HYPERGRAPHS IN MODEL CHECKING: ACYCLICITY AND HYPERTREE-WIDTH VERSUS CLIQUE-WIDTH*

GEORG GOTTLOB† AND REINHARD PICHLER‡

Abstract. The principal aim of model checking is to provide efficient decision procedures for the evaluation of certain logical formulae over finite relational structures. Graphs and hypergraphs are important examples of such structures. If no restrictions are imposed on the logical formulae and on the structures under consideration, then this problem of model checking has a very high computational complexity. Hence, several restrictions have been proposed in the literature on the logical formulae and/or on the structures under consideration, in order to guarantee the tractability of this decision problem, e.g.: acyclicity, bounded tree-width, query-width and hypertree-width in case of queries as well as bounded tree-width and clique-width in case of structures. The aim of this paper is a detailed comparison of the expressive power of these restrictions.

Key words. model checking, hypergraphs, tractability, database queries, clique-width, acyclicity, hypertree-width

AMS subject classifications. 03C13, 05C75, 68Q17, 68R10

1. Introduction.

Model checking is the problem of deciding whether a logical formula or query Q is satisfied by a finite structure $\mathcal{S}$, which is formally written as $\mathcal{S} \models Q$. Q may be a formula in first-order logic, monadic second-order logic, existential second-order logic, and so on. Model checking is a central issue in database systems [1], where $\mathcal{S}$ represents a database and the formula Q represents a database query. If Q is a closed formula, then Q is a Boolean query, otherwise $Q(\mathbf{x})$ with free variables $\mathbf{x}$ represents the query whose output consists of all tuples of domain values $\mathbf{a}$ such that $\mathcal{S} \models Q(\mathbf{a})$. Model checking is also a basic issue in the area of constraint satisfaction, which is essentially the same problem as conjunctive query evaluation [5, 33]. Finally, model checking is used in computer aided verification [13], where $\mathcal{S}$ represents a state transition graph and Q is typically a formula of modal logic describing some temporal system behaviour. The results of the present paper are, however, more relevant to the former applications, namely, conjunctive database queries and constraint satisfaction.

Without any further restriction on the form of the structures and/or the queries, these problems have a very high computational complexity. Hence, several restrictions have been proposed in the literature both for the structures and the queries, in order to make these problems tractable. In particular, the evaluation problem for acyclic queries or for queries whose tree-width, query-width or hypertree-width is bounded by some fixed constant k is tractable on arbitrary finite structures (combined complexity). On the other hand, arbitrary but fixed formulae of monadic second order logic (precisely, so-called MS_1 formulae) can be evaluated in polynomial time on graphs whose tree-width or clique-width is bounded by some fixed constant k [16, 18, 19]. In other terms, MS_1 queries have polynomial data complexity in case of bounded tree-width or bounded query-width. MS_1 extends first order logic by the possibility of quantifying over monadic relational variables representing sets of vertices. Note that only the concept of bounded tree-width has so far been applied both to the

*This work was supported by the Austrian Science Fund (FWF) Project Z29-N04. A short version of this paper was presented at ICALP'01 (cf. [30]).

†Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9, A-1040 Vienna, Austria (gottlob@dbai.tuwien.ac.at)

‡Institut für Computersprachen, Technische Universität Wien, Favoritenstraße 9, A-1040 Vienna, Austria (reini@logic.at)

queries and the structures. On the other hand, acyclicity, bounded query-width and hypertree-width have primarily been investigated as restrictions on the queries, while bounded clique-width has only been considered as a restriction on the structures. In this paper, we apply all of these restrictions both to the queries and to the structures. For reasons to be explained in §2, we consider the clique-width of a hypergraph as the clique-width of its *incidence* graph (for definitions, see §2.1). We shall thus answer the following questions:

(i) *Question 1*: How do the various notions of acyclicity and of bounded hypertree-width relate to the concept of bounded clique-width?

(ii) *Question 2*: Are Boolean conjunctive queries tractable if their clique-width is bounded by some fixed constant k?

(iii) *Question 3*: Bounded clique-width is currently the most general restriction on structures which makes the model checking problem for monadic second order formulae (so-called MS_1 formulae, see §2.2) tractable. Can the tractability barrier be pushed any further by using known generalizations of acyclicity that are more powerful than clique-width?

As for the first question, we provide an exact classification of the expressive power of the various restrictions. The result is depicted in Figure 1.1 (Definitions of all these concepts are provided in §2.1). The arrows in the figure point from the less powerful concept to the more powerful one. In particular, it is shown in this paper, that if a class $\mathcal{C}$ of hypergraphs is of bounded clique-width, then $\mathcal{C}$ is of bounded query-width (and, hence, also of bounded hypertree-width). Moreover, it is also shown e.g. that β-acyclicity is uncomparable with bounded clique-width.

In [29] it was shown that the evaluation of a class $\mathcal{C}$ of Boolean conjunctive queries is tractable (actually, it is even in LOGCFL), if $\mathcal{C}$ is of bounded hypertree-width. Putting this together with our new result that bounded clique-width implies bounded query-width (see Figure 1.1), we immediately obtain that the evaluation of a class $\mathcal{C}$ of Boolean conjunctive queries is tractable, if $\mathcal{C}$ is of bounded clique-width. Thus Question 2 is positively answered.

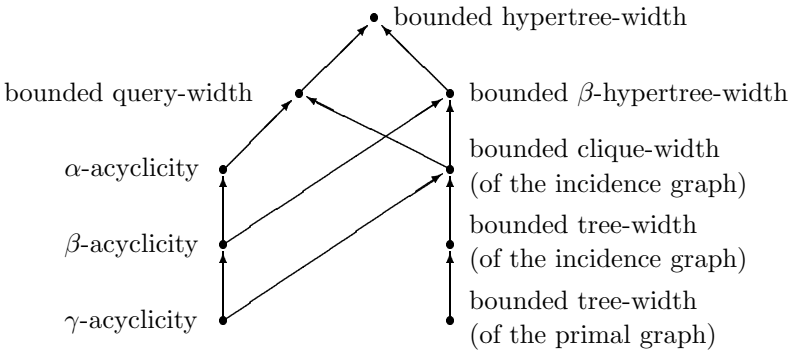


FIG. 1.1. *Expressive power of various restrictions on hypergraphs*

As for the third question, we prove that the restriction to hypergraphs of bounded query-width or bounded β-hypertree-width is not sufficient to guarantee tractability of MS_1 queries. Thus bounded clique-width remains so far the most general restriction on structures that guarantees the tractability of arbitrary fixed MS_1 queries.

While tree-width can be recognized in linear time [7], it is currently unclear whether bounded clique-width can be recognized in polynomial time. We therefore

propose *generalized tree-width (gtw)*, a cyclicity measure located between tree-width and clique-width. It will be easy to see that bounded *gtw* is recognizable in polynomial time. Moreover, we shall prove that the evaluation of MS_1 queries over structures of bounded *gtw* is indeed tractable.

This paper is structured as follows: In §2 we recall some basic notions and results. Restrictions on the form of the queries and of the structures will be considered in §3 and in §4, respectively. In §5, we show how the considerations of §4 can be used for defining generalized tree-width. Finally, in §6, we give some concluding remarks.

2. Preliminaries.

2.1. Various notions of width and acyclicity. A *graph* is a pair $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ consisting of a set $\mathcal{V}$ of vertices (or nodes) and a set $\mathcal{E}$ of edges. We only consider undirected graphs without self-loops and without multiple edges here. Hence, in particular, every edge $E \in \mathcal{E}$ is a two-element subset of $\mathcal{V}$. Two vertices are called *adjacent*, iff they are the endpoints of an edge in $\mathcal{E}$. A set $\mathcal{W} \subseteq \mathcal{V}$ is a *module* of a graph, iff the elements in $\mathcal{W}$ cannot be distinguished by the other vertices, i.e.: every vertex in $\mathcal{V} - \mathcal{W}$ is either adjacent to all vertices $W \in \mathcal{W}$ or to none. A *subgraph* is a graph $\langle \mathcal{V}', \mathcal{E}' \rangle$, s.t. $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$ hold. A subgraph is *induced*, iff $\mathcal{E}'$ is the restriction of $\mathcal{E}$ to those edges with both endpoints in $\mathcal{V}'$. In a *labelled graph*, every vertex has exactly one label. A *k-graph* is a labelled graph with k labels. Usually, these labels are taken from the set $\{1, \ldots, k\}$. A k-graph can be represented as $\langle \mathcal{V}, \mathcal{E}, \mathcal{V}_1, \ldots, \mathcal{V}_k \rangle$, where $\mathcal{V}$ is a set of vertices, $\mathcal{E}$ is a set of edges and the sets $\mathcal{V}_1, \ldots, \mathcal{V}_k$ are (possibly empty) subsets of $\mathcal{V}$ that form a partition of $\mathcal{V}$.

A *hypergraph* is a pair $\mathcal{H} = \langle \mathcal{V}, \mathcal{E} \rangle$ consisting of a set $\mathcal{V}$ of vertices and a set $\mathcal{E}$ of hyperedges. A hyperedge $H \in \mathcal{E}$ is a subset of $\mathcal{V}$. A *subhypergraph* $\langle \mathcal{V}', \mathcal{E}' \rangle$ of $\langle \mathcal{V}, \mathcal{E} \rangle$ is obtained by deleting vertices and/or hyperedges, i.e.: $\mathcal{V}' \subseteq \mathcal{V}$ and there exists a subset $\mathcal{F} \subseteq \mathcal{E}$, s.t. $\mathcal{E}' = \{H \cap \mathcal{V}' \mid H \in \mathcal{F}\}$. With every hypergraph $\mathcal{H} = \langle \mathcal{V}, \mathcal{E} \rangle$, we can associate the following two graphs: The *primal graph* (which is also called the *Gaifmann graph*) $\mathcal{P}(\mathcal{H})$ has the same vertices $\mathcal{V}$ as $\mathcal{H}$. Moreover, two vertices $V_1, V_2 \in \mathcal{V}$ are connected by an edge in $\mathcal{P}(\mathcal{H})$, iff there is a hyperedge $H \in \mathcal{E}$, s.t. both V_1 and V_2 are contained in $\mathcal{E}$. The *incidence graph* $\mathcal{I}(\mathcal{H})$ is a bipartite graph with vertices in $\mathcal{V} \cup \mathcal{E}$. Moreover, there is an edge in $\mathcal{I}(\mathcal{H})$ between two vertices $V \in \mathcal{V}$ and $H \in \mathcal{E}$, iff (in the hypergraph $\mathcal{H}$) V occurs in the hyperedge H. The incidence graph can either be considered as an unlabelled graph or as a 2-graph, with the labels $\mathcal{V}$ and $\mathcal{E}$, respectively.

In order to determine the *clique-width* of a (labelled or unlabelled) graph, we have to deal with so-called *k-expressions* t and the graphs $G(t)$ generated by such k-expressions:

(i) Let $i \in \{1, \ldots, k\}$ and let V be a vertex. Then $i(V)$ is a k-expression. The corresponding graph consists of a single vertex V whose label is i.

(ii) Let r and s be k-expressions that have no vertices in common. Then $r \oplus s$ is also a k-expression. The graph thus generated is the (disjoint) union of the graphs $G(r)$ and $G(s)$.

(iii) Let $i, j \in \{1, \ldots, k\}$ with $i \neq j$ and let r be a k-expression. Then $\eta_{i,j}(r)$ is also a k-expression. The corresponding graph is the same as $G(r)$ augmented by edges from every vertex with label i to every vertex with label j.

(iv) Let $i, j \in \{1, \ldots, k\}$ with $i \neq j$ and let r be a k-expression. Then $\rho_{i \to j}(r)$ is also a k-expression whose graph is the same as $G(r)$ except that all vertices with label i in $G(r)$ are relabelled to j.

The graph generated by a k-expression t can be either considered as a labelled graph with labels in $\{1, \ldots, k\}$ or as an unlabelled graph (by ignoring the labels assigned by t). Every subexpression s of a k-expression t generates a subgraph $G(s)$ of $G(t)$ (when considered as an unlabelled graph). The clique-width $cw(\mathcal{G})$ of a (labelled or unlabelled) graph $\mathcal{G}$ is the minimum k, s.t. there exists a k-expression that generates $\mathcal{G}$. Obviously, $cw(\mathcal{G}) \leq n$ for every graph with n vertices. It can be shown that every clique has clique-width 2, e.g. the clique with four nodes V_1, V_2, V_3, V_4 can be generated by the 2-expression $\eta_{1,2}(2(V_4) \oplus \rho_{2\to 1}(\eta_{1,2}(2(V_3) \oplus \rho_{2\to 1}(\eta_{1,2}(2(V_2) \oplus 1(V_1))))))$.

A *tree decomposition* of a graph $\langle \mathcal{V}, \mathcal{E} \rangle$ is a pair $\langle T, \lambda \rangle$, where $T = \langle N, F \rangle$ is a tree and λ is a labelling function with $\lambda(p) \subseteq \mathcal{V}$ for every node $p \in N$, s.t. the following conditions hold:

(i) $\forall V \in \mathcal{V}$, $\exists p \in N$, s.t. $V \in \lambda(p)$.

(ii) $\forall E \in \mathcal{E}$ with endpoints V_1 and V_2, $\exists p \in N$, s.t. $V_1 \in \lambda(p)$ and $V_2 \in \lambda(p)$.

(iii) $\forall V \in \mathcal{V}$, the set $\{p \in N : V \in \lambda(p)\}$ induces a connected subtree of T.

The width of a tree decomposition $\langle T, \lambda \rangle$ is $\max(\{|\lambda(p)| - 1 : p \in N\})$. The tree-width $tw(\mathcal{G})$ of a graph $\mathcal{G}$ is the minimum width over all its tree decompositions.

A *join tree* of a hypergraph $\langle \mathcal{V}, \mathcal{H} \rangle$ is a labelled tree $\langle T, \lambda \rangle$ with $T = \langle N, F \rangle$ and a labelling function λ with $\lambda(p) \in \mathcal{H}$ for every node $p \in N$. Moreover, the following conditions hold:

(i) $\forall H \in \mathcal{H}$, $\exists p \in N$, s.t. $\lambda(p) = H$.

(ii) "connectedness condition": Let $\lambda(p_1) = H_1$ and $\lambda(p_2) = H_2$ for two distinct nodes p_1 and p_2. Moreover, suppose that some vertex $V \in \mathcal{V}$ occurs in both hyperedges H_1 and H_2. Then V must also occur in all hyperedges that are used as labels on the path from p_1 to p_2.

A hypergraph is *α-acyclic*, iff it has a join tree.

In [12], a *query decomposition* of a hypergraph $\langle \mathcal{V}, \mathcal{H} \rangle$ is defined as a pair $\langle T, \lambda \rangle$ where $T = \langle N, F \rangle$ is a tree and λ is a labelling function with $\lambda(p) \subseteq (\mathcal{V} \cup \mathcal{H})$ for every $p \in N$ and

(i) $\forall H \in \mathcal{H}$, $\exists p \in N$, s.t. $H \subseteq \{V \,|\, V \in \lambda(p) \cap \mathcal{V}$ or $\exists H' \in \lambda(p) \cap \mathcal{H}$ with $V \in H'\}$.

(ii) "connectedness condition": $\forall V \in \mathcal{V}$, the set $\{p \in N : V \in \lambda(p)\} \cup \{q \in N : \exists H \in \mathcal{H}$, s.t. $H \in \lambda(q)$ and V occurs in the hyperedge $H\}$ induces a connected subtree of T.[1]

The width of a query decomposition $\langle T, \lambda \rangle$ is $\max(\{|\lambda(p)| : p \in N\})$. The query-width $qw(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is the minimum width over all its query decompositions. From [12] we know that a hypergraph $\mathcal{H}$ is α-acyclic, iff $qw(\mathcal{H}) = 1$ holds.

