COMPLEXITY COLUMN

NEIL IMMERMAN, University of Massachusetts Amherst
immerman@cs.umass.edu

Monadic Second-Order Logic (MSO) expresses many natural NP Complete problems as well as problems complete for all levels of the Polynomial-Time Hierarchy. Thus, it was surprising when Bruno Courcelle proved that for inputs of any fixed tree width, all MSO properties are checkable in linear time [Courcelle 1990].

Recently, Michael Elberfeld and his coauthors have refined Courcelle's Theorem in several striking ways. In particular, they have characterized the complexity of the relevant decision, optimization and counting problems, for structures of bounded tree width as well as for more restricted inputs. All of these problems are inside deterministic logspace, L. Some of them are complete for L, and others live in a variety of smaller classes inside L.

Elberfeld explains these ground-breaking results in the following clear and high-level exposition.

Variants of Courcelle's Theorem for Complexity Classes inside P

Michael Elberfeld, RWTH Aachen University

It is well known that monadic-second order logic (MSO) expresses many natural NP-complete problems. However, a famous theorem of Courcelle states that every problem expressible in MSO can be solved in linear time for input graphs whose tree width is bounded by a fixed constant [Courcelle 1990]. Courcelle's Theorem is the prototypical example of an algorithmic "meta theorem", which states an algorithmic upper bound for a whole family of problems. This column reviews a series of meta theorems several of which refine Courcelle's Theorem by more precisely classifying the complexity of natural families of problems.

I will describe concepts, results, and applications that appeared in joint works with Andreas Jakoby and Till Tantau and in my dissertation [Elberfeld et al. 2010; 2012; Elberfeld 2012]. I will also describe some recent developments and open problems. I would like to thank Neil Immerman for inviting me to write a column about "Courcelle's Theorem and Logspace complexity". I hope you enjoy it.

1. INTRODUCTION

Many computational problems can be defined in monadic second-order logic. For example, the 3-COLORABILITY problem is expressed by the following MSO formula,

$$\varphi_{\text{3-colorable}} := \exists R\, \exists G\, \exists B\, \forall v((R(v) \vee G(v) \vee B(v)) \wedge \forall w(E(v,w) \rightarrow \neg(R(v) \wedge R(w)) \wedge \neg(G(v) \wedge G(w)) \wedge \neg(B(v) \wedge B(w)))) \,.$$

A graph $\mathcal{G}$ satisfies this formula ($\mathcal{G} \models \varphi_{\text{3-colorable}}$) if its vertices may be colored red, green or blue so that adjacent vertices do not have the same color. For example, $\triangledown \models \varphi_{\text{3-colorable}}$ but $\boxtimes \not\models \varphi_{\text{3-colorable}}$.

In addition to defining *decision* problems like 3-COLORABILITY, we can use MSO-formulas to define *counting* and *optimization* problems. Consider the formula,

$$\varphi_{\text{dominates}}(X_1) \quad := \quad \forall v\,(X_1(v) \vee \exists w\,(X_1(w) \wedge E(v,w))) \tag{1}$$

which has a free set-variable, X_1. A graph $\mathcal{G}$ and a subset of its vertices $D \subseteq V(\mathcal{G})$ satisfies $\varphi_{\text{dominates}}$ if D is a dominating set in $\mathcal{G}$. We can also use the formula $\varphi_{\text{dominates}}(X_1)$ to express the problem of *counting the number* of dominating sets, or to compute the size of a smallest dominating set. The latter is the well known optimization problem DOMINATING-SET.

The problem 3-COLORABILITY and the decision variant of DOMINATING-SET are well known NP-complete problems [Garey and Johnson 1979]. However, by Courcelle's Theorem, for graphs of bounded tree width, these problems can be solved in polynomial time.

