Query evaluation via tree-decompositions

Jörg Flum Markus Frick Martin Grohe
Institut für Mathematische Logik, Eckerstr. 1, 79104 Freiburg, Germany
Email: flum@uni-freiburg.de, {frick,grohe}@logik.mathematik.uni-freiburg.de

A number of efficient methods for evaluating first-order and monadic-second order queries on finite relational structures are based on tree-decompositions of structures or queries. We systematically study these methods.

In the first-part of the paper we consider arbitrary formulas on tree-like structures. We generalize a theorem of Courcelle [2] by showing that on structures of bounded tree-width a monadic second-order formula (with free first- and second-order variables) can be evaluated in time linear in the structure size plus the size of the output.

In the second part we study tree-like formulas on arbitrary structures. We generalize the notions of acyclicity and bounded tree-width from conjunctive queries to arbitrary first-order formulas in a straightforward way and analyze the complexity of evaluating formulas of these fragments. Moreover, we show that the acyclic and bounded tree-width fragments have the same expressive power as the well-known guarded fragment and the finite-variable fragments of first-order logic, respectively.

This is an abstract of [4].

Tree-Decompositions

A *hypergraph* $\mathcal{H}$ is pair $(H, E^{\mathcal{H}})$ consisting of a set H of *vertices* and a set $E^{\mathcal{H}}$ of subsets of H called *hyperedges*.

A *tree-decomposition* of a hypergraph $\mathcal{H}$ is a pair $(\mathcal{T}, (H_t)_{t\in T})$, where $\mathcal{T}$ is a tree and $(H_t)_{t\in T}$ a family of subsets of H (called the *blocks* of the decomposition) such that

(1) For every $v \in H$, the set $\{t \in T \mid v \in H_t\}$ is non-empty and connected (i.e. a subtree).

(2) For every $e \in E^{\mathcal{H}}$ there is a $t \in T$ such that $e \subseteq H_t$.

The *width* of a tree-decomposition $(\mathcal{T}, (H_t)_{t\in T})$ is $\max\{|H_t| \mid t \in T\} - 1$. The *tree-width* $\mathrm{tw}(\mathcal{H})$ of $\mathcal{H}$ is the minimal width of a tree-decomposition of $\mathcal{H}$.

A hypergraph $\mathcal{H}$ is *acyclic* if it has a tree-decomposition $(\mathcal{T}, (H_t)_{t\in T})$ where for every $t \in T$ there exists an $e \in E^{\mathcal{H}}$ such that $H_t = e$.

Part 1: Tree-Like Structures

A *tree-decomposition of a relational structure* $\mathcal{A}$ is a tree-decomposition of the hypergraph whose vertex set is the universe A of $\mathcal{A}$ and whose hyperedges are all sets $\{a_1, \dots, a_r\}$ such that the tuple $(a_1, \dots, a_r)$ is in some relation of $\mathcal{A}$. The *tree-width* of $\mathcal{A}$ is defined accordingly.

Let $\varphi(X_1, \dots, X_l, x_1, \dots, x_m)$ be a formula of monadic second-order logic with free set variables $X_1, \dots, X_l$ and free element variables $x_1, \dots, x_m$. Then for every structure $\mathcal{A}$ we let $\varphi(\mathcal{A})$ be the set of all tuples $(B_1, \dots, B_l, b_1, \dots, b_m)$ such that $A \models \varphi(B_1, \dots, B_l, b_1, \dots, b_m)$.

Theorem 1. *There exists a function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and an algorithm that, given a structure $\mathcal{A}$ and a formula φ of monadic second-order logic, computes the set $\varphi(\mathcal{A})$ in time*

$$f\big(||\varphi||, \mathrm{tw}(\mathcal{A})\big)\Big(|A| + ||\varphi(\mathcal{A})||\Big).$$

Here $||\varphi||$ is the length of the formula φ, $|A|$ the cardinality of the universe of the structure $\mathcal{A}$, and $||\varphi(\mathcal{A})||$ the size of a reasonable encoding of the set $\varphi(\mathcal{A})$.

As applications of this theorem, for instance, we can find *all* cliques of of a graph $\mathcal{G}$ of bounded tree-width in time $O(|G| + \sum_{C \text{ clique in } \mathcal{G}} |C|)$, or we can find all Hamiltonian cycles of $\mathcal{G}$ in time $O(|G| \cdot h)$, where h is the number of Hamiltonian cycles of $\mathcal{G}$.

Part 2: Tree-Like Formulas

With every first-order formula φ we associate a hypergraph $\mathcal{H}_\varphi$ whose vertices are the variables of φ and whose edges are the sets $\{x_1, \dots, x_k\}$ of variables such that φ contains an atom $\alpha(x_1, \dots, x_k)$. A *tree-decomposition of* φ is a tree-decomposition of $\mathcal{H}_\varphi$, and *acyclicity* and *tree-width* are defined with respect to such decompositions. There is a refined notion that sometimes yields better results: A tree-decomposition of φ is *strict* if the set of all free variables of φ is contained in a block of the decomposition. This yields the notions of *strict* acyclicity and strict tree-width.