In [29], a *hypertree decomposition* of a hypergraph $\langle \mathcal{V}, \mathcal{H} \rangle$ is defined as a triple $\langle T, \chi, \lambda \rangle$ where $T = \langle N, F \rangle$ is a tree and χ and λ are labelling functions with $\chi(p) \subseteq \mathcal{V}$ and $\lambda(p) \subseteq \mathcal{H}$ for every $p \in N$. Moreover, the following conditions hold:

(i) $\forall H \in \mathcal{H}$, $\exists p \in N$, s.t. $H \subseteq \chi(p)$, i.e.: "p covers H".

(ii) "connectedness condition": $\forall V \in \mathcal{V}$, the set $\{p \in N : V \in \chi(p)\}$ induces a connected subtree of T.

(iii) $\forall p \in N$, $\chi(p)$ contains only vertices that actually occur in at least one hyperedge of $\lambda(p)$.

(iv) For every $p \in N$, if a vertex V occurs in some hyperedge $H \in \lambda(p)$ and if V is contained in $\chi(q)$ for some node q in the subtree below p, then V must also be

[1] Note that, in the original definition in [12], it is also required that $\forall H \in \mathcal{H}$, the set $\{p \in N : H \in \lambda(p)\}$ is a connected subtree of T. However, this restriction is of no use as far as the tractability of the evaluation of queries is concerned. We have therefore omitted this condition here.

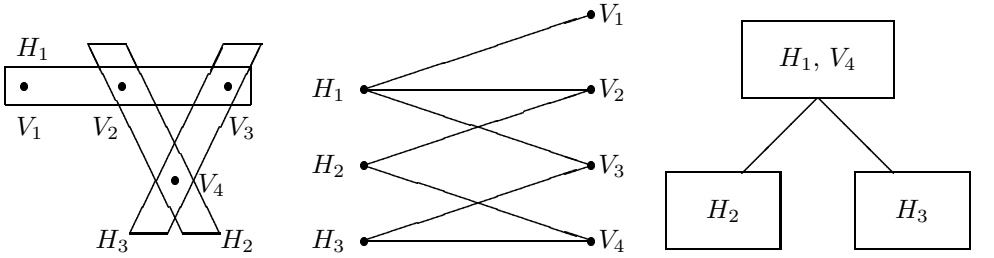
contained in $\chi(p)$.

The width of a hypertree decomposition $\langle T, \chi, \lambda \rangle$ is $\max(\{|\lambda(p)| : p \in N\})$. The hypertree-width $hw(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is the minimum width over all its hypertree decompositions.

A conjunctive query Q is a first-order formula in prenex form whose only connectives are $\exists$ and $\wedge$. With every conjunctive query, we can associate a hypergraph $\mathcal{H}$, whose vertices $V_1, \ldots, V_n$ correspond to the variables $x_1, \ldots, x_n$ occurring in Q. Moreover, for every atom A with variables $Var(A) = \{x_{i_1}, \ldots, x_{i_\alpha}\}$, there is a hyperedge $H = \{V_{i_1}, \ldots, V_{i_\alpha}\}$ in the hypergraph and vice versa. Then the notions of hypertree-width, query-width and acyclicity carry over in a natural way from hypergraphs to conjunctive queries. Likewise, the incidence graph or the primal graph of a conjunctive query Q is simply the corresponding graph of the associated hypergraph $\mathcal{H}$. Actually, the clique-width or tree-width of a hypergraph $\mathcal{H}$ can be either defined as the corresponding width of the incidence graph or the primal graph. If not indicated otherwise, we shall assume that the clique-width and the tree-width of a hypergraph $\mathcal{H}$ refer to the incidence graph (considered as an unlabelled graph). As a justification of this choice, note that there are NP-hard classes of queries, s.t. the clique-width of their primal graphs is bounded by some fixed constant k. Consider, for example, the class $\mathcal{C}$ of conjunctive queries $CLIQUE_m$ asking whether a graph $(\mathcal{V}, \mathcal{E})$ has a clique of size m, for integers m. This class of queries is well known to be NP-hard (cf. [26], problem GT19). $CLIQUE_m$ is expressed by the formula $\exists x_1, \exists x_2, \ldots, \exists x_m : \bigwedge_{1 \leq i < j \leq m} ((x_i \neq x_j) \wedge E(x_i, x_j))$. The primal graph associated to $CLIQUE_m$ is a clique of m vertices and has clique-width $k = 2$ for any $m > 1$. Thus, *Question 2* asked in the introduction has a trivial negative answer, in case the clique-width of a query is defined based on its primal graph. The question becomes highly non-trivial instead if, as done in the present paper, we define the clique-width of a query (or of a hypergraph) on the basis of its incidence graph.

Clique-width and tree-width are *hereditary* properties in that $cw(\mathcal{G}') \leq cw(\mathcal{G})$ and $tw(\mathcal{G}') \leq tw(\mathcal{G})$ hold for every induced subgraph $\mathcal{G}'$ of a graph $\mathcal{G}$. Moreover, any subhypergraph $\mathcal{H}'$ of a hypergraph $\mathcal{H}$ gives rise to an induced subgraph $\mathcal{I}(\mathcal{H}')$ of the incidence graph $\mathcal{I}(\mathcal{H})$. Hence, if $\mathcal{H}'$ is an arbitrary subhypergraph of $\mathcal{H}$, then $cw(\mathcal{H}') \leq cw(\mathcal{H})$ and $tw(\mathcal{H}') \leq tw(\mathcal{H})$ hold, where cw and tw are defined via the incidence graph of a hypergraph. In contrast, α-acyclicity, query-width and hypertree-width do not share this property, e.g.: a hypergraph $\mathcal{H}$ can be α-acyclic even though some subhypergraph $\mathcal{H}'$ is not. Likewise, $\mathcal{H}$ can have a subhypergraph $\mathcal{H}'$, s.t. $qw(\mathcal{H}) < qw(\mathcal{H}')$ or $hw(\mathcal{H}) < hw(\mathcal{H}')$ hold. The notions of *β-acyclicity* and *β-hypertree-width* can be regarded as the hereditary counterparts of α-acyclicity and hypertree-width: In [24], a hypergraph $\mathcal{H}$ is defined to be β-acyclic, iff every (not necessarily induced) subhypergraph $\mathcal{H}'$ of $\mathcal{H}$ is α-acyclic. Analogously, we can define the β-hypertree-width of $\mathcal{H}$ as the $\max(\{hw(\mathcal{H}') : \mathcal{H}' \text{ is a subhypergraph of } \mathcal{H}\})$. In [24], another notion of acyclicity is presented, namely γ-acyclicity. Any hypergraph, that is γ-acyclic, is also β-acyclic. An algorithmic definition of γ-acyclicity will be given in §4.1.

In this work, we shall compare the expressive power of the above defined notions of width. Let $x, y \in \{\text{tree, clique, query, hypertree, } \beta\text{-hypertree}\}$. Moreover, let $\mathcal{C}$ be a class of hypergraphs (or graphs). We say that $\mathcal{C}$ is of *bounded x-width*, if there exists some constant k, s.t. every hypergraph (or graph, respectively) in $\mathcal{C}$ has x-width less than or equal to k. Moreover, we say that *bounded x-width implies bounded y-width*, if every class $\mathcal{C}$ of hypergraphs (or graphs) that is of bounded x-width is also of bounded y-width.



a) hypergraph $\mathcal{H}$ b) incidence graph $\mathcal{I}(\mathcal{H})$ c) query decomposition QD

FIG. 2.1. *$\mathcal{H}$, $\mathcal{I}(\mathcal{H})$ and QD*

We conclude this section with an example, that should help to illustrate some of the main concepts used in this paper:

Example. Consider the *conjunctive query* $Q = A(x_1, x_2, x_3) \wedge B(x_2, x_4) \wedge C(x_3, x_4)$ consisting of 3 atoms and 4 variables. Consequently, the corresponding *hypergraph* $\mathcal{H} = \langle \mathcal{V}, \mathcal{E} \rangle$ is made up of 4 vertices $\mathcal{V} = \{V_1, V_2, V_3, V_4\}$ and 3 hyperedges $\mathcal{E} = \{H_1, H_2, H_3\}$ with $H_1 = \{V_1, V_2, V_3\}$, $H_2 = \{V_2, V_4\}$ and $H_3 = \{V_3, V_4\}$. Then the *incidence graph* $\mathcal{I}(\mathcal{H})$ is the bipartite graph with vertices in $\{V_1, V_2, V_3, V_4, H_1, H_2, H_3\}$, s.t. two nodes V_i and H_j are adjacent in $\mathcal{I}(\mathcal{H})$, iff, in the hypergraph $\mathcal{H}$, V_i occurs in the hyperedge H_j.

The hypergraph $\mathcal{H}$ and the incidence graph $\mathcal{I}(\mathcal{H})$ are displayed in Figure 2.1. Moreover, also a *query decomposition* QD of $\mathcal{H}$ is shown there. Note that QD has width 2. Due to the cycle $\{V_2, V_3\}$, $\{V_3, V_4\}$, $\{V_4, V_2\}$, the hypergraph $\mathcal{H}$ is not acyclic and, therefore, $\mathcal{H}$ cannot have a query decomposition of width 1. Hence, we have in fact $qw(\mathcal{H}) = 2$. Finally, a k-expression t (with $k = 5$) generating the graph $\mathcal{I}(\mathcal{H})$ can be obtained via the following algorithm:

Initialization (nodes H_i): First we introduce all nodes H_1, H_2 and H_3 with pairwise distinct labels, i.e.: we set $s_0 := 1(H_1) \oplus 2(H_2) \oplus 3(H_3)$.

Iteration (nodes V_j): In a loop over all nodes $V_1, \ldots, V_4$, we carry out the following steps: Introduce the node V_j with label 4, draw all required edges between V_j and the H_i's and relabel V_j to 5, i.e.:

$s_1 := \rho_{4\to5}(\eta_{4,1}(4(V_1) \oplus s_0))$ $\quad$ $s_2 := \rho_{4\to5}(\eta_{4,1}(\eta_{4,2}(4(V_2) \oplus s_1)))$

$s_3 := \rho_{4\to5}(\eta_{4,1}(\eta_{4,3}(4(V_3) \oplus s_2)))$ $\quad$ $s_4 := \rho_{4\to5}(\eta_{4,2}(\eta_{4,3}(4(V_4) \oplus s_3)))$

Then s_4 is the desired k-expression that generates $\mathcal{I}(\mathcal{H})$. Note that the above algorithm is applicable to any bipartite graph. Hence, in any bipartite graph $\mathcal{B}$ with nodes in $\mathcal{N}_1 \cup \mathcal{N}_2$, the condition $cw(\mathcal{B}) \leq \min(|\mathcal{N}_1|, |\mathcal{N}_2|) + 2$ is fulfilled. □

2.2. Tractability via bounded width or acyclicity. In model checking, one is interested in the (efficient) evaluation of certain logical formulae (= "queries") over finite relational structures. To this end, the various notions of width and acyclicity recalled in the previous section have been explored in two principal ways: They have been either used to restrict the relational structures or the queries. Restrictions on the structures in terms of tree-width and clique-width have been investigated in the area of graph grammars and graph algorithms. On the other hand, restrictions imposed on the queries like the various forms of acyclicity as well as bounded query-width and hypertree-width have been mainly analysed in database theory and constraint satisfaction. Note that, so far, only tree-width is common ground for both directions of research.

In this paper, we shall first deal with restrictions on the form of the queries. Recall that the evaluation of arbitrary first-order queries is PSPACE-complete (cf. [34, 35]). Actually, even if we restrict the form of first-order queries to conjunctive queries (where only conjunctions and existential quantification are allowed), then the query evaluation is still NP-complete (see [11]). If conjunctive queries are further restricted to α-acyclic conjunctive queries, then this problem becomes tractable (cf. [36]). However, acyclicity is a very severe restriction. Hence, in recent years, several attempts to deal with "almost acyclic queries" have been made. In particular, several notions of width have been introduced, in order to extend the class of tractable conjunctive queries, namely tree-width, query-width and hypertree-width (cf. [12, 25, 29, 31, 33]). In [28] and [29], it has been shown that, for some fixed k, the class of conjunctive queries with hypertree-width $\leq k$ properly contains the classes where the tree-width (of the incidence graph or of the primal graph) or the query-width, respectively, is bounded by k. Moreover, the concept of hypertree-width is a generalization of α-acyclicity in that a conjunctive query is acyclic, iff it has hypertree-width 1.

In [15], the complexity of testing certain graph properties is investigated. In terms of model checking, this corresponds to evaluating a fixed query over finite graphs. If the queries are (arbitrary but fixed) first-order formulae, then this problem is tractable for all finite graphs without any further restrictions. However, the expressive power of first-order logic is comparatively weak. Hence, attempts were made to investigate larger classes of queries. In fact, many interesting graph properties like 3-colourability, Hamiltonian circuit, partition into triangle, ... (cf. [15, 26]) are expressible as *monadic second order* queries. Note that there are basically two ways of representing a graph by a logical structure, namely: either the domain consists both of vertices and edges or the domain consists of vertices only. In the former case, quantified variables of a monadic second order formula may refer to edges or vertices whereas in the latter case, only quantification over vertices is allowed. Formulae in the former case are referred to as MS_1 formulae while formulae in the latter case are called MS_2 formulae. It has been shown that MS_2 formulae can be evaluated in polynomial time (in fact, even linear time suffices) over a class $\mathcal{C}$ of graphs, if $\mathcal{C}$ is of bounded tree-width. The restriction to bounded clique-width has proved to allow for a much larger class of structures than bounded tree-width. In particular, bounded tree-width implies bounded clique-width (cf. [19]), while the converse is in general not true, e.g., the class of cliques is of bounded clique-width (where the bound is simply 2) but of unbounded tree-width. It has been shown that the evaluation of fixed MS_1 formulae over a class $\mathcal{C}$ of graphs is tractable, if $\mathcal{C}$ is of bounded clique-width (cf. [15, 16, 17]).

3. Restricting the Form of the Queries. In this section we consider the case of conjunctive queries over arbitrary relational structures, where the queries are subjected to restrictions that guarantee tractability. It has already been recalled in the previous section that bounded hypertree-width is the most powerful concept studied so far. We shall now show that bounded clique-width does not allow for a bigger class of conjunctive queries than bounded hypertree-width. More precisely, in the proof of Theorem 3.1, we shall provide an algorithm that, when given a k-expression for (the incidence graph of) some hypergraph $\mathcal{H}$, constructs a query decomposition of $\mathcal{H}$ whose width is $\leq k$.