A graph has tree width w if it can be decomposed into a tree of subgraphs called bags of size $\leq w+1$. (See Chapter 11 in the book of Flum and Grohe [2006] for formal definitions and basic properties related to tree decompositions.) The tree structure of the decomposition captures the global tree-likeness of the input and the small subgraphs cover local connectivity patterns that may be far from being trees. A related notion is the *tree depth* of a graph; graphs with tree depth d have tree decompositions where not only the width is bounded in terms of d, but, in addition, the depth of the

underlying trees is bounded in terms of d. While tree width can be seen as measuring the similarity of graphs to trees (trees have tree width 1 and graphs with cycles have tree width at least 2), tree depth can be seen as measuring the similarity of graphs to star graphs (star graphs have tree depth 2, and the tree depth of a graph grows with the length of paths in it). We say that the tree width (tree depth) of a class of structures is *bounded* if there is a constant that upper-bounds their tree width (tree depth). Solving computational problems on structures of bounded tree width is often done by a two-step approach that computes a tree decomposition and, then, solves the problem by using a bottom-up dynamic programming approach along the tree.

For structures of bounded tree width, the polynomial-time bound of solving MSO-definable problems has been refined from the algorithmic point of view: sequential algorithms running in linear time [Courcelle 1990] and parallel algorithms running in log time [Bodlaender and Hagerup 1998] are known to solve MSO-definable decision problems for structures of bounded tree width and similar results hold for counting and optimization problems whose definition is based on MSO-formulas. A variety of results of a similar flavor have been developed during the last years and are commonly called *algorithmic meta theorems* [Grohe and Kreutzer 2011]: Instead of presenting an algorithmic result for some particular problem, these theorems state that "all problems of a certain kind on structures of a certain kind are solvable by an efficient algorithm of a certain kind". Courcelle's Theorem falls into this category since it shows that all MSO-definable problems for structures whose tree width is bounded are solvable in linear time.

Since many important problems are MSO-definable, Courcelle's Theorem and its variants yield unified frameworks for showing that numerous problems on structures of bounded tree width are efficiently solvable. Moreover, often algorithmic meta theorems show their real power when used as subroutines in algorithms solving problems that are normally (1) not MSO-definable, or (2) whose inputs are not tree-decomposable in an algorithmically useful way. A problem of the first kind is computing the chromatic number of a graph, the least number of colors needed for a valid coloring of a graph. Since graphs of tree width w have chromatic number at most $w + 1$, we can use modifications of the formula for 3-COLORABILITY to test whether the graph is c-colorable for a $c \leq w + 1$. A problem of the second kind is EVEN-CYCLE—deciding whether an undirected loop-free graph has a cycle of even length. This problem can be solved by first testing whether the tree width of the graph is higher than some constant. If this is the case, we can use a graph-theoretic insight of Thomassen [1988] showing that there is always an even cycle in such graphs and we answer "yes". If the tree width is bounded by a constant, we use Courcelle's Theorem to solve the problem via an MSO-formula that defines even-length cycles on the incidence representation of graphs (that means, logical structures for graphs where both vertices and edges are represented as individual elements and they are related by a binary incidence relation).

The algorithmic meta theorems mentioned above provide a unified framework for finding fast sequential and parallel algorithms to solve MSO-definable problems on tree-decomposable structures. I will describe some results that exactly characterize the complexity of families of MSO-definable problems. We already know that MSO-definable decision problems on structures of bounded tree width lie in P, but how deep inside P can we place them? Bodlaender [1989] and Wanke [1994] contributed important steps to clarify this question: Bodlaender showed that all problems that are covered by Courcelle's Theorem lie in NC (they can be solved by families of polylog-depth circuits) and Wanke showed that in fact they lie in $\mathrm{LOGCFL} = \mathrm{SAC}^1$ (they can be solved by families of log-depth circuits with unbounded fan-in $\vee$ gates but bounded fan-in $\wedge$-gates). This complexity class is sandwiched between NL (problems decidable by nondeterministic logspace Turing machines) and AC^1 (problems decidable by families of log-depth

circuits over Boolean gates of unbounded fan-in); that means, $\mathrm{NL} \subseteq \mathrm{SAC}^1 \subseteq \mathrm{AC}^1$. We review results that refine these insights in terms of complexity classes inside L (deterministic logspace).

Section 2 reviews results and techniques from [Elberfeld et al. 2010] that are related to algorithmic meta theorems for logspace DTMs. Sections 3 and 4 review results and techniques from [Elberfeld et al. 2012] about algorithmic meta theorems for circuit complexity classes that lie inside L. Section 5 concludes with a summary and an outlook on current and possible future work.

2. VARIANTS OF COURCELLE'S THEOREM FOR LOGARITHMIC SPACE

A key problem in the study of logspace DTMs is REACHABILITY, detecting whether there is a path from some start to some target vertex in a directed graph. While REACHABILITY was identified to be NL-complete for general directed graphs [Jones 1975], Reingold gave an L upper bound for the case of undirected graphs [Reingold 2008]. Paths and thus, reachability, are MSO-definable on the incidence representation of directed graphs. Questions about the complexity of reachability for classes of directed graphs motivated us to prove Theorem 2.1.

THEOREM 2.1. *For every $w \in \mathbb{N}$, and every* MSO*-formula φ, there is a logspace* DTM *that, on input a structure $\mathcal{A}$ of tree width at most w, decides whether $\mathcal{A} \models \varphi$.*

Since reachability is expressible in MSO, it follows from Theorem 2.1 that it is also in L for graphs of bounded tree width.

Another problem of great interest is PERFECT-MATCHING, detecting whether an undirected graph has a perfect matching. PERFECT-MATCHING is known to be in P and in fact even in Random NC (RNC), but it is not known to be in NC. In addition to deciding whether a graph has a perfect matching, counting the number of perfect matchings was identified as an important complexity problem by Valiant [1979].

To move from decision problems to counting and optimization problems, Elberfeld et al. [2012] developed the following methodology. Let $\varphi(X_1, \ldots, X_k, Y_1, \ldots, Y_\ell)$ be an MSO-formula with two sets of free set-variables, namely the X_i and the Y_j, and let $\mathcal{A}$ be a structure with universe A. The *solution histogram* of $\mathcal{A}$ and φ ($\mathrm{histogram}(\mathcal{A}, \varphi)$), is a k-dimensional integer array that tells us how many solutions of a certain size exist. In detail, let $s = (s_1, \ldots, s_k) \in \{0, \ldots, |A|\}^k$ be an index vector of sizes for the sets that are substituted for the X_i. Then $\mathrm{histogram}(\mathcal{A}, \varphi)[s]$ equals the number of $(S_1, \ldots, S_k, S'_1, \ldots, S'_\ell) \in \mathrm{POW}(A)^{k+\ell}$, such that $|S_1| = s_1$, ..., $|S_k| = s_k$ and $\mathcal{A} \models \varphi(S_1, \ldots, S_k, S'_1, \ldots, S'_\ell)$ hold. In other words, we count how often φ can be satisfied when the sets assigned to the X_i-variables have the specified sizes. (No restrictions are imposed on the sizes of the Y_j.)

For example, recall the formula $\varphi_{\text{dominates}}(X_1)$ (Eqn. 1), with $k = 1$ and $\ell = 0$. $\mathrm{histogram}(\mathcal{G}, \varphi_{\text{dominates}})[s]$ is the number of dominating sets of size s in the graph $\mathcal{G}$. For example, the histogram for the graph $\mathcal{G} =$  with respect to the formula $\varphi_{\text{dominates}}(X_1)$ is the following.

# of dominating sets	0	0	3	8	5	1
of size	0	1	2	3	4	5

(For example, $\mathcal{G}$ has no dominating sets of size 1, and 3 of size 2.)

As another example, consider an MSO-formula $\varphi_{\text{is-matching}}(Y_1)$ with $k = 0$ and $\ell = 1$ over the incidence representation of graphs that defines sets of edges that are matchings. That means, it states that every vertex in the graph is incident to at most one edge in Y_1. Then $\mathrm{histogram}(\mathcal{G}, \varphi_{\text{is-matching}})$ is just a scalar value that tells us how many matchings $\mathcal{G}$ contains. Similarly, one can state a formula $\varphi_{\text{is-perfect-matching}}(Y_1)$ that helps

to count the number of perfect matchings; that means, matchings that cover all vertices.

The following result generalizes Thm. 2.1 showing that not only are MSO decision problems in L for structures of bounded tree width, but the same is true for MSO counting problems:

THEOREM 2.2. *For every $w \in \mathbb{N}$, and every* MSO*-formula $\varphi(X_1, \ldots, X_k, Y_1, \ldots, Y_\ell)$, there is a logspace* DTM *that, on input a structure $\mathcal{A}$ of tree width at most w, outputs* $\mathrm{histogram}(\mathcal{A}, \varphi)$.

Theorem 2.2 has various applications. First of all, it can be applied to $\varphi_{\text{is-perfect-matching}}(Y_1)$ to show that counting the number of perfect matchings can be done in logspace for graphs of bounded tree width. Moreover, it follows that for graphs of bounded tree width, the optimization problem DOMINATING-SET is in L: on input, graph $\mathcal{G}$, we compute $\mathrm{histogram}(\mathcal{G}, \varphi_{\text{dominates}})$ and output the smallest nonzero position. Furthermore, we can determine, in logspace, whether there is a dominating set of a given size and we can determine the number of dominating sets of a given size.

Tree decompositions of approximate width in logspace. A technique developed for proving the theorems related to tree-width-bounded structures and logspace is the computation of tree decompositions of an approximate bounded width. That means, tree decompositions whose width is bounded by a linear function in the (exact) tree width of the given graph. Once such tree decompositions are constructed, one can use a translation from MSO-formulas on structures to automata on node-labeled trees and an arithmetic simulation approach for the automata to prove the above theorems. Computing tree decompositions of approximate width is also a step for removing the witness on the tree width bound from both theorems stated above, and test the tree width bound together with the MSO-defined property. That means, it is used to prove the following formal statement in [Elberfeld et al. 2010]:

LEMMA 2.3. *For every $w \in \mathbb{N}$, there is a logspace* DTM *that, on input a structure $\mathcal{A}$, either (1) outputs a width-w tree decomposition of $\mathcal{A}$, or (2) outputs "no" and the tree width of $\mathcal{A}$ exceeds w in this case.*

The proof of this combines techniques developed for computing approximate tree decomposition with calling MSO-definable problems, which are logspace-solvable due to the above theorems, as subroutines. As a consequence, we also know the following:

PROPOSITION 2.4. *For every $w \geq 1$, deciding whether a given graph has tree width at most w is* L*-complete.*

A different application of the approximate-width tree decompositions was recently developed in a work [Elberfeld and Schweitzer 2015] that studies the complexity of the isomorphism problem for graphs of bounded tree width. This problem was known to be in P due to a result of [Bodlaender 1990], but since then it remained an open question to completely classify the complexity with respect to circuit or space complexity classes (see the introduction of [Elberfeld and Schweitzer 2015] for details on the problem's motivation and history). The above mentioned work uses the framework for computing approximate tree decompositions as part of a logspace procedure for graphs of bounded tree width with the final consequence that the graph isomorphism problem restricted to graphs of tree width w is proven to be L-complete for each $w \geq 1$.

3. VARIANTS OF COURCELLE'S THEOREM FOR LOG-DEPTH CIRCUITS

Restricted to trees, reachability is complete for L [Cook and McKenzie 1987]. In fact, many MSO-definable problems on structures of bounded tree width are complete for L.

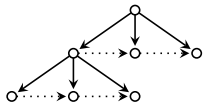
The following meta theorem shows that when the input is given via an explicit width-w tree decomposition in a format we call "term representation", MSO problems are doable in the circuit complