau_\emptyset$ the empty vocabulary, and we denote by SET the class of

finite structures over $\tau_\emptyset$. We denote by SOL^2 the formulas in SOL in which there are no function symbols, and the relation symbols are restricted to being either unary or binary. Recall that $\mathbf{P}_1$ ($\mathbf{NP}_1$) denotes the class of languages over one letter (also called tally languages), which are in $\mathbf{P}$ ($\mathbf{NP}$). Note that $\mathbf{P} = \mathbf{NP}$ implies $\mathbf{P}_1 = \mathbf{NP}_1$, but the other direction is not known. Note also that $\mathbf{P}_1 = \mathbf{NP}_1$ iff $\mathbf{EXPTIME} = \mathbf{NEXPTIME}$ (see [Boo] and [Har]).

Definition 38 (Spectrum, *BIN*).

(i) Let S be a set of structures over $\tau_\emptyset$. S is a *spectrum* if there exists a formula φ of the form $\exists X_1, X_2, \ldots, X_l \sigma$, such that σ is first order, $X_1, X_2, \ldots, X_l$ are the only free variables of σ, and, for every finite structure $\mathcal{A}$ over $\tau_\emptyset$, $\mathcal{A} \in S$ if and only if φ holds in $\mathcal{A}$. In this case we say that the spectrum S is definable by the formula φ.

(ii) We denote by *BIN* the set of all spectra definable by formulas using only one binary predicate symbol which presents a graph relation, i.e., a relation which is irreflexive and symmetric. In other words a spectrum S is in *BIN* if it can be defined by a formula φ of the form $\exists Q \sigma$, where σ is first order, such that Q is the only free variable in σ, and Q is a binary predicate symbol presenting a graph relation.

(iii) We observe that *BIN* is included in $\mathbf{P}_1$ iff for every spectrum S in *BIN* there exists a polynomial time deterministic Turing machine M, such that given an integer n presented as a string in unary notation as an input (i.e., the length of the input is n and not $\log(n)$), M accepts n if and only if the structure in *SET* having n elements is in S.

The following theorem is due to [Fag]:

Theorem 39. $\mathbf{P}_1 = \mathbf{NP}_1$ *if and only if* $BIN \subseteq \mathbf{P}_1$.

We are now ready to prove Theorem 6 which we restate here for convenience.

Theorem 6. *If* $\mathbf{P}_1 \neq \mathbf{NP}_1$, *then there is an* $MSOL(\tau_2)$ *definable decision problem over the class of cliques which is not solvable in polynomial time.*

Proof. Let $\mathcal{A}$ be a structure in *SET*. We denote by $K_\mathcal{A}$ the clique corresponding to $\mathcal{A}$, such that the number of elements in the domain of $\mathcal{A}$ equals the number of vertices of the clique $K_\mathcal{A}$.

Recall (see Definition 8) that $R(t, x)$ holds if and only if the vertex x is incident with the edge t. Let φ be an $SOL^2(\tau_\emptyset)$ sentence. We denote by $\varphi^\sharp$ the $MSOL(\tau_2)$ sentence which is constructed from φ by replacing every subformula $U(x, y)$ where U is a binary relation symbol by the formula

$$\exists t(U(t) \wedge R(t, x) \wedge R(t, y)).$$

Since in a clique all the edges between all pairs of vertices exist, each pair of vertices (x, y) can be identified by the unique edge t, incident to both x and y. Therefore, quantification over pairs of vertices in cliques can be replaced by quantification over edges, as indicated by the above formula which replaces the binary relation symbol $U(x, y)$. Therefore, for every structure $\mathcal{A}$ in *SET* and every $SOL^2(\tau_\emptyset)$ sentence φ,

$$\mathcal{A} \models \varphi \iff K_{\mathcal{A}}(\tau_2) \models \varphi^\sharp.$$

Assuming that, over the class of cliques, every $MSOL(\tau_2)$-definable decision problem can be solved in polynomial time, we get that $BIN \subseteq \mathbf{P}_1$. For, let S be a spectrum in *BIN*, then there is an $SOL^2(\tau_\emptyset)$ sentence φ which defines S. By our assumption on the cliques, there is a Turing machine M which given an integer n in unary presentation decides, in time bounded by a polynomial in n, whether $K_n(\tau_2) \models \varphi^\sharp$. Hence, by the above equality, the machine M decides in polynomial time in n whether $\mathcal{A} \models \varphi$, where n is the number of elements in $\mathcal{A}$. It follows that $S \in \mathbf{P}_1$, and hence that $BIN \subseteq \mathbf{P}_1$. By Theorem 39 this implies that $\mathbf{P}_1 = \mathbf{NP}_1$, a contradiction. □

Question 40. Can we still prove Theorem 6 if we replace the condition $\mathbf{P}_1 \neq \mathbf{NP}_1$ by the condition $\mathbf{P} \neq \mathbf{NP}$?

Acknowledgments

We are grateful to Joost Engelfriet for his many valuable comments and suggestions which helped to improve the content of this paper. We thank the referees for their careful reading and their helpful comments.

References

[ALS] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree decomposable graphs. *J. Algorithms*, 12:308–340, 1991.

[BM] H. Buer and R.H. Möhring. A fast algorithm for the decomposition of graphs and posets. *Math. Oper. Res*., 8:170–184, 1983.

[BO] L. Babel and S. Olariu. On the isomorphism of graphs with few P_4's. In M. Nagl, editor, *Graph Theoretic Concepts in Computer Science*, 21*st International Workshop, WG* '95, Lecture Notes in Computer Science, volume 1017, pages 24–36. Springer-Verlag, Berlin, 1995.

[Bod1] H.L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput*., 25:1305–1317, 1996.

[Bod2] H.L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoret. Comput. Sci*., 209:1–45, 1998.

[Boo] R.V. Book. Tally languages and complexity classes. *Inform. and Control*, 26:186–194, 1974.

[CER] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *J. Comput. System Sci*., 46:218–270, 1993.

[CH] A. Cournier and M. Habib. A new linear algorithm for modular decomposition. In S. Tison, editor, *Trees in Algebra and Programming, CAAP* '94: 19*th International Colloquium*, Lecture Notes in Computer Science, volume 787, pages 68–84. Springer-Verlag, Berlin, 1994.

[CHL+] D.G. Corneil, M. Habib, J.M. Lanlignel, B. Reed, and U. Rotics. Polynomial time algorithm for the 3-clique-width problem. In preparation.

[CM] B. Courcelle and M. Mosbah. Monadic second–order evaluations on tree-decomposable graphs. *Theoret. Comput. Sci.*, 109:49–82, 1993.

[CMR] B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width, Extended abstract. In J. Hromkovič and O. Sýkora, editors, *Graph Theoretic Concepts in Computer Science, 24th International Workshop, WG '98*, Lecture Notes in Computer Science, volume 1517, pages 1–16. Springer-Verlag, Berlin, 1998.

[CO] B. Courcelle and S. Olariu. Upper bounds to the clique-width of graphs. *Discrete Appl. Math.*, to appear. (http://dept-info.labri.u-bordeaux.fr/∼courcell/ActSci.html).

[Cou1] B. Courcelle. The monadic second-order logic of graphs, I: Recognizable sets of finite graphs. *Inform. and Comput.*, 85:12–75, 1990.

[Cou2] B. Courcelle. The monadic second-order logic of graphs, V: On closing the gap between definability and recognizability. *Theoret. Comput. Sci.*, 80:153–202, 1991.

[Cou3] B. Courcelle. Monadic second-order definable graph transductions: a survey. *Theoret. Comput. Sci.*, 126:53–75, 1994.

[Cou4] B. Courcelle. The monadic second-order logic of graphs, VI: On several representations of graphs by relational structures. *Discrete Appl. Math.*, 54:117–149, 1994.

[Cou5] B. Courcelle. The monadic second-order logic of graphs, VIII: Orientations. *Ann. Pure Appl. Logic*, 72:103–143, 1995.

[Cou6] B. Courcelle. The monadic second-order logic of graphs, X: Linear orders. *Theoret. Comput. Sci.*, 160:87–143, 1996.

[Cou7] B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformations*, volume 1, chapter 5, pages 313–400. World Scientific, Singapore, 1997.

[EF] H.D. Ebbinghaus and J. Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer-Verlag, New York, 1995.

[EHPR] J. Engelfriet, T. Harju, A. Prokurowski, and G. Rozenberg. Characterization and complexity of uniformly nonprimitive labeled 2-structures. *Theoret. Comput. Sci.*, 154:247–282, 1996.

[Ehr] A. Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fund. Math.*, 49:129–141, 1961.

[EO] J. Engelfriet and V. van Oostrom. Logical description of context-free graph languages. *J. Comput. System Sci.*, 55:489–503, 1997.

[Fag] R. Fagin. Generalized first-order spectra and polynomial time recognizable sets. *Proc. Amer. Math. Soc.*, 7:27–41, 1974.

[Fef] S. Feferman. Some recent work of Ehrenfeucht and Fraïssé. *Proceedings of the Summer Institute of Symbolic Logic*, Ithaca, NY, 1957, pages 201–209.

[FV] S. Feferman and R. Vaught. The first order properties of algebraic systems. *Fund. Math.*, 47:57–103, 1959.

[GJ] M.G. Garey and D.S. Johnson. *Computers and Intractability*. Mathematical Series. Freeman, San Francisco, CA, 1979.

[GRT] V. Giakoumakis, F. Roussel, and H. Thuillier. On P_4-tidy graphs. *Discrete Math. and Theoret. Comput. Sci.*, 1:17–41, 1997.

[Gur1] Y. Gurevich. Modest theory of short chains, I. *J. Symbolic Logic*, 44:481–490, 1979.

[Gur2] Y. Gurevich. Monadic second order theories. In J. Barwise and S. Feferman, editors, *Model-Theoretic Logics*, Perspectives in Mathematical Logic, chapter 14. Springer-Verlag, New York, 1985.

[GV] V. Giakoumakis and J. Vanherpe. On extended P_4-reducible and extended P_4-sparse graphs. *Theoret. Comput. Sci.*, 180:269–286, 1997.

[Har] J. Hartmanis. On sparse sets in NP-P. *Inform. Process. Lett.*, 16:55–60, 1983.

[Hoà] C. Hoàng. Doctoral thesis. McGill University, Montreal, 1985.

[JO1] B. Jamison and S. Olariu. P_4-reducible graphs a class of tree representable graphs. *Stud. Appl. Math.*, 81:79–87, 1989.

[JO2] B. Jamison and S. Olariu. A linear-time recognition algorithm for P_4-sparse graphs. *SIAM J. Comput.*, 21:381–406, 1992.

[JO3] B. Jamison and S. Olariu. A unique tree representation for P_4-sparse graphs. *Discrete Appl. Math.*, 35:115–129, 1992.

[JO4] B. Jamison and S. Olariu. A linear-time algorithm to recognize P_4-reducible graphs. *Theoret. Comput. Sci.*, 145:329–344, 1995.

[JO5] B. Jamison and S. Olariu. Linear-time optimization algorithms for P_4-sparse graphs. *Discrete Appl. Math.*, 61:155–175, 1995.

[Läu1] H. Läuchli. A decision procedure for the weak second order theory of linear order. In *Logic Colloquium* '66, pages 189–197. North-Holland, Amsterdam, 1968.

[Lau2] C. Lautemann. Logical definability of NP-optimization problems with monadic auxiliary predicates. In E. Borger, editor, *Computer Science Logic*: 6*th Workshop, CSL* '92, Lecture Notes in Computer Science, volume 702, pages 327–339, Springer-Verlag, Berlin, 1993.

[Mak] J.A. Makowsky. Translations, interpretations and reductions. Course given at ESSLLI '97, August 11–22, 1997, Aix-en-Provence, 1997.

[Pap] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[She] S. Shelah. The monadic theory of order. *Ann. of Math.*, 102:379–419, 1975.

[Spi] J. Spinrad. P_4-trees and substitution decomposition. *Discrete Appl. Math.*, 39:263–291, 1992.

[Wan] E. Wanke. k-NLC graphs and polynomial algorithms. *Discrete Appl. Math.*, 54:251–266, 1994.

Received April 13, 1998, *and in revised form June* 22, 1999, *and in final form August* 20, 1999.