It is convenient to introduce some additional notation first. In the proof of Theorem 3.1, we are going to deal with three kinds of graphs or hypergraphs, respectively, i.e.: a hypergraph $\mathcal{H}'$, the incidence graph $\mathcal{I}'$ of $\mathcal{H}'$ and the query decomposition QD' of $\mathcal{H}'$. We shall therefore speak about H-vertices, I-vertices and Q-vertices (or,

equivalently, H-nodes, I-nodes and Q-nodes), when referring to the vertices in $\mathcal{H}'$, $\mathcal{I}'$ or QD', respectively. Likewise, we shall encounter two kinds of labels, namely the labels assigned by the k-expression t' and the labels of the Q-nodes in the query decomposition QD'. In order to avoid confusion, we shall refer to these labels as t-labels (for the labels of the I-vertices according to the k-expression t') and Q-labels (in case of the Q-nodes), respectively. Strictly speaking, t' assigns t-labels to the nodes in $\mathcal{I}'$ (i.e.: the I-nodes). However, every I-node uniquely corresponds either to a hyperedge or to an H-vertex in $\mathcal{H}'$. Hence, as an abbreviation, we shall speak about the "t-label of a hyperedge" or the "t-label of an H-vertex", when we refer to the t-label of the I-vertex corresponding to this hyperedge or H-vertex, respectively. We shall say that a Q-node q of the query decomposition QD' "*covers*" an H-vertex V, iff either V is contained in the Q-label $\lambda'(q)$ or $\lambda'(q)$ contains some hyperedge H, s.t. V is an H-vertex occurring in this hyperedge H. Finally, for every $\ell \in \{1, \ldots, k\}$, we define the set $\mathcal{Q}_\ell(QD')$ of Q-nodes as $\mathcal{Q}_\ell(QD') = \{p : p \text{ is a } Q\text{-node in } QD' \text{ and the } Q\text{-label of } p \text{ contains a hyperedge or an } H\text{-vertex with } t\text{-label } \ell\}$.

THEOREM 3.1 (query-width is bounded by clique-width). *Let $\mathcal{H}$ be an arbitrary hypergraph with incidence graph $\mathcal{I}(\mathcal{H})$. Then $qw(\mathcal{H}) \leq cw(\mathcal{I}(\mathcal{H}))$ holds.*

Proof. Let $\mathcal{H}$ be a hypergraph with incidence graph $\mathcal{I}$ and let t be a k-expression that generates $\mathcal{I}$. Note that every subexpression t' of t generates a subgraph $\mathcal{I}'$ of the incidence graph $\mathcal{I}$. Moreover, every such $\mathcal{I}'$ uniquely defines a hypergraph $\mathcal{H}'$. Then we construct a query decomposition QD' of $\mathcal{H}'$ inductively on the structure of t' with the following properties:

1. *QD' is a query decomposition of $\mathcal{H}'$ of width $\leq k$.*
2. *For every $\ell \in \{1, \ldots, k\}$, the above defined set $\mathcal{Q}_\ell(QD')$ of Q-vertices, if not empty, forms a connected subtree of QD', s.t. the root of this subtree coincides with the root of QD' itself. Moreover, if the incidence graph $\mathcal{I}'$ actually contains an I-node with t-label ℓ, then the Q-label $\lambda(r)$ of the root r of QD' also contains at least one I-node with t-label ℓ.*
3. *Suppose that p is a Q-node in QD', V is an H-vertex with t-label ℓ and p covers V. Moreover, let the Q-node q be the parent of p in QD'. Then either q also covers V or the Q-label $\lambda'(q)$ contains some I-node N whose t-label is ℓ.*

For the construction of QD', we consider each of the four basic operations of a k-expression separately.

Introduction of a new vertex: Let $t' = i(N)$ for some node N in $\mathcal{I}$, i.e.: N either corresponds to a hyperedge or to an H-vertex in $\mathcal{H}'$. In either case, the corresponding query decomposition QD' consists of a single node r whose Q-label is the singleton $\{N\}$. Thus, QD' trivially fulfills the above Conditions 1 through 3.

Disjoint union: Let $t' = s_1 \oplus s_2$. Moreover, let $\mathcal{J}_1$ and $\mathcal{J}_2$ be the (disjoint) subgraphs of $\mathcal{I}$ defined by s_1 and s_2, respectively, and let $\mathcal{H}_1$ and $\mathcal{H}_2$ be the corresponding hypergraphs. By the induction hypothesis, there exist query decompositions $QD_1 = \langle T_1, \lambda_1 \rangle$ and $QD_2 = \langle T_2, \lambda_2 \rangle$ of $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, for which the above three conditions hold. Let r_1 and r_2 denote the root nodes of T_1 and T_2, respectively. Then we construct the new query decomposition $QD' = \langle T', \lambda' \rangle$ in the following way: T' has a new root node r and the subtrees T_1 and T_2, s.t. r_1 and r_2 are the child nodes of r. As for the labelling function λ', the Q-labels of the Q-nodes in QD_1 and QD_2 are left unchanged. The Q-label $\lambda'(r)$ of the new root r is defined from the Q-labels of r_1 and r_2 as follows: Let $\mathcal{R} = \lambda_1(r_1) \cup \lambda_2(r_2)$. By assumption, the H-vertices and hyperedges in $\mathcal{R}$ are assigned at most k different t-labels by t'. Then we construct the Q-label $\lambda'(r)$ by selecting one representative from $\mathcal{R}$ for each t-label according to

t'. Of course, there can be at most k such representatives. We shall now show that the Conditions 1 through 3 hold for QD':

(i) *Condition 1*: By the above construction of λ' and by the induction hypothesis, the width of QD' is clearly $\leq k$. It remains to prove that QD' is indeed a query decomposition of $\mathcal{H}'$. By the induction hypothesis, every hyperedge of $\mathcal{H}_1$ occurs in some Q-label of QD_1 and every hyperedge of $\mathcal{H}_2$ occurs in some Q-label of QD_2 and, thus, also in some Q-label of QD'. Moreover, the connectedness condition follows from the induction hypothesis and the fact that $\mathcal{I}'$ is obtained as the *disjoint* union of $\mathcal{J}_1$ and $\mathcal{J}_2$. Hence, in particular, the hypergraphs $\mathcal{H}_1$ and $\mathcal{H}_2$ have no H-vertices in common.

(ii) *Condition 2*: Let $\ell \in \{1, \ldots, k\}$. By the induction hypothesis, $\mathcal{Q}_\ell(QD_1)$ and $\mathcal{Q}_\ell(QD_2)$ form connected subtrees of the query decompositions QD_1 and QD_2, respectively. Moreover, the roots of these subtrees coincide with the roots of QD_1 and QD_2, respectively, and if $\mathcal{J}_1$ or $\mathcal{J}_2$ contains an I-node with label ℓ, then $\lambda_1(r_1)$ or $\lambda_1(r_2)$, respectively, indeed contains an I-node with t-label ℓ. By our construction, $\mathcal{Q}_\ell(QD')$ is obtained as follows:

$$\mathcal{Q}_\ell(QD') = \begin{cases} \mathcal{Q}_\ell(QD_1) \cup & \\ \quad \mathcal{Q}_\ell(QD_2) \cup \{r\} & \text{if } s_1 \text{ and } s_2 \text{ contain an } I\text{-node with } t\text{-label } \ell \\ \mathcal{Q}_\ell(QD_1) \cup \{r\} & \text{if only } s_1 \text{ contains an } I\text{-node with } t\text{-label } \ell \\ \mathcal{Q}_\ell(QD_2) \cup \{r\} & \text{if only } s_2 \text{ contains an } I\text{-node with } t\text{-label } \ell \\ \emptyset & \text{otherwise} \end{cases}$$

In all of these four cases, Condition 2 clearly holds.

(iii) *Condition 3*: Let p be a Q-node in QD' and let q be the parent of p in QD'. In particular, p is not the new root node r in QD' and, therefore, p already existed before, say in QD_1. Now suppose that V is an H-vertex with t-label ℓ, s.t. p covers V in QD'. If p is the root r_1 of QD_1, then q is the new root r in QD', which (by Condition 2) contains some I-node N with t-label ℓ. On the other hand, if $p \neq r_1$, then also q is a Q-node of QD_1 and Condition 3 holds by the induction hypothesis.

Introduction of edges: Let $t' = \eta_{i,j}(s_1)$. Moreover, let $\mathcal{J}_1$ be the subgraph of $\mathcal{I}$ defined by s_1 and let $\mathcal{H}_1$ be the corresponding hypergraph. Note that the $\eta_{i,j}$ operation may have added some new edges in the incidence graph and, therefore, a hyperedge H in $\mathcal{H}'$ will, in general, contain more H-vertices than if we consider H as a hyperedge in $\mathcal{H}_1$. In fact, we may assume w.l.o.g., that the application of $\eta_{i,j}$ to s_1 indeed creates a new edge in $\mathcal{I}'$, which did not exist in $\mathcal{J}_1$. Otherwise, the hypergraphs $\mathcal{H}'$ and $\mathcal{H}_1$ would be identical and the query decomposition QD_1 of $\mathcal{H}_1$ would also be the desired query decomposition of $\mathcal{H}'$.

In order to distinguish between hyperedges in $\mathcal{H}'$ and $\mathcal{H}_1$, we shall write H and H^-, respectively, i.e.: by H^-, we denote a hyperedge in $\mathcal{H}_1$ and by H we denote the corresponding hyperedge in $\mathcal{H}'$. Of course, we have $H^- \subseteq H$ but, in general, $H^- = H$ is not true.

By the induction hypothesis, there exists a query decomposition $QD_1 = \langle T_1, \lambda_1 \rangle$ of $\mathcal{H}_1$ for which the above three conditions hold. Then we construct the query decomposition $QD' = \langle T_1, \lambda' \rangle$ of $\mathcal{H}'$ in such a way, that the tree T_1 is left unchanged. As for the labelling function λ' of QD', recall that $\mathcal{I}'$ is a bipartite graph. Moreover, by assumption, $\eta_{i,j}$ creates at least one new edge in $\mathcal{I}'$. Hence, we may assume w.l.o.g., that i is the t-label of H-vertices only and j is the t-label of hyperedges only. By Condition 2 of the induction hypothesis, $\mathcal{Q}_j(QD_1)$ is a connected subtree of QD_1 whose root coincides with the root r_1 of QD_1. Thus, the Q-label $\lambda_1(r_1)$ contains a

hyperedge G^- with t-label j. Then λ' is defined as follows: The Q-label of the root r_1 is left unchanged, i.e., $\lambda'(r_1) := \lambda_1(r_1)$. Likewise, if the Q-label of some Q-node $p \neq r_1$ does not contain an H-vertex with label i, then we do not alter this Q-label. In the Q-label of all other Q-nodes, we replace the H-vertices with t-label i by the hyperedge G.[2]

Now it can be easily checked, that *Condition 2* holds for QD', i.e.: by our construction of the labelling function λ', the sets $\mathcal{Q}_\ell(QD')$ are obtained from $\mathcal{Q}_\ell(QD_1)$ for $\ell \in \{1, \ldots, k\}$ in the following way:

$$\mathcal{Q}_\ell(QD') = \begin{cases} \{r_1\} & \text{if } \ell = i \\ \mathcal{Q}_i(QD_1) \cup \mathcal{Q}_j(QD_1) & \text{if } \ell = j \\ \mathcal{Q}_\ell(QD_1) & \text{otherwise} \end{cases}$$

In all of these three cases, Condition 2 obviously holds.

For the proof of *Condition 3*, let p be a Q-node in QD' and let q be the parent of p in QD'. Moreover, let V be an H-vertex with t-label ℓ, s.t. p covers V. Then we have to show that either q also covers V or the Q-label $\lambda'(q)$ contains some I-node N whose t-label is ℓ. Of course, p and q are also Q-nodes in QD_1. Then we distinguish the following cases:

(i) *Case 1*: Suppose that $\ell \neq i$. If V occurs in some hyperedge H of the Q-label $\lambda'(p)$, s.t. the corresponding hyperedge H^- already occurred in the Q-label $\lambda_1(p)$ in QD_1, then Condition 3 follows immediately from the induction hypothesis. Likewise, if V itself is contained in $\lambda'(p)$, then V was already contained in $\lambda_1(p)$ and we may again conclude by the induction hypothesis that Condition 3 still holds for QD'. On the other hand, suppose that V occurs in some hyperedge H from the Q-label $\lambda'(p)$, s.t. the corresponding hyperedge H^- did not occur in the Q-label $\lambda_1(p)$ in QD_1. In other words, $H = G$ and G was introduced into $\lambda'(p)$, when we replaced the H-vertices with label i by the hyperedge G in the Q-labels of all Q-nodes except for the root node r_1 of QD'. If $q = r_1$, then Condition 3 clearly holds, since $\lambda'(q)$ also contains G. Otherwise, by Condition 2 of the induction hypothesis, also the parent node q of p in QD_1 contains some I-node N with t-label i in its Q-label $\lambda_1(q)$. By our construction, N is replaced by the hyperedge G in $\lambda'(q)$. Hence, V is also covered by q in QD'.

(ii) *Case 2*: Suppose that $\ell = i$. W.l.o.g., we may assume that $q \neq r_1$, since otherwise q contains some I-node with t-label i by Condition 2 and we are done. By our construction, V (with t-label i) itself cannot occur in the Q-label of the Q-node $p \neq r_1$ in QD'. Hence, there exists some hyperedge H in $\mathcal{H}'$, s.t. V occurs in H and $H \in \lambda'(p)$. We distinguish two subcases for the t-label of H:
If H has the t-label j, then, by Condition 2, also $\lambda'(q)$ contains some hyperedge H' with t-label j. Moreover, by the application of the $\eta_{i,j}$ operation in the incidence graph $\mathcal{I}'$, the hyperedge H' in $\mathcal{H}'$ indeed contains all H-vertices with t-label i. Thus, q clearly covers V in QD'.
Finally, suppose that H has a t-label different from j. Then H^- already existed in $\lambda_1(p)$ and the H-vertex V already occurred in the hyperedge H^- of the hypergraph $\mathcal{H}_1$. Hence, we may apply Condition 3 of the induction hypothesis, i.e.: either V

[2] Actually, this formulation is slightly inaccurate. Recall that, in general, a hyperedge H in $\mathcal{H}'$ may have some additional vertices, which are not contained in the corresponding hyperedge H^- in $\mathcal{H}_1$, e.g.: Strictly speaking, we would have to write $\lambda'(r_1) := \{H : H^- \in \lambda_1(r_1)\} \cup \{V \in \lambda_1(r_1)\}$ rather than $\lambda'(r_1) := \lambda_1(r_1)$. However, the meaning of the latter formulation is clear. Moreover, as far as the incidence graph is concerned, H^- and H refer to exactly the same I-node, anyway.

occurs in some hyperedge F^- in $\mathcal{H}_1$, s.t. $F^- \in \lambda_1(q)$ or $\lambda_1(q)$ contains some I-node N whose t-label is i. In the former case, $F \in \lambda'(q)$ and, therefore, q also covers V in QD'. In the latter case, N is replaced by G in $\lambda'(q)$ and, again, q covers V in QD'.

It remains to prove that also *Condition 1* still holds, i.e.: QD' indeed is a query decomposition of $\mathcal{H}'$ with width $\leq k$. Actually, the bound on the width follows immediately from the induction hypothesis and the fact that $|\lambda'(p)| \leq |\lambda_1(p)|$ holds for every Q-node $p \in T_1$. Moreover, for every hyperedge H in $\mathcal{H}'$, QD_1 contains a Q-node p with $H^- \in \lambda_1(p)$ and, therefore, also $H \in \lambda'(p)$ holds. The only difficult part of the proof is to show that QD' fulfills the connectedness condition. In fact, this is the only place in the whole proof of Theorem 3.1, where we actually need the Conditions 2 and 3 of our construction.

Now let V be an arbitrary H-vertex of $\mathcal{H}'$ and let ℓ denote the t-label of V. Then we have to show that the set of Q-nodes that cover V form a connected subtree in QD'. To this end we distinguish the following cases:

(i) *Case 1*: Let $\ell \neq i$. In this case, for every hyperedge H in $\mathcal{H}'$ with $V \in H$, we know that also $V \in H^-$ holds. Moreover, by the induction hypothesis, the set of Q-nodes $\mathcal{P}_1 = \{p \in T_1 : V \in \lambda_1(p)\} \cup \{q \in T_1 : \exists H^-$, s.t. H^- is a hyperedge in $\mathcal{H}_1$, $H^- \in \lambda_1(q)$ and $V \in H^-\}$ induces a connected subtree of T_1. First, suppose that V does not occur in the hyperedge G by which all H-vertices with t-label i were replaced when we constructed λ' from λ_1. In this case, the set of Q-nodes $\mathcal{P}' = \{p \in T_1 : V \in \lambda'(p)\} \cup \{q \in T_1 : \exists H$, s.t. H is a hyperedge in $\mathcal{H}'$, $H \in \lambda'(q)$ and $V \in H\}$ coincides with $\mathcal{P}_1$ and we are done. So suppose that V occurs in the hyperedge G. But then, by our construction, $\mathcal{P}' = \mathcal{Q}_i(QD_1) \cup \mathcal{P}_1$ holds, where both of the sets $\mathcal{Q}_i(QD_1)$ and $\mathcal{P}_1$ are connected subtrees of T_1 containing the root r_1 of T_1. Thus, $\mathcal{P}'$ is also connected.

(ii) *Case 2*: Suppose that $\ell = i$. It suffices to show the following fact: *Let q be an arbitrary Q-node that covers V and let $q_0, q_1, \ldots, q_m$ for some $m \geq 0$ denote the path in QD' from the root $r_1 = q_0$ of T_1 to the Q-node $q = q_m$. Then every Q-node q_α along this path covers V.*

Of course, the root $r_1 = q_0$ covers V, since $\lambda'(r_1)$ contains the hyperedge G with t-label j, which (by the $\eta_{i,j}$ operation) is adjacent to the H-vertex V in $\mathcal{I}'$. For the other Q-nodes q_α with $\alpha > 0$, we know that they can no longer contain the H-vertex V itself in their Q-label $\lambda'(q_\alpha)$. Hence, in particular, $\lambda'(q_m)$ contains some hyperedge H, s.t. the H-vertex V occurs in H. Similarly to Case 2 in the proof of Condition 2 above, we have to distinguish two subcases depending on whether the t-label of H equals j or not.

If H has the t-label j, then we know by Condition 2, that every Q-node q_α on the path from q_m to the root r_1 of QD' contains some hyperedge H' with t-label j in its Q-label $\lambda'(q_\alpha)$. Moreover, (by the $\eta_{i,j}$ operation) the hyperedge H' in $\mathcal{H}'$ contains all H-vertices with t-label i. Hence, in this case, every such Q-node q_α in QD' indeed covers V.

On the other hand, suppose that the t-label of H is different from j. Then no H-vertex is added to this hyperedge by the $\eta_{i,j}$ operation. Hence, V already occurred in the corresponding hyperedge H^- of the hypergraph $\mathcal{H}_1$ and also the Q-label $\lambda_1(q_m)$ in QD_1 already contained the hyperedge H^-. Hence, we may apply Condition 3 of the induction hypothesis, i.e.: either V occurs in some hyperedge F^- in $\mathcal{H}_1$, s.t. $F^- \in \lambda_1(q_{m-1})$ or $\lambda_1(q_{m-1})$ contains some I-node N whose t-label is i. In the former case, $F \in \lambda'(q_{m-1})$ and, therefore, q also covers V in QD'. In the latter case, N is replaced by G in $\lambda'(q_{m-1})$ and, hence, again q covers V in QD'. By an easy induction

argument, we can show that in fact every Q-node $q_{m-\beta}$ with $\beta \in \{1, \ldots, m-1\}$ covers V in QD'.

Renaming of labels: Let $t' = \rho_{i \to j}(s_1)$. Moreover, let $\mathcal{J}_1$ be the subgraph of $\mathcal{I}$ defined by s_1 and let $\mathcal{H}_1$ be the corresponding hypergraph. By the induction hypothesis, there exists a query decomposition QD_1 of $\mathcal{H}_1$ for which the above three conditions hold. We claim that then QD_1 is also the desired query decomposition QD' of $\mathcal{H}'$. Note that by the renaming of labels, no new vertices or edges are introduced in $\mathcal{I}'$. Hence, $\mathcal{H}'$ is identical to $\mathcal{H}_1$ and, therefore, QD_1 is clearly a query decomposition of $\mathcal{H}'$, i.e.: *Condition 1* holds.

The sets of Q-nodes $\mathcal{Q}_\ell(QD')$ with $\ell \in \{1, \ldots, k\}$ are obtained from the sets $\mathcal{Q}_\ell(QD_1)$ in the following way:

$$\mathcal{Q}_\ell(QD') = \begin{cases} \emptyset & \text{if } \ell = i \\ \mathcal{Q}_i(QD_1) \cup \mathcal{Q}_j(QD_1) & \text{if } \ell = j \\ \mathcal{Q}_\ell(QD_1) & \text{otherwise} \end{cases}$$

Hence, also *Condition 2* follows immediately from the induction hypothesis.

As for *Condition 3*, let p be a Q-node in QD' and let q be the parent of p in QD'. Moreover, let V be an H-vertex with t-label ℓ, s.t. p covers V. The only interesting case is that the t-label of V was changed from i to j by the $\rho_{i \to j}$ operation. But then, by the induction hypothesis, either q covers V in QD_1 or the Q-label $\lambda_1(q)$ contains some I-node N whose t-label (in s_1) is i. In the former case, q still covers V in QD' and in the latter case the Q-label $\lambda'(q)$ contains the I-node N, which now has the t-label j. Hence, in either case, Condition 3 holds. □

The converse of Theorem 3.1 is clearly not true. This is due to the fact that the clique-width is a hereditary property w.r.t. induced subgraphs (and any subhypergraph $\mathcal{H}'$ of a hypergaph $\mathcal{H}$ indeed gives rise to an induced subgraph $\mathcal{I}(\mathcal{H}')$ of the incidence graph $\mathcal{I}(\mathcal{H})$) whereas α-acyclicity, query-width and hypertree-width are not. In particular, we can take any hypergraph $\mathcal{H}'$ with high clique-width and possibly high hypertree-width and transform it into the following hypergraph $\mathcal{H}$: Let H be a new hyperedge that contains all vertices of $\mathcal{H}'$ and let $\mathcal{H}$ be the result of adding H to $\mathcal{H}'$. Then $\mathcal{H}$ is α-acyclic and, therefore, $qw(\mathcal{H}) = hw(\mathcal{H}) = 1$ holds. On the other hand, the incidence graph of $\mathcal{H}'$ is an induced subgraph of the incidence graph of $\mathcal{H}$. Hence, the clique-width of $\mathcal{H}$ is at least as high as in case of $\mathcal{H}'$.

The following example will help to illustrate the construction in the proof of Theorem 3.1.

Example. Consider the conjunctive query $A(x_1, x_2, x_3) \wedge B(x_2, x_3, x_4)$. The corresponding hypergraph $\mathcal{H}$ has 4 vertices $\{V_1, V_2, V_3, V_4\}$ and 2 hyperedges $H_1 = \{V_1, V_2, V_3\}$ and $H_2 = \{V_2, V_3, V_4\}$. The incidence graph $\mathcal{I}$ of $\mathcal{H}$ and the tree representation of a k-expression t (with $k = 3$) generating $\mathcal{I}$ are displayed in Figure 3.1.

Now let us traverse the tree representation of t bottom-up and see, what the various subexpressions of t and the corresponding query decompositions look like. Actually, we shall only discuss the subexpressions s_i along the left-most path of the tree representation of t in detail. The corresponding query decompositions QD_i are depicted in Figure 3.2.

The query decomposition QD_1 corresponding to $s_1 = 1(V_1)$ consists of a single node labelled by V_1. Likewise, from $3(H_1)$ we get a query decomposition with a single node labelled by H_1. Hence, the labelling of the new root in the query decomposition QD_2 corresponding to $s_2 = 1(V_1) \oplus 3(H_1)$ contains both V_1 and H_1 from its child nodes, since they have different labels in s_2. In the query decomposition QD_3 obtained

a) incidence graph $\mathcal{I}$: b) k-expression t:

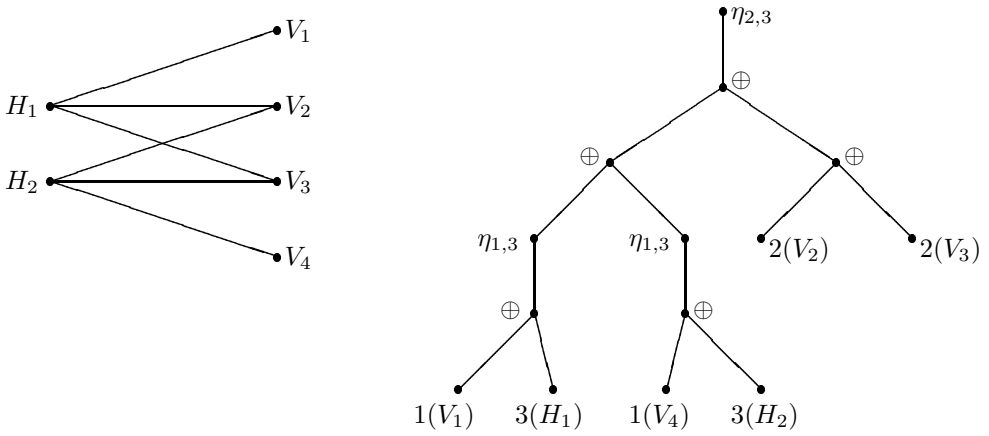


FIG. 3.1. *$\mathcal{I}$ and t*

from $s_3 = \eta_{1,3}(s_2)$, we have to replace all occurrences of V_1 outside the root of QD_2 by the hyperedge H_1. Actually, in this case, this step was not really necessary. However, when we discuss the k-expression s_6 below, it will become clear why this replacement is, in general, required.

Now consider the query decomposition QD_4 corresponding to the k-expression $s_4 = s_3 \oplus \eta_{1,3}[1(V_4) \oplus 3(H_2)]$. The subtree in QD_4 corresponding to $\eta_{1,3}[1(V_4) \oplus 3(H_2)]$ is obtained analogously to the query decomposition QD_3 corresponding to s_3. Moreover, in the root of QD_4, we are only allowed to select one representative from the sets $\{V_1, V_4\}$ and $\{H_1, H_2\}$, respectively, since V_1 and V_4 (when considered as nodes in the graph generated by s_4), on the one hand, and H_1 and H_2, on the other hand, have the same label in s_4.

No new ideas are required for the construction of the query decomposition QD_5 corresponding to the k-expression $s_5 = s_4 \oplus (2(V_2) \oplus 2(V_3))$. Note that in the query decomposition QD_6 corresponding to $s_6 = t = \eta_{2,3}(s_5)$, it is indeed necessary to replace the occurrences of V_2 and V_3 by the hyperedge H_1. In particular, QD_5 is no longer a valid query decomposition after the $\eta_{2,3}$ operation has been applied to the incidence graph. This is due to the fact that after this operation, the hyperedge H_1 contains the vertex V_3 in the corresponding hypergraph. But then the connectedness condition would be violated by QD_5, since the root covers the vertex V_3 (since this vertex is now contained in H_1) and also the right-most leaf node of QD_5 covers the vertex V_3. However, in QD_5, the node lying in between them only contains V_2. □

Recall from [29] that $qw(\mathcal{H}) \geq hw(\mathcal{H})$ holds for every hypergraph $\mathcal{H}$. Hence, by Theorem 3.1, we immediately get:

COROLLARY 3.2 (hypertree-width is bounded by clique-width). *Let $\mathcal{H}$ be an arbitrary hypergraph with incidence graph $\mathcal{I}(\mathcal{H})$. Then $hw(\mathcal{H}) \leq cw(\mathcal{I}(\mathcal{H}))$ holds.*

Moreover, by the correspondence between queries and hypergraphs, we have:

COROLLARY 3.3 (width of a conjunctive query). *Let Q be a conjunctive query with incidence graph $\mathcal{I}(Q)$. Then $hw(Q) \leq qw(Q) \leq cw(\mathcal{I}(Q))$ holds.*

The above corollary has another interesting aspect: Apart from the special case of $k \leq 3$, it is not known whether graphs with clique-width $\leq k$ can be recognized in polynomial time for fixed k (cf. [14]). In contrast, conjunctive queries (or, equivalently,

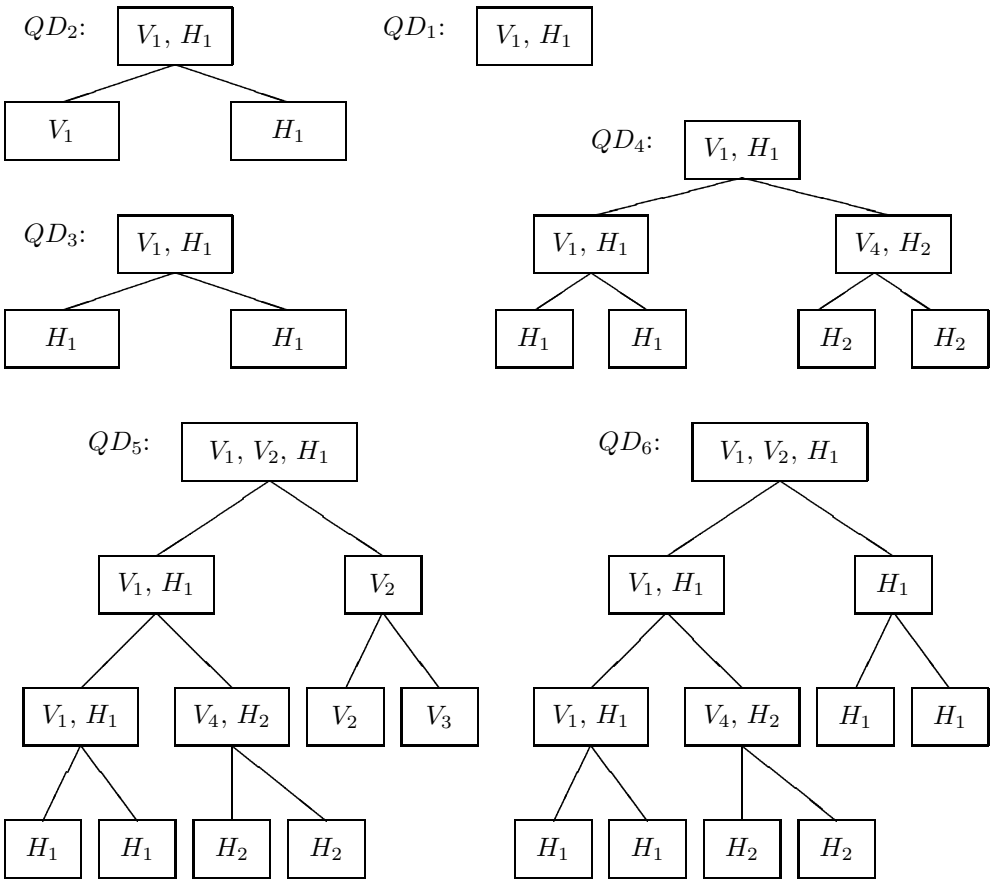


FIG. 3.2. *Query decompositions* $QD_1, \ldots, QD_6$

hypergraphs) with hypertree-width $\leq k$ actually can be recognized in polynomial time. In fact, this decision problem is highly parallelizable since it is even contained in the low complexity class LOGCFL (cf. [29]). In other words, apart from being the more general concept, bounded hypertree-width also has better properties as far as recognizing such conjunctive queries is concerned.

4. Restricting the Form of the Structures. In this section we consider monadic second-order queries over hypergraphs, where quantification is allowed over variables that stand for vertices or hyperedges. Moreover, there are two unary predicates P_V, P_H and a binary predicate edg with the following meaning: $P_V(x)$, $P_H(x)$ state that the argument x is a vertex or a hyperedge, respectively, of the hypergraph. By $edg(v, h)$ we can express that the vertex v is contained in the hyperedge h. Clearly, these formulae correspond to MS_1 formulae that are evaluated over the incidence graphs (when considered as a labelled graph with two labels) of hypergraphs. Thus, the evaluation of such formulae is tractable, if the incidence graphs under consideration are of bounded clique-width. From [16] we know that a class of labelled graphs with p labels (for fixed $p \geq 1$) is of bounded clique-width, iff the same graphs

without labels are of bounded clique-width. Hence, in the sequel, we shall ignore the two different labels of the nodes of the incidence graph, since they have no effect on the tractability of the evaluation of MS_1 formulae.

It has already been mentioned that clique-width is a hereditary property while α-acyclicity and hypertree-width are not. In case of restrictions on the queries, this does not matter. However, if we look for appropriate restrictions on the structures, then it can be easily verified that α-acyclicity or bounded hypertree-width will clearly not suffice to make the evaluation of any fixed MS_1 formula tractable. Instead, we shall consider β-acyclicity and β-hypertree-width here as well as γ-acyclicity, which is even slightly more restrictive than β-acyclicity. It will turn out that γ-acyclic hypergraphs have clique-width ≤ 3 and that the β-hypertree-width of any hypergraph is less than or equal to the clique-width (of the incidence graph). Hence, γ-acyclicity of a class of hypergraphs implies bounded clique-width (w.r.t. the incidence graph) which, in turn, implies bounded β-hypertree-width. In other words, γ-acyclicity and bounded β-hypertree-width can be considered as "lower and upper bounds", respectively, on the expressive power of the notion of bounded clique-width. However, we shall also show that bounded β-hypertree-width is not sufficient to ensure the tractability of the evaluation of an arbitrary but fixed MS_1 formula.

In the second part of this section, we shall have a brief look at the primal graph of hypergraphs.

4.1. Clique-width of the incidence graph. In [20], D'Atri and Moscarini provided an algorithm for recognizing γ-acyclic hypergraphs. (For details, see the original paper or [24]). In terms of the incidence graph of a hypergraph, we get an algorithm consisting of the following rules:

1. *deletion of isolated nodes*: If a node N in $\mathcal{I}$ has no adjacent node, then N may be deleted.
2. *deletion of "ear nodes"*: If a node N in $\mathcal{I}$ has exactly one adjacent node, then N may be deleted.
3. *contraction of two-element modules*: If two nodes N and N' in $\mathcal{I}$ are adjacent to exactly the same nodes in $\mathcal{V} - \{N, N'\}$, then one of them may be deleted.

Moreover, we assume that an edge is deleted from the incidence graph, if one of its endpoints is deleted by one of these rules. Then a hypergraph $\mathcal{H}$ is γ-acyclic, iff the exhaustive, non-deterministic application of the above rules transforms the incidence graph $\mathcal{I}(\mathcal{H})$ into the empty graph.

As far as the clique-width (of the incidence graphs) of γ-acyclic hypergraphs is concerned, several known results can be combined to get

THEOREM 4.1 (γ-acyclicity implies bounded clique-width). *Every γ-acyclic hypergraph has clique-width ≤ 3 (w.r.t. the incidence graph).*

Proof. ([8]) Let $\mathcal{H}$ be an arbitrary γ-acyclic hypergraph. We know from [2] that the incidence graph $\mathcal{I}(\mathcal{H})$ is "(6,2)-chordal". Building upon a characterization of "distance-hereditary" graphs in [4], it was shown in [21] that a graph is bipartite, (6,2)-chordal, iff it is bipartite, distance-hereditary. Finally, in [27], it is shown that any distance-hereditary graph has clique-width ≤ 3. □

When we define the notion of generalized tree-width in §5, we shall revisit the algorithm from [27] which constructs a 3-expression of any given distance-hereditary graph. In particular, this algorithm can be used to compute a 3-expression for the incidence graph of a given γ-acyclic hypergraph. The example in §5 for a modified version of this algorithm will also help to illustrate the above theorem.

In the remainder of this section, we compare the clique-width (of the incidence graph) with β-acyclicity and β-hypertree-width.

THEOREM 4.2 (clique-width of the incidence graph versus β-acyclicity). *The class of β-acyclic hypergraphs is of unbounded clique-width (w.r.t. to the incidence graph).*

Proof. Consider the sequence $(\mathcal{H}_n)_{n\geq 1}$ of hypergraphs, where $\mathcal{H}_n$ has the vertices $V = \{y_1, \ldots, y_n\} \cup \{x_{ij} : 1 \leq i < j \leq n\}$ and n hyperedges $H_1, \ldots, H_n$ with

$$H_l = \{y_l\} \cup \{x_{\alpha\beta} : \alpha < \beta \leq l\} \cup \{x_{l\gamma} : l < \gamma \leq n\} \text{ for } l \in \{1, \ldots, n\},$$

i.e.: $H_1 = \{y_1, x_{12}, x_{13}, \ldots, x_{1n}\}$, $H_2 = \{y_2, x_{12}, x_{23}, \ldots, x_{2n}\}$, $H_3 = \{y_3, x_{12}, x_{13}, x_{23}, x_{34}, \ldots, x_{3n}\}$, $\ldots$, $H_n = \{y_n, x_{12}, \ldots, x_{1n}, x_{23}, \ldots, x_{2n}, \ldots, x_{(n-1)n}\}$. The β-acyclicity of $\mathcal{H}_n$ can be shown via the following observation: Let H_i, H_j, H_k be hyperedges of $\mathcal{H}_n$ with $i < j < k$. Then the relation $H_i \cap H_j \subseteq H_k$ holds. Now let $\mathcal{H}'$ be a subhypergraph of $\mathcal{H}_n$ with vertices $V' \subseteq V$ and m hyperedges $H'_{i_1}, \ldots, H'_{i_m}$, s.t. $H_{i_1}, \ldots, H_{i_m}$ are hyperedges in $\mathcal{H}_n$ and $H'_{i_j} = H_{i_j} \cap V'$ holds for every $j \in \{1, \ldots, m\}$. W.l.o.g., let $1 \leq i_1 < i_2 < \ldots < i_m \leq n$. Then $\mathcal{H}'$ is α-acyclic, since a join tree of $\mathcal{H}'$ can be obtained by labelling the root node with H'_{i_m} and attaching $m-1$ child nodes with the labels $H'_{i_1}, \ldots, H'_{i_{(m-1)}}$ to the root.

It remains to prove that the incidence graphs $(\mathcal{I}_n)_{n\geq 1}$ of $(\mathcal{H}_n)_{n\geq 1}$ are of unbounded clique-width. In fact, we show that $cw(\mathcal{I}_n) \geq n$ holds for every $n \geq 2$. Suppose on the contrary that $\mathcal{I}_n$ is defined by some k-expression t with $k \leq n-1$. Moreover, let t' be a subexpression in t, s.t. for some $i \in \{1, \ldots, n\}$ the nodes H_i and y_i as well as at least $n-2$ nodes from the set $\{x_{1i}, \ldots, x_{(i-1)i}, x_{i(i+1)}, \ldots, x_{in}\}$ (consisting of $n-1$ nodes) have already been introduced in t' and let t' be minimal with this property, i.e.: no proper subexpression t'' of t' has this property. By the minimality of t', this subexpression t' is of the form $t' = r \oplus s$ for appropriately chosen k-expressions r and s. Then we derive a contradiction in the following way:

Fact 1: Suppose that two vertices H_α and H_β with $\alpha \neq \beta$ have been introduced by the k-expression t'. Then H_α and H_β have different labels in the graph generated by t'. This can be seen as follows: Suppose on the contrary that H_α and H_β have the same label. W.l.o.g., let $\alpha < \beta$. Then the vertices in $V_\beta = \{y_\beta\} \cup \{x_{1\beta}, \ldots, x_{(\alpha-1)\beta}, x_{(\alpha+1)\beta}, \ldots, x_{(\beta-1)\beta}, x_{\beta(\beta+1)}, \ldots, x_{\beta n}\}$ distinguish the vertices H_α and H_β. Hence, all of the edges connecting H_β with the vertices in V_β must already exist in t'. Thus, all of the vertices in $\{H_\beta\} \cup V_\beta$ must have already been introduced in one of the subexpressions r or s of t', which contradicts the minimality of t'.

Fact 2: Suppose that the vertices H_α and $x_{\beta\gamma}$ with arbitrary α, β and γ have been introduced by the k-expression t'. Note that H_α is adjacent to all of the vertices in $V_\alpha = \{y_\alpha\} \cup \{x_{1\alpha}, \ldots, x_{(\alpha-1)\alpha}, x_{\alpha(\alpha+1)}, \ldots, x_{\alpha n}\}$, while $x_{\beta\gamma}$ is not. Hence, one can show analogously to Fact 1 above that H_α and $x_{\beta\gamma}$ have different labels.

Fact 3: If the vertices H_α and y_β with arbitrary α and β have been introduced by the k-expression t', then H_α and y_β have different labels, since H_α and y_β are also distinguished by the vertices in the above set V_α.

In order to conclude the proof, we define a set $\mathcal{S}$ of n vertices of t', s.t. these vertices have pairwise distinct labels in t'.

(i) H_i is in $\mathcal{S}$.

(ii) Let X denote those vertices in $\{x_{1i}, \ldots, x_{(i-1)i}, x_{i(i+1)}, \ldots, x_{in}\}$, that already exist in the k-expression t'. By assumption, X has at least $n-2$ elements. Now we traverse X from right to left. If the label of some $x_{\alpha\beta} \in X$ does not occur in X to the right of $x_{\alpha\beta}$, then we add $x_{\alpha\beta}$ to $\mathcal{S}$. Otherwise, suppose that $x_{\gamma\delta}$ is a vertex to the right of $x_{\alpha\beta}$, s.t. $x_{\alpha\beta}$ and $x_{\gamma\delta}$ have the same label. Then we distinguish two cases: If $\alpha < i$ and $\beta = i$ hold, then H_α

distinguishes the nodes $x_{\alpha i}$ and $x_{\gamma\delta}$. Hence, H_α already exists in t' and we may add H_α to $\mathcal{S}$. Otherwise (i.e.: $\alpha = i$ and $\beta > i$ hold), H_β distinguishes the nodes $x_{i\beta}$ and $x_{\gamma\delta}$. Hence, H_β already exists in t' and we add H_β to $\mathcal{S}$.

(iii) Finally we consider y_i. Let $X \subseteq \{x_{1i}, \ldots, x_{(i-1)i}, x_{i(i+1)}, \ldots, x_{in}\}$ be defined as above. If the label of y_i does not occur in X, then we add y_i to $\mathcal{S}$. Otherwise, let $x_{\alpha\beta}$ denote the right-most vertex in X that has the same label as y_i. Then we distinguish the same cases as above: If $\alpha < i$ and $\beta = i$ hold, then H_α distinguishes the nodes $x_{\alpha i}$ and y_i. Hence, we may add H_α to $\mathcal{S}$. Otherwise (i.e.: $\alpha = i$ and $\beta > i$ hold), we add H_β to $\mathcal{S}$.

Then $\mathcal{S}$ contains n vertices. In particular, all of the vertices H_α and H_β that are added to $\mathcal{S}$ by the above construction are pairwise distinct. Moreover, by the construction of $\mathcal{S}$ and by the Facts 1 through 3 above, all of the vertices in $\mathcal{S}$ have pairwise distinct labels. □

Actually, the above lower bound on the clique-width of the incidence graphs $(\mathcal{I}_n)_{n\geq 1}$ is quite tight. This follows from the fact that the incidence graphs $(\mathcal{I}_n)_{n\geq 1}$ are bipartite graphs, where one part of the partition has n nodes (namely $\{H_1, \ldots, H_n\}$). Hence, as we have seen in the example in §2.1, $\mathcal{I}_n$ has clique-width $\leq n + 2$.

Remark. Theorem 4.2 was shown independently in [8] as follows: It was shown in [2], that β-acyclic hypergraphs have bipartite, "chordal" incidence graphs. Moreover, we know from [9], that bipartite permutation graphs are a subclass of bipartite, chordal graphs. Finally, in [10] it was shown that the clique-width of bipartite permutation graphs is unbounded. □

Now recall from Corollary 3.2 that the hypertree-width of any hypergraph $\mathcal{H}$ is less than or equal to the clique-width of the incidence graph of a hypergraph. Moreover, as has already been mentioned, the incidence graph $\mathcal{I}(\mathcal{H}')$ of any subhypergraph $\mathcal{H}'$ of $\mathcal{H}$ is an induced subgraph of $\mathcal{I}(\mathcal{H})$. Thus, $cw(\mathcal{I}(\mathcal{H}')) \leq cw(\mathcal{I}(\mathcal{H}))$ holds. We therefore immediately get:

COROLLARY 4.3 (β-hypertree-width is bounded by clique-width). *Let $\mathcal{H}$ be a hypergraph with clique-width $= k$ (w.r.t. the incidence graph). Then $\mathcal{H}$ has β-hypertree-width $\leq k$.*

It has already been recalled from [29] that α-acyclic hypergraphs have hypertree-width $= 1$. Consequently, the β-acyclic hypergraphs have β-hypertree-width $= 1$. Thus, by Theorem 4.2 and Corollary 4.3, we know that bounded clique-width implies bounded β-hypertree-width while the converse is not true. Now the question naturally arises as to whether bounded β-hypertree-width of the structures under consideration suffices to guarantee the tractability of the evaluation of any MS_1 formula. Unfortunately, the answer given in Theorem 4.5 below is negative. Thus, bounded clique-width remains the concept with the highest expressive power known so far, s.t. MS_1 queries are still tractable (cf. Figure 1.1). The proof of this result will be based on the following lemma:

LEMMA 4.4 (β-hypertree-width of hypergraphs with big hyperedges). *Let $\mathcal{H} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a hypergraph, where every hyperedge $H \in \mathcal{E}$ has at least $|\mathcal{V}| - 2$ vertices. Then the β-hypertree-width of $\mathcal{H}$ is ≤ 3.*

Proof. Let $\mathcal{H}' = \langle \mathcal{V}', \mathcal{E}' \rangle$ be an arbitrary subhypergraph of $\mathcal{H}$, i.e.: $\mathcal{H}'$ is obtained from $\mathcal{H}$ by deleting some hyperedges and/or vertices. Note that then also $\mathcal{H}'$ contains only "big" hyperedges, i.e.: every hyperedge $H' \in \mathcal{E}'$ has at least $|\mathcal{V}'| - 2$ vertices.

We have to show that $\mathcal{H}'$ has a *hypertree decomposition* of width ≤ 3. In fact, we show that $\mathcal{H}'$ even has a *query decomposition* of width ≤ 3. Let $H \in \mathcal{E}'$ be an arbitrary hyperedge. Then $|\mathcal{V}' - H| \leq 2$ holds. Let $V_1, V_2 \in \mathcal{V}'$, s.t. $H \cup \{V_1, V_2\} = \mathcal{V}'$. Now we can construct a query decomposition QD of width ≤ 3 for $\mathcal{H}'$ as follows: Let the root r of QD be labelled with $\lambda(r) = \{H, V_1, V_2\}$. Moreover, for every $H' \in \mathcal{H}' - \{H\}$, we attach one child node p to r with label $\lambda(p) = \{H'\}$. □

THEOREM 4.5 (MS_1 queries and bounded β-hypertree-width). *The evaluation of an arbitrary fixed MS_1 query over a class $\mathcal{C}$ of hypergraphs is, in general, not tractable, even if $\mathcal{C}$ is of bounded β-hypertree-width.*

Proof. Let $\mathcal{G} = \langle V, E\rangle$ be an arbitrary graph and let $\mathcal{H} = \langle V, H\rangle$ be a hypergraph, where the set H of hyperedges is defined as follows: $H = \{V - \{x, y\} : \{x, y\}$ is an edge in $E\}$, i.e.: every edge e of $\mathcal{G}$ is encoded by a hyperedge which contains all vertices of V except for the endpoints of e. Then every hyperedge of $\mathcal{H}$ has $|V| - 2$ vertices. Thus, by Lemma 4.4 above, this hypergraph $\mathcal{H}$ has β-hypertree-width at most 3. Moreover, the well known NP-complete problem of graph-3-colourability can be expressed as an MS_1 query on the incidence graph of $\mathcal{H}$ (when considered as a labelled graph with two distinct labels for the nodes corresponding to vertices and hyperedges in $\mathcal{H}$, respectively. The unary predicates P_V and P_H refer to these labels.) in the following way:

$$(\exists C_1)(\exists C_2)(\exists C_3)\ \text{“}C_1, C_2 \text{ and } C_3 \text{ provide a partition of } V\text{”} \wedge$$
$$(\forall x)(\forall y)[\,(P_V(x) \wedge P_V(y) \wedge (\exists h)(P_H(h) \wedge \neg edg(x,h) \wedge \neg edg(y,h)))$$
$$\rightarrow \text{“}x \text{ and } y \text{ have different colours”}]$$

Of course, the sentences “C_1, C_2 and C_3 provide a partition of V” and “x and y have different colours” can be easily expressed as MS_1 formulae, namely:

$(\forall x)[x \in C_1 \wedge x \notin C_2 \wedge x \notin C_3] \vee [x \in C_2 \wedge x \notin C_1 \wedge x \notin C_3] \vee [x \in C_3 \wedge x \notin C_1 \wedge x \notin C_2]$ and $[x \in C_1 \wedge y \notin C_1] \vee [x \in C_2 \wedge y \notin C_2] \vee [x \in C_3 \wedge y \notin C_3]$, respectively. □

4.2. Clique-width of the primal graph. In this section we compare the β-acyclicity and β-hypertree-width with clique-width of the primal graph. By Corollary 4.3, we know that bounded clique-width of the *incidence graph* implies bounded hypertree-width. It can be easily checked that this implication is no longer true if we consider the clique-width of the *primal graph* instead. This can be seen by inspecting the class of cliques, whose primal graphs (which coincide with the cliques themselves) have clique-width 2. On the other hand, the class of cliques is of unbounded tree-width and, therefore, also of unbounded β-hypertree-width, since – in contrast to *hypergraphs* – bounded tree-width and bounded hypertree-width coincide on *graphs*. This is due to the fact that, for every graph, a hypertree decomposition $\langle T, \chi, \lambda\rangle$ of width k corresponds to a tree decomposition $\langle T, \chi\rangle$ of width $\leq 2k$.

Now the question naturally arises as to whether bounded clique-width of the primal graphs allows for a strictly larger class of hypergraphs than bounded hypertree-width or at least than bounded β-hypertree-width. Below, we give a negative answer.

THEOREM 4.6 (clique-width of the primal graph versus β-acyclicity). *The class of β-acyclic hypergraphs is of unbounded clique-width w.r.t. the primal graphs.*

Proof. We consider again the class $(\mathcal{H}_n)_{n\geq 1}$ of β-acyclic hypergraphs of Theorem 4.2, where $\mathcal{H}_n$ has the vertices $V = \{y_1, \ldots, y_n\} \cup \{x_{ij} : 1 \leq i < j \leq n\}$ and the hyperedges $H_1, \ldots, H_n$ with $H_l = \{y_l\} \cup \{x_{\alpha\beta} : \alpha < \beta \leq l\} \cup \{x_{l\gamma} : l < \gamma \leq n\}$ for $l \in \{1, \ldots, n\}$.

Similarly to Theorem 4.2, we show that the clique-width of the primal graph $\mathcal{P}_n$ of $\mathcal{H}_n$ increases with n. Note however, that the situation here is a bit different from Theorem 4.2. In particular, $\mathcal{P}_n$ contains only nodes y_i and x_{ij}, but no H_i's. Moreover, the x_{ij}'s form a big clique in $\mathcal{P}_n$, since all of these vertices occur in the hyperedge H_n. Nevertheless, we can show that $cw(\mathcal{P}_n) \geq \frac{n-1}{2}$ holds for every $n \geq 2$.

Suppose on the contrary that $\mathcal{P}_n$ is defined by some k-expression t with $k < \frac{n-1}{2}$. Moreover, let t' be a subexpression in t, s.t. for some $i \in \{1, \ldots, n\}$ the node y_i and at least $n-2$ nodes from the set $\{x_{1i}, \ldots, x_{(i-1)i}, x_{i(i+1)}, \ldots, x_{in}\}$ have already been

introduced in t'. Moreover, let t' be minimal with this property. Then t' is again of the form $t' = r \oplus s$. We derive a contradiction in the following way:

Fact 1: Suppose that two vertices y_α and y_β with $\alpha \neq \beta$ have been introduced by the subexpression t' of t. Then y_α and y_β have different labels in the graph generated by t'. This is due to the fact that, for $\alpha < \beta$, the vertices in $X_\beta = \{x_{1\beta}, \ldots, x_{(\alpha-1)\beta}, x_{(\alpha+1)\beta}, \ldots, x_{(\beta-1)\beta}, x_{\beta(\beta+1)}, \ldots, x_{\beta n}\}$ distinguish the vertices y_α and y_β.

Fact 2: Let X_i be defined as $X_i = \{x_{1i}, \ldots, x_{(i-1)i}, x_{i(i+1)}, \ldots, x_{in}\}$. Moreover, suppose that $x_{\alpha i}$ and $x_{\gamma\delta}$ are nodes in X_i, s.t. $\alpha < \gamma$ (i.e.: $x_{\alpha i}$ occurs in X_i "to the left" of $x_{\gamma\delta}$) and both nodes already exist in t'. Then either $x_{\alpha i}$ and $x_{\gamma\delta}$ have distinct labels in t' or y_α already exists in t'. This follows immediately from the fact that $x_{\alpha i}$ is adjacent to y_α in $\mathcal{P}_n$, whereas $x_{\gamma\delta}$ is not.

Fact 3: Let X_i be defined as above and suppose that $x_{i\beta}$ and $x_{\gamma\delta}$ are vertices in X_i, s.t. both vertices already exist in t' and $\beta < \delta$ holds (i.e., again $x_{i\beta}$ occurs in X_i "to the left" of $x_{\gamma\delta}$). Then either $x_{i\beta}$ and $x_{\gamma\delta}$ have distinct labels in t' or y_β already exists in t', since y_β is adjacent to $x_{i\beta}$ in $\mathcal{P}_n$ but not to $x_{\gamma\delta}$.

Now we define two sets $\mathcal{X}$ and $\mathcal{Y}$ of vertices which have already been introduced by t'. These two sets together will contain $n-1$ vertices in total. Moreover, we can show that the vertices contained in each of these sets have pairwise distinct labels.

(i) Initially, we set $\mathcal{X} := \emptyset$ and $\mathcal{Y} := \{y_i\}$.

(ii) Let X denote the set of those vertices in $\{x_{1i}, \ldots, x_{(i-1)i}, x_{i(i+1)}, \ldots, x_{in}\}$, that already exist in the k-expression t'. By assumption, there are $n-2$ such vertices. Now we traverse the elements in X from right to left. If the label of some $x_{\alpha\beta} \in X$ does not occur in X to the right of $x_{\alpha\beta}$, then we add $x_{\alpha\beta}$ to $\mathcal{X}$. Otherwise, suppose that $x_{\gamma\delta}$ is a vertex to the right of $x_{\alpha\beta}$, s.t. $x_{\alpha\beta}$ and $x_{\gamma\delta}$ have the same label. Then we distinguish two cases: If $\alpha < i$ and $\beta = i$ hold, then y_α already exists in the k-expression t' by Fact 2. Hence, we may add y_α to $\mathcal{Y}$. Otherwise (i.e.: $\alpha = i$ and $\beta > i$ hold), y_β must already exist in t' by Fact 3 and we add y_β to $\mathcal{Y}$.

We have $|\mathcal{X}| + |\mathcal{Y}| = n-1$. Thus, one of the sets $\mathcal{X}$ or $\mathcal{Y}$ must have at least $\frac{n-1}{2}$ vertices. Moreover, these vertices have pairwise distinct labels in t' by the construction (in case of $\mathcal{X}$) and by Fact 1 above (in case of $\mathcal{Y}$). □

By Theorem 4.6 above and the fact that β-acyclic hypergraphs have β-hypertree-width 1, we know that bounded β-hypertree-width does not necessarily imply bounded clique-width of the primal graph. Moreover, it has already been explained above that bounded clique-width of the primal graph does not necessarily imply bounded β-hypertree-width. We thus get the following result:

COROLLARY 4.7 (uncomparability). *The concepts of bounded β-hypertree-width of hypergraphs and bounded clique-width of the corresponding primal graphs are uncomparable, i.e., on the one hand, there exists a class $\mathcal{C}_1$ of hypergraphs, s.t. the β-hypertree-width of the hypergraphs in $\mathcal{C}_1$ is bounded by some fixed constant k while the corresponding class of primal graphs is of unbounded clique-width. On the other hand, there exists a class $\mathcal{C}_2$ of hypergraphs, s.t. the β-hypertree-width of the hypergraphs in $\mathcal{C}_2$ is unbounded while the clique-width of the corresponding primal graphs is bounded by some fixed constant k.*

5. Generalized Tree-width. As has already been mentioned in §2, clique-width is much more powerful than tree-width. On the other hand, the lack of an efficient procedure for recognizing graphs with clique-width $\leq k$ for some arbitrary but fixed k is a major drawback of clique-width. Hence, it is worth trying to extend the notion of tree-width to some kind of "generalized tree-width", which is more powerful than tree-width and which is still efficiently recognizable. One such generalization is proposed below.

Recall from [23], that the existence of a big complete bipartite graph as a subgraph of a graph $\mathcal{G}$ has a very bad effect on the tree-width of $\mathcal{G}$, e.g.: consider the sequence $(\mathcal{H}_n)_{n\geq 1}$ of hypergraphs, where $\mathcal{H}_n$ has vertices $V = \{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_n\}$ and n hyperedges $H_1, \ldots, H_n$ with $H_i = \{y_i, x_1, \ldots, x_n\}$. Then, for every n, the (incidence graph $\mathcal{I}_n$ of the) hypergraph $\mathcal{H}_n$ has tree-width n, since it contains the complete bipartite graph with nodes $\{x_1, \ldots, x_n\}$ and $\{H_1, \ldots, H_n\}$, respectively. On the other hand, for every n, $\mathcal{H}_n$ is γ-acyclic, i.e.: the simple transformations of the γ-acyclicity algorithm in [20] (recalled in §4.1) suffice to reduce the incidence graph of $\mathcal{H}_n$ to the empty graph. In particular, the complete bipartite graph contained in the incidence graph of any such hypergraph can be eliminated by these simple transformations. It therefore makes sense to consider the following generalization of the tree-width:

Definition 5.1 (generalized tree-width). *Let $\mathcal{G}$ be an arbitrary graph and let $\mathcal{G}'$ be the graph that results from exhaustive application of the following rules: deletion of isolated nodes, deletion of ear nodes and contraction of two-element modules. Then we define the* generalized tree-width *of $\mathcal{G}$ as $gtw(\mathcal{G}) = tw(\mathcal{G}')$.*

In order make sure that $gtw(\mathcal{G})$ is well defined, we need the following property of the above transformation rules:

Proposition 5.2. *Let $\mathcal{G}$ be an arbitrary graph and let $\mathcal{G}'$ and $\mathcal{G}''$ be graphs that result from exhaustive application of the following rules: deletion of isolated nodes, deletion of ear nodes and contraction of two-element modules. Then $\mathcal{G}'$ and $\mathcal{G}''$ are isomorphic.*

Proof. The number of possible applications of the above transformation rules is clearly finite. Hence, by general considerations on rewrite systems (cf. [3, 22]), it suffices to prove the following "local confluence" property: Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be graphs that can be obtained from some graph $\mathcal{G}$ via a single rule application. Then there exist graphs $\mathcal{G}_1'$ and $\mathcal{G}_2'$, s.t. $\mathcal{G}_1'$ and $\mathcal{G}_2'$ are isomorphic and $\mathcal{G}_i$ can be transformed into $\mathcal{G}_i'$ (for $i \in \{1, 2\}$) via finitely many applications of the above transformation rules. This can be easily shown by a case distinction over all 3×3 possibilities of rules that lead to $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively, e.g.: Suppose that $\mathcal{G}_1$ is obtained from $\mathcal{G}$ via the first rule (i.e., deletion of isolated nodes) and that $\mathcal{G}_2$ is obtained from $\mathcal{G}$ via the third rule (i.e., contraction of two-element modules). Moreover, let N_i with $i \in \{1, 2\}$ denote the node that is deleted from $\mathcal{G}$ in order to arrive at $\mathcal{G}_i$. Then $\mathcal{G}_1' = \mathcal{G}_2'$ is simply obtained from $\mathcal{G}_1$ by deleting also N_2 (via the third rule) and by deleting N_1 from $\mathcal{G}_2$ (via the first rule), respectively. The remaining cases are handled similarly. □

A polynomial time algorithm for recognizing the graphs with $gtw \leq k$ for some fixed k can be constructed in the obvious way, namely: First, an input graph $\mathcal{G}$ is transformed into $\mathcal{G}'$ via the transformation given in Definition 5.1 above. Then we can apply the algorithm of [6], which decides in linear time, whether $tw(\mathcal{G}') \leq k$ holds.

For our considerations on the tractability of evaluating MS_1 formulae, we are ultimately interested in the relationship between generalized tree-width and clique-width. In Theorem 5.7 we shall show that bounded generalized tree-width implies bounded clique-width. For the proof of this result, we have to revisit and appropriately modify the algorithm in [27] for constructing a 3-expression of an arbitrary distance-hereditary graph.

The notions of "pruning sequence" and "pruning tree" are central to the algorithm in [27]. By slightly modifying these notions to our purposes here, we get the following definitions.

Definition 5.3 (pruning sequence). *Let $\mathcal{G}_0 = \mathcal{G}$, $\mathcal{G}_1, \ldots, \mathcal{G}_n = \mathcal{G}'$ be a sequence of graphs, s.t., for every $i \geq 1$, $\mathcal{G}_i$ is obtained from $\mathcal{G}_{i-1}$ by deleting some node V_{α_i}*

via one of the following rules: deletion of isolated nodes, deletion of ear nodes and contraction of two-element modules. Then $S = s_1, \ldots, s_n$ *is called a* pruning sequence *from* $\mathcal{G}$ *to* $\mathcal{G}'$*, where* s_i *with* $1 \leq i \leq n$ *is defined as follows:*[3]

(i) *If* V_{α_i} *is an isolated node in* $\mathcal{G}_{i-1}$*, then* $s_i = \langle (V_{\alpha_i}, *), \mathit{isolated} \rangle$.

(ii) *If* V_{α_i} *is an ear node in* $\mathcal{G}_{i-1}$*, s.t. its only neighbour is* V_{β_i}*, then* $s_i = \langle (V_{\alpha_i}, V_{\beta_i}), \mathit{ear} \rangle$.

(iii) *If* V_{α_i} *and* V_{β_i} *form a two-element module in* $\mathcal{G}_{i-1}$*, s.t.* V_{α_i} *and* V_{β_i} *are adjacent, then* $s_i = \langle (V_{\alpha_i}, V_{\beta_i}), \mathit{true} \rangle$*. In this case,* V_{α_i} *is called a* true twin *of* V_{β_i} *in* $\mathcal{G}_{i-1}$.

(iv) *If* V_{α_i} *and* V_{β_i} *form a two-element module in* $\mathcal{G}_{i-1}$*, s.t.* V_{α_i} *and* V_{β_i} *are not adjacent, then* $s_i = \langle (V_{\alpha_i}, V_{\beta_i}), \mathit{false} \rangle$*. In this case,* V_{α_i} *is called a* false twin *of* V_{β_i} *in* $\mathcal{G}_{i-1}$.

Definition 5.4 (pruning trees). *Let* $\mathcal{G}_0 = \mathcal{G}, \mathcal{G}_1, \ldots, \mathcal{G}_n = \mathcal{G}'$ *be a sequence of graphs with* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *and* $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$*. Moreover, let* $S = s_1, \ldots, s_n$ *be a pruning sequence from* $\mathcal{G}$ *to* $\mathcal{G}'$ *and let* $\mathcal{V}'' \subseteq \mathcal{V}$ *be defined as*

$$\begin{aligned} \mathcal{V}'' = \{V_{\alpha_i} \mid s_i = \langle (V_{\alpha_i}, *), \mathit{isolated} \rangle\} \cup \\ \{V_{\beta_i} \mid s_i = \langle (V_{\alpha_i}, V_{\beta_i}), x \rangle \text{ with } x \in \{\mathit{ear}, \mathit{true}, \mathit{false}\} \text{ and } V_{\beta_i} \in \mathcal{V}'\}, \end{aligned}$$

i.e., $\mathcal{V}''$ *contains those nodes from* $\mathcal{V}$ *that were eventually deleted as isolated nodes plus those nodes which were "responsible" for the deletion of other nodes (i.e., of ear nodes or true/false twins) without ever being deleted themselves.*

Now let $m = |\mathcal{V}''|$*. Moreover, w.l.o.g., we assume that* $\mathcal{V}'' = \{V_1, \ldots, V_m\}$*. Then there exist* m pruning trees $\mathcal{T}_1, \ldots, \mathcal{T}_m$ *corresponding to the pruning sequence* S*. These pruning trees are directed, edge-labelled trees. They are obtained by the algorithm* construct-pruning-trees *given below.*

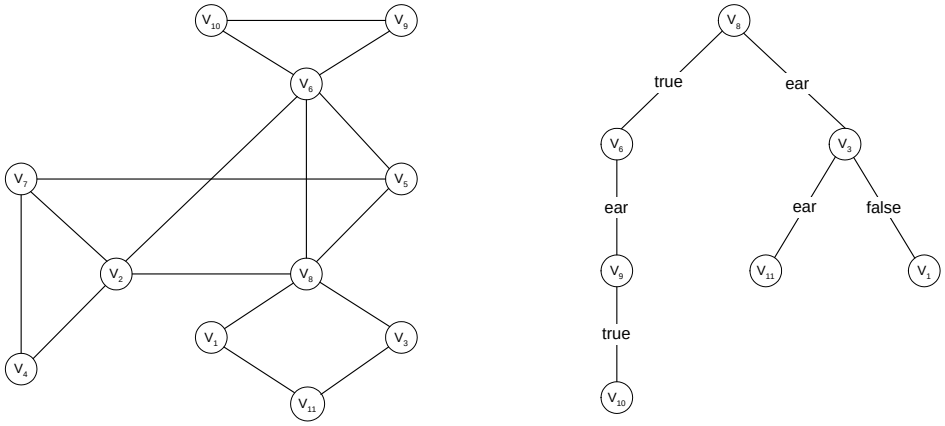
Algorithm construct-pruning-trees.

begin
/* initialization of the pruning trees */
  for $i := 1$ **to** m **do**
    $\mathcal{T}_i :=$ the tree consisting of the root node V_i only;
  od;
/* adding the nodes deleted from $\mathcal{G}$ to the pruning trees */
  for $i := n$ **downto** 1 **do**
    if $s_i = \langle (V_{\alpha_i}, V_{\beta_i}), \mathit{ear} \rangle$ **then** append V_{α_i} as right-most child to V_{β_i} and label the edge from V_{β_i} to V_{α_i} as "ear";
    elsif $s_i = \langle (V_{\alpha_i}, V_{\beta_i}), \mathit{true} \rangle$ **then** append V_{α_i} as right-most child to V_{β_i} and label the edge from V_{β_i} to V_{α_i} as "true";
    elsif $s_i = \langle (V_{\alpha_i}, V_{\beta_i}), \mathit{false} \rangle$ **then** append V_{α_i} as right-most child to V_{β_i} and label the edge from V_{β_i} to V_{α_i} as "false";
    fi;
  od;
end.

This algorithm is illustrated by the following example:

Example. Let the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be defined as $\mathcal{V} = \{V_1, \ldots, V_{11}\}$ and $\mathcal{E} = \{\{V_1, V_8\}, \{V_1, V_{11}\}, \{V_2, V_4\}, \{V_2, V_7\}, \{V_2, V_8\}, \{V_3, V_8\}, \{V_3, V_{11}\}, \{V_4, V_7\}, \{V_5,$

[3] In [27] and [32], emphasis is put on the introduction of nodes rather than on their deletion. Hence, in the former papers, the order of the elements in a pruning sequence is reversed w.r.t. to the definition here. However, the pruning trees defined next have essentially the same form as in [27].



a) graph $\mathcal{G}$ b) pruning tree $\mathcal{T}$

FIG. 5.1. *$\mathcal{G}$ and $\mathcal{T}$.*

$V_6\}$, $\{V_5, V_7\}$, $\{V_5, V_8\}$, $\{V_6, V_8\}$, $\{V_6, V_9\}$, $\{V_6, V_{10}\}$, $\{V_9, V_{10}\}\}$. The graph $\mathcal{G}$ is displayed in part a) of Figure 5.1.

The nodes $\{V_1, V_3, V_6, V_9, V_{10}, V_{11}\}$ can be deleted via the following pruning sequence: $\langle (V_1, V_3), false\rangle$, $\langle (V_{11}, V_3), ear\rangle$, $\langle (V_3, V_8), ear\rangle$, $\langle (V_{10}, V_9), true\rangle$, $\langle (V_9, V_6), ear\rangle$, $\langle (V_6, V_8), true\rangle$. We thus get the graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ with $\mathcal{V}' = \{V_2, V_4, V_5, V_7, V_8\}$ and $\mathcal{E}' = \{\{V_2, V_4\}, \{V_2, V_7\}, \{V_2, V_8\}, \{V_4, V_7\}, \{V_5, V_7\}, \{V_5, V_8\}\}$. Note that no isolated node was thus deleted. Moreover, V_8 is the only node in $\mathcal{G}'$ that was responsible for the deletion of other nodes. Hence, by the algorithm CONSTRUCT-PRUNING-TREES, only one pruning tree $\mathcal{T}$ can be constructed, which is depicted in part b) of Figure 5.1. □

In [27], several fundamental properties of pruning trees are proven. In particular, it is shown how a pruning tree $\mathcal{T}$ can be used to construct a 3-expression for the subgraph of $\mathcal{G}$ that is induced by the nodes of $\mathcal{T}$. Before we come to this algorithm, we recall the following terminology from [27]: For a node V in the pruning tree $\mathcal{T}$, we write $\mathcal{T}_V$ to denote the set of nodes of the subtree of $\mathcal{T}$ rooted at V. Moreover, we call a node U in $\mathcal{T}_V$ a *twin descendant* of V, iff either $V = U$ or all the edges along the path from V to U are labelled with "true" or "false". Finally, by $\mathcal{G}[\mathcal{T}_V]$ we denote the subgraph of $\mathcal{G}$ that is induced by the nodes in $\mathcal{T}_V$.

LEMMA 5.5 (pruning tree and adjacency). *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ be graphs and let S be a pruning sequence from $\mathcal{G}$ to $\mathcal{G}'$. Moreover, let $\mathcal{T}$ with root V be a pruning tree that is obtained from S via the algorithm* CONSTRUCT-PRUNING-TREES. *Finally let U and U' be nodes in $\mathcal{V}$, s.t. U is in $\mathcal{T}$ and U' is outside $\mathcal{T}$. Then U and U' are adjacent in $\mathcal{G}$, iff V is adjacent to U' in $\mathcal{G}$ and U is a twin descendant of V.*

Proof. Implicit in the proof of Lemma 3.7 in [27] □

Example. Recall the graph $\mathcal{G}$ and the pruning tree $\mathcal{T}$ with root V_8 in Figure 5.1. The only twin descendants of V_8 are the nodes V_6 and V_8 itself. On the one hand, the nodes V_6 and V_8 are indeed adjacent to exactly the same nodes outside $\mathcal{T}$, namely V_2 and V_5. On the other hand, the other nodes in $\mathcal{T}$ (namely $V_1, V_3, V_9, V_{10}, V_{11}$) are not adjacent to any node outside $\mathcal{T}$.

LEMMA 5.6 (3-expressions). *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ be graphs and let S be a pruning sequence from $\mathcal{G}$ to $\mathcal{G}'$. Moreover, let $\mathcal{T}$ denote a pruning tree that is obtained from S via the algorithm* CONSTRUCT-PRUNING-TREES. *Finally, let V be an*

arbitrary node in $\mathcal{T}$ with child nodes $U_1, \ldots, U_l$ from left to right. Then there exists a 3-expression t_V that defines the graph $\mathcal{G}[\mathcal{T}_V]$, s.t. all twin descendants of V are labelled with 2 in $\mathcal{G}[\mathcal{T}_V]$ and all other nodes in $\mathcal{T}_V$ are labelled with 1. Such a 3-expression can be computed inductively by the algorithm CONSTRUCT-3-EXPRESSION *given below.*

Proof. See Claim 3.9 in [27] □

ALGORITHM CONSTRUCT-3-EXPRESSION.

begin
  $t := 2(V)$;
  let $U_1, \ldots, U_l$ denote the child nodes of V from left to right;
  for $i := l$ **downto** 1 **do**
    if the edge from V to U_i is labelled with "ear" **then**
      $t := \rho_{3\to1}(\eta_{2,3}(\rho_{2\to3}(t_{U_i}) \oplus t))$;
    if the edge from V to U_i is labelled with "true" **then**
      $t := \rho_{3\to2}(\eta_{2,3}(\rho_{2\to3}(t_{U_i}) \oplus t))$;
    if the edge from V to U_i is labelled with "false" **then**
      $t := t_{U_i} \oplus t$;
    fi;
  od;
  $t_v := t$;
end.

Note that if the pruning sequence from $\mathcal{G}$ to $\mathcal{G}'$ contains the deletion of an isolated node V, then the subgraph $\mathcal{G}[\mathcal{T}_V]$ of $\mathcal{G}$ is a distance-hereditary connected component of $\mathcal{G}$. As has already been recalled in §4.1, the incidence graphs of γ-acyclic hypergraphs are precisely the distance-hereditary, bipartite graphs. Hence, the above algorithm can be used to compute the 3-expression of (every connected component of) the incidence graph of any γ-acyclic hypergraph. Moreover, it is now also clear why we never had to delete an isolated node in the example above, since the graph $\mathcal{G}$ in part a) of Figure 5.1 consists of a single connected component and this connected component is not distance-hereditary.

Example. We put the algorithm CONSTRUCT-3-EXPRESSION to work by continuing the above example. The 3-expressions t_V for the nodes V in the pruning tree $\mathcal{T}$ in part b) of Figure 5.1 are obtained as follows. As a short-hand notation, we shall write t_i to denote the 3-expression t_{V_i} for any i.

$t_1 = 2(V_1)$
$t_{11} = 2(V_{11})$
$t_3 = \rho_{3\to1}(\eta_{2,3}(\rho_{2\to3}(t_{11}) \oplus (t_1 \oplus 2(V_3))))$
$t_{10} = 2(V_{10})$
$t_9 = \rho_{3\to2}(\eta_{2,3}(\rho_{2\to3}(t_{10}) \oplus 2(V_9)))$
$t_6 = \rho_{3\to1}(\eta_{2,3}(\rho_{2\to3}(t_9) \oplus 2(V_6)))$
$t_8 = \rho_{3\to2}(\eta_{2,3}(\rho_{2\to3}(t_6) \oplus \rho_{3\to1}(\eta_{2,3}(\rho_{2\to3}(t_3) \oplus 2(V_8)))))$ □

We are now ready to prove the following desirable property of the generalized tree-width.

THEOREM 5.7 (bounded generalized tree-width implies bounded clique-width). *Let $\mathcal{C}$ be a class of graphs and let k be some fixed constant, s.t. $gtw(\mathcal{G}) \leq k$ for all $\mathcal{G} \in \mathcal{C}$. Then there exists a constant k', s.t. $cw(\mathcal{G}) \leq k'$ for all $\mathcal{G} \in \mathcal{C}$.*

Proof. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an arbitrary graph and let $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ be the graph obtained from $\mathcal{G}$ by exhaustive application of the rules in Definition 5.1 (i.e., deletion of isolated nodes, deletion of ear nodes and contraction of two-element modules). Moreover, let $gtw(\mathcal{G}) \leq k$, i.e., the condition $tw(\mathcal{G}') \leq k$ holds. Recall from [19],

that then we have $cw(\mathcal{G}') \leq \tau$ with $\tau = 2^{k+1} + 1$. It suffices to show that then also $cw(\mathcal{G}) \leq k'$ with $k' = \tau + 1$ holds.

Let t' be a τ-expression that generates $\mathcal{G}'$. We assume that t' uses the τ labels $\{2, \ldots, k'\}$. Let S be a pruning sequence from $\mathcal{G}$ to $\mathcal{G}'$ and let $\mathcal{T}_1, \ldots, \mathcal{T}_m$ denote the corresponding pruning trees with root nodes $V_1, \ldots, V_m$. Moreover, for every $i \in \{1, \ldots, m\}$, let t_i denote the 3-expression obtained by the above algorithm CONSTRUCT-3-EXPRESSION for the subgraph $\mathcal{G}[\mathcal{T}_i]$ of $\mathcal{G}$. From these 3-expressions $t_1, \ldots, t_m$ together with t', we construct a k'-expression s for $\mathcal{G}$ as follows:

First we transform t' into t'' by the following replacement steps: For every V_i with $i \in \{1, \ldots, m\}$, s.t. V_i is still contained in $\mathcal{G}'$, we know that V_i is eventually introduced in t' by some subexpression of the form $\ell(V_i)$. By Lemma 5.6, V_i has the label 2 in t_i. Then we replace the subexpression $\ell(V_i)$ in t' either by t_i (if $\ell = 2$) or by $\rho_{2\to\ell}(t_i)$ (if $\ell \neq 2$).

Finally, all 3-expressions in $\{t_1, \ldots, t_m\}$ whose root node does not occur in $\mathcal{G}'$ any more, are added to t'' via the $\oplus$-operator. We thus set

$$s := t'' \oplus \bigoplus_{V_i \notin \mathcal{V}'} t_i$$

Obviously, t' is thus transformed into a k'-expression s using the labels $\{1, \ldots, k'\}$. It remains to show that s indeed generates the original graph $\mathcal{G}$.

As far as the nodes are concerned, every node in $\mathcal{V}$ is either introduced by some 3-expression t_i or it is still contained in $\mathcal{V}'$. In the latter case, it is introduced by t'. Hence, by construction, s indeed introduces every node $V \in \mathcal{V}$. On the other hand, no node of $\mathcal{G}$ is introduced twice in s. This is due to the fact that any two distinct pruning trees $\mathcal{T}_i$ and $\mathcal{T}_j$ have no nodes from $\mathcal{V}$ in common and for any pruning tree $\mathcal{T}_i$, only the root node of $\mathcal{T}_i$ can be contained in $\mathcal{V}'$.

As far as the edges of $\mathcal{G}$ are concerned, we know that each 3-expression t_i clearly introduces all edges in the subgraph $\mathcal{G}[\mathcal{T}_i]$ of $\mathcal{G}$. Likewise, t' introduces the edges in $\mathcal{E}'$. Hence, in order to show that s indeed introduces all edges $\{U, V\}$ it only remains to consider the following cases:

(i) Suppose that the nodes U and V occur in two distinct pruning trees $\mathcal{T}_i$ and $\mathcal{T}_j$, respectively. Moreover, let V_i and V_j denote the root nodes of these pruning trees. It follows from Lemma 5.5 that then also V_i and V_j are adjacent. Moreover, V_i and V_j must still be contained in $\mathcal{V}'$. Hence, this edge $\{V_i, V_j\}$ is eventually introduced by t' and thus by s. Moreover, by our construction of s and by Lemma 5.6, we know that U has the same label ℓ as V_i when the subexpression $\ell(V_i)$ in t' is replaced by t_i (if $\ell = 2$) or by $\rho_{2\to\ell}(t_i)$ (if $\ell \neq 2$). Likewise, U gets the same label as V_j. Hence, when eventually the edge between V_i and V_j is introduced in s, then the edge between U and V will be introduced as well.

(ii) Suppose that the node U occurs in some pruning tree $\mathcal{T}_i$ with root node V_i and that V is contained in $\mathcal{V}'$. By the same considerations as above, we know from Lemma 5.5 that then also V_i and V are adjacent. Moreover, U is introduced with the same label ℓ as V_i when the subexpression $\ell(V_i)$ in t' is replaced by t_i or by $\rho_{2\to\ell}(t_i)$, respectively. Hence, when eventually the edge between V_i and V is introduced in s, then the edge between U and V will be introduced as well.

Now consider also the opposite direction, i.e., we have to show that all edges $\{U, V\}$ introduced by s indeed exist in $\mathcal{G}$. Again, if both nodes U and V are in the same pruning tree $\mathcal{T}_i$ or if both nodes are left in $\mathcal{V}'$, then this is obviously the case. Like above, it remains to consider the following two cases:

(i) Suppose that the nodes U and V occur in two distinct pruning trees $\mathcal{T}_i$ and $\mathcal{T}_j$, respectively. Moreover, let V_i and V_j denote the root nodes of these pruning trees. In order to introduce the edge $\{U, V\}$, the expression s and, therefore, t' actually introduces the edge $\{V_i, V_j\}$. Hence, V_i and V_j are indeed adjacent in $\mathcal{G}$. Moreover, U must have the same label as V_i in t_i. Likewise, V and V_j have the identical labels in t_j. Hence, by Lemma 5.6, U is a twin descendant of V_i in the pruning tree $\mathcal{T}_i$ and V is a twin descendant of V_j in $\mathcal{T}_j$. Thus, by Lemma 5.5, U and V are indeed adjacent in $\mathcal{G}$.

(ii) Suppose that the node U occurs in some pruning tree $\mathcal{T}_i$ with root node V_i and that V is contained in $\mathcal{V}'$. By the same considerations as above, we may conclude by the Lemmas 5.5 and 5.6 that V_i and V are indeed adjacent, that V_i is a twin descendant of V_i in the pruning tree $\mathcal{T}_i$ and, finally, that U and V are indeed adjacent in $\mathcal{G}$.

In other words, the k'-expression s introduces exactly the nodes in $\mathcal{V}$ and exactly the edges in $\mathcal{E}$. Hence, s is indeed the desired k'-expression. □

Example. Let $\mathcal{G}$ and $\mathcal{G}'$ be the graphs from the example above. The graph $\mathcal{G}$ together with the pruning tree $\mathcal{T}$ are shown in Figure 5.1. Recall that $\mathcal{G}'$ has the form $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ with $\mathcal{V}' = \{V_2, V_4, V_5, V_7, V_8\}$ and $\mathcal{E}' = \{\{V_2, V_4\}, \{V_2, V_7\}, \{V_2, V_8\}, \{V_4, V_7\}, \{V_5, V_7\}, \{V_5, V_8\}\}$. A 4-expression t' of $\mathcal{G}'$ can be constructed as follows.

(introduce V_4)	$s_1 := 2(V_4)$	(introduce V_2)	$s_2 := 3(V_2) \oplus s_1$
(connect V_2, V_4)	$s_3 := \eta_{2,3}(s_2)$	(introduce V_7)	$s_4 := 4(V_7) \oplus s_3$
(connect V_4, V_7)	$s_5 := \eta_{2,4}(s_4)$	(connect V_2, V_7)	$s_6 := \eta_{3,4}(s_5)$
(introduce V_8)	$s_7 := 5(V_8) \oplus s_6$	(connect V_2, V_8)	$s_8 := \eta_{3,5}(s_7)$
(relabel V_2)	$s_9 := \rho_{3\to 2}(s_8)$	(introduce V_5)	$s_{10} := 3(V_5) \oplus s_9$
(connect V_5, V_7)	$s_{11} := \eta_{3,4}(s_{10})$	(connect V_5, V_8)	$s_{12} := \eta_{3,5}(s_{11})$

Then s_{12} is the desired 4-expression t' using the labels $\{2, \ldots, 5\}$.

Now recall that the 3-expression t of $\mathcal{G}[\mathcal{T}]$ has already been computed above. The 5-expression s that generates $\mathcal{G}$ can be constructed according to the proof of Theorem 5.7 by redefining s_7 as $s_7 := \rho_{2\to 5}(t) \oplus s_6$. Then s_{12} yields the 5-expression s that generates $\mathcal{G}$. Note that in s, the $\eta_{3,5}$ operation applied to s_7 not only introduces the edge $\{V_2, V_8\}$ but also $\{V_2, V_6\}$. Likewise, by the $\eta_{3,5}$ operation applied to s_{11}, the edge $\{V_5, V_6\}$ is introduced. □

By the tractability results recalled in §2.2, Theorem 5.7 immediately yields

COROLLARY 5.8 (MS_1 queries and generalized tree-width). *The evaluation of an arbitrary fixed MS_1 query over a class $\mathcal{C}$ of graphs is tractable, if $\mathcal{C}$ is of bounded generalized tree-width.*

By Theorem 5.7, bounded generalized tree-width implies bounded clique-width while the converse is clearly not true. Just consider the class of cliques, whose generalized tree-width is unbounded while the clique-width of all these graphs is 2. Hence, the concept of bounded generalized tree-width does not allow us to push the tractability barrier for the evaluation of MS_1 queries any further. However, the advantage of this new concept is that, in contrast to bounded clique-width, it can be efficiently recognized.

6. Conclusion. In this paper, we have compared query-width, hypertree-width, and several notions of acyclicity of hypergraphs with clique-width. Note that we have mainly considered the clique-width of the *incidence graph* here. When considering restrictions on conjunctive queries, this choice is somehow justified by the fact that the clique-width of the *primal graph* is irrelevant for the tractability. In particular, as we have pointed out in §2, there are NP-hard classes of queries whose primal graphs

are of bounded clique-width. On the other hand, when considering restrictions on the form of the structures, the primal graphs also play an important role. Actually, we have shown that β-acyclicity and bounded clique-width of the primal graph are uncomparable. However, the exact position of bounded clique-width of the primal graph in Figure 1.1 has to be determined yet.

In §5 we have shown how the insights from the comparison of γ-acyclicity with bounded clique-width can be used for an easy generalization of the tree-width. As long as no polynomial time algorithm for recognizing graphs with clique-width $\leq k$ (for some arbitrary but fixed k) has been found, the search for an appropriate generalization of the tree-width is an interesting research area. We have provided a first and very simple step in this direction, to which further steps should be added, e.g.: rather than just considering two-element modules, we might take arbitrary modules and investigate recursively the generalized tree-width of such a module. Likewise, rather than just deleting ear nodes, we might consider the splitting of a graph into its biconnected components. Here we also have the effect that a single biconnected component may contain a module, that was not a module in the overall graph. Likewise, contraction of a module to a single node may allow us to further decompose a graph into biconnected components. The interplay between these two kinds of decompositions deserves further research efforts.

Acknowledgments. We are very grateful to Andreas Brandstädt for numerous pointers to the literature. In particular, he suggested the proof of Theorem 4.1, which uses known results and which is thus much simpler than the proof originally provided by the authors. We also thank the anonymous referees whose detailed comments have greatly helped us to improve the presentation of this work.

REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*, Addison-Wesley, 1995.
[2] G. Ausiello, A. D'Atri, and M. Moscarini, *Chordality Properties on Graphs and Minimal Conceptual Connections in Semantic Data Models*, Journal of Computer and System Sciences, 33 (1986), pp. 179–202.
[3] F. Baader and T. Nipkow, *Term Rewriting and All That*, Cambridge University Press, 1998.
[4] H.-J. Bandelt and H. M. Mulder, *Distance-hereditary graphs*, Journal of Combinatorial Theory, Series B, 41 (1986), pp. 182–208.
[5] W. Bibel, *Constraint Satisfaction from a Deductive Viewpoint*, Artificial Intelligence, 35 (1988), pp. 401–413.
[6] H. L. Bodlaender, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM Journal on Computing, 25 (1996), pp. 1305–1317.
[7] ———, *Treewidth: Algorithmic Techniques and Results*, in Proc. Mathematical Foundations of Computer Science, 22nd Int. Symp. (MFCS'97), I. Prívara and P. Ruzicka, eds., no. 1295 in Lecture Notes in Computer Science, Springer Verlag, 1997, pp. 19–36.
[8] A. Brandstädt, 2002. personal communication.
[9] A. Brandstädt, V. B. Lee, and J. P. Spinrad, *Graph Classes: A Survey*, Monographs on Discrete Mathematics and Applications, Volume 3, SIAM, 1999.
[10] A. Brandstädt and V. V. Lozin, *On the linear structure of clique-width of bipartite permutation graphs*, Tech. Report RRR 29-2001, Rutgers, 2001.
[11] A. K. Chandra and P. M. Merlin, *Optimal Implementation of Conjunctive Queries in Relational Data Bases*, in Proc. 9th Annual ACM Symp. on Theory of Computing (STOC'77), ACM Press, 1977, pp. 77–90.
[12] C. Chekuri and A. Rajaraman, *Conjunctive Query Containment Revisited*, in Proc. 6th Int. Conf. on Database Theory (ICDT'97), no. 1186 in Lecture Notes in Computer Science, Springer Verlag, 1997, pp. 130–144.
[13] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, MIT Press, 1999.
[14] D. G. Corneil, M. Habib, J.-M. Langlinel, and U. R. B. Reed, *Polynomial Time Recognition of clique-width ≤ 3 graphs, extended abstract.*, in Proc. Latin American Theoretical

Informatics (LATIN 2000), no. 1776 in Lecture Notes in Computer Science, Springer Verlag, 2000, pp. 126–134.

[15] B. Courcelle, *Graph Rewriting: An Algebraic and Logic Approach*, in Handbook of Theoretical Computer Science, Vol. 2, J. van Leeuwen, ed., Elsevier Science Publishers, 1990, pp. 194–241.

[16] ———, *Monadic second-order logic of graphs VII: Graphs as relational structures*, Theoretical Computer Science, 101 (1992), pp. 3–33.

[17] B. Courcelle, J. Engelfriet, and G. Rozenberg, *Handle-rewriting hypergraph grammars*, Journal of Computer and System Sciences, 46 (1993), pp. 218–270.

[18] B. Courcelle, J. A. Makowsky, and U. Rotics, *Linear time solvable optimization problems on graphs of bounded clique-width*, Theory of Computing Systems, 33 (2000), pp. 125–150.

[19] B. Courcelle and S. Olariu, *Upper bounds on the clique-width of graphs*, Discrete Applied Mathematics, 101 (2000), pp. 77–114.

[20] A. D'Atri and M. Moscarini, *Acyclic Hypergraphs: Their recognition and top-down versus bottom-up generation*, Tech. Report R.29, Consiglio Nazionale delle Ricerche, Istituto di Analisi dei Sistemi ed Informatica, 1982.

[21] A. D'Atri and M. Moscarini, *Distance-Hereditary Graphs, Steiner Trees, and Connected Domination*, SIAM Journal on Computing, 17 (1988), pp. 521–538.

[22] N. Dershowitz and J.-P. Jouannaud, *Rewrite Systems*, in Handbook of Theoretical Computer Science (Vol. B): Formal Models and Semantics, J. van Leeuwen, ed., Elsevier, 1990, pp. 243–320.

[23] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer Verlag, 1997.

[24] R. Fagin, *Degrees of Acyclicity for Hypergraphs and Relational Database Schemes*, Journal of the ACM, 30 (1983), pp. 514–550.

[25] J. Flum, M. Frick, and M. Grohe, *Query Evaluation via Tree-Decomposition*, Journal of the ACM, 49 (2002), pp. 716–752.

[26] M. R. Garey and D. S. Johnson, *Computers and Intractability, A guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.

[27] M. C. Golumbic and U. Rotics, *On the clique-width of perfect graph classes*, Int. Journal of Foundations of Computer Science, 11 (2000), pp. 423–443.

[28] G. Gottlob, N. Leone, and F. Scarcello, *A Comparison of Structural CSP Decomposition Methods*, Artificial Intelligence, 124 (2000), pp. 243–282.

[29] ———, *Hypertree Decompositions and Tractable Queries*, Journal of Computer and System Sciences, 64 (2002), pp. 579–627.

[30] G. Gottlob and R. Pichler, *Hypergraphs in model checking: Acyclicity and hypertree-width versus clique-width*, in Proc. 28th Int. Coll. on Automata, Languages and Programming (ICALP 2001), F. Orejas, P. G. Spirakis, and J. van Leeuwen, eds., no. 2076 in Lecture Notes in Computer Science, Springer Verlag, 2001, pp. 708–719.

[31] M. Grohe, T. Schwentick, and L. Segoufin, *When is the Evaluation of Conjunctive Queries Tractable?*, in Proc. 32nd Annual ACM Symp. on Theory of Computing (STOC'01), ACM Press, 2001, pp. 657–666.

[32] P. L. Hammer and F. Maffray, *Completely separable graphs*, Discrete Applied Mathematics, 27 (1990), pp. 85–99.

[33] P. G. Kolaitis and M. Y. Vardi, *Conjunctive-Query Containment and Constraint Satisfaction*, in Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS'98), ACM Press, 1998, pp. 205–213.

[34] K. Kunen, *Answer Sets and Negation as Failure*, in Proc. 4th Int. Conf. on Logic Programming (ICLP'87), J.-L. Lassez, ed., MIT Press, 1987, pp. 219–228.

[35] L. J. Stockmeyer, *The Complexity of Decision Problems in Automata Theory*, PhD thesis, Department of Electrical Engineering, MIT, 1974.

[36] M. Yannakakis, *Algorithms for Acyclic Database Schemes*, in Proc. Int. Conf. on Very Large Data Bases (VLDB'81), C. Zaniolo and C. Delobel, eds., IEEE Computer Society, 1981, pp. 82–94.