We start by reviewing (and precisely analyzing) an algorithm due to Yannakakis for evaluating *acyclic conjunctive queries*. Recall that a conjunctive query is a formula of the form $\exists \bar{y} \bigwedge_{i=1}^{n} \alpha_i$, where all α_i are atoms. The *evaluation problem* for a class L of formulas is the problem of computing $\varphi(\mathcal{A})$, given $\varphi \in \mathrm{L}$ and $\mathcal{A}$.

Theorem 2 (Yannakakis [7]). *There is an algorithm that solves the evaluation problem for acyclic conjunctive queries in time* $O\big(||\varphi|| \cdot ||\mathcal{A}|| \cdot (||\varphi(\mathcal{A})|| + 1)\big)$.

For strictly acyclic conjunctive queries, this can be improved to $O\big(||\varphi|| \cdot ||\mathcal{A}||\big)$.

An easy consequence of this theorem is an analogous result for conjunctive queries of bounded tree-width due to Chekuri and Rajaraman [1].

We generalize (strict) acyclicity and tree-width from conjunctive queries to *non-recursive stratified datalog programs* (*NRSD-programs* for short). Then Yannakakis's algorithm yields:

Corollary 3. *(1) The evaluation problem for acyclic NRSD-programs* Π *can be solved in time* $O\big(||\Pi|| \cdot ||\mathcal{A}|| \cdot (||\Pi(\mathcal{A})|| + 1)\big)$. *For strictly acyclic NRSD-programs, this can be improved to* $O\big(||\Pi|| \cdot ||\mathcal{A}||\big)$.

(2) The evaluation problem for NRSD-programs Π *can be solved in time* $O\big(||\Pi||^{w+2} + ||\Pi|| \cdot (|A|^{w+1} + ||\mathcal{A}||) \cdot (||\Pi(\mathcal{A})|| + 1)\big)$, *where* $w := \mathrm{tw}(\Pi)$, *and in time* $O\big(||\Pi||^{s+2} + ||\Pi|| \cdot (|A|^{s+1} + ||\mathcal{A}||)\big)$, *where* s *is the strict tree-width of* Π.

The purpose of this generalization to NRSD-programs is a well-known translation between NRSD-programs and formulas of first-order logic. It turns out that under this translation acyclic NRSD-programs and programs of bounded tree-width correspond to well-known fragments of first-order logic:

Theorem 4. *(1) A Boolean query is definable by a strictly acyclic NRSD-program if, and only if, it is definable in the* guarded fragment *of first-order logic.*

(2) Let $k \geq 1$. *Then a query is definable by an NRSD-program of strict tree-width at most* $k-1$ *if, and only if, it is definable in* FO^k, *the* k-variable fragment *of first-order logic.*

Let us remark that there is a generalization of (1) to non-Boolean queries. Furthermore, the translation between formulas and programs are effective; we also analyze their complexity.

Gottlob, Grädel, and Veith [5] give a similar translation between sentences of the guarded fragment and so-called *Datalog LITE*-programs. Moreover, Kolaitis and Vardi [6] established (2) on the level of conjunctive queries.

It is well-known that queries in the guarded fragment and the finite-variable fragments can be evaluated much more efficiently than arbitrary first-order formulas. The interesting consequence of Theorem 4 is that these results, which look quite different at first sight, can both be naturally explained in terms of a tree-like structure of the formulas.

The Overall Picture

The algorithms in both parts of the paper are very similar. This can best be explained in terms of tree-automata. To prove Theorem 1, we translate the input formula φ to a tree-automaton and then run this automaton on a tree-decomposition of the input structure. Theorem 4 can be proved analogously with the roles of formula and structure switched: We translate the input structure to an automaton and run this automaton on a tree-decomposition of the formula.

If we just want to evaluate sentences, this shows that the algorithms in both parts of the paper are more or less the same. A difference between the two parts is in how to extract the output when evaluating formulas with free variables.

On a more abstract level, both problems amount to deciding whether there is a homomorphism between two relational structures [3]. And in both cases we use the well-known fact that it is decidable in polynomial time if an arbitrary structure $\mathcal{A}$ contains a homomorphic image of a structure $\mathcal{B}$ of bounded tree-width.

References

[1] Ch. Chekuri and A. Rajaraman. Conjunctive query containment revisited. In Ph. Kolaitis and F. Afrati, editors, *Proceedings of the 5th International Conference on Database Theory*, volume 1186 of *Lecture Notes in Computer Science*, pages 56–70. Springer-Verlag, 1997.

[2] B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 2, pages 194–242. Elsevier Science Publishers, 1990.

[3] T. Feder and M.Y. Vardi. Monotone monadic SNP and constraint satisfaction. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 612–622, 1993.

[4] J. Flum, M. Frick, and M. Grohe. Query-evaluation via tree-decompositions. Submitted for publication, 2000. Available at http://sunpool.mathematik.uni-freiburg.de/home/grohe.

[5] G. Gottlob, E. Grädel, and H. Veith. Datalog lite: Temporal versus deductive reasoning in verification. Technical Report DBAI-TR-98-22, Technische Universität Wien, 1998.

[6] Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, pages 205–213, 1998.

[7] M. Yannakakis. Algorithms for acyclic database schemes. In *7th International Conference on Very Large Data Bases*, pages 82–94, 1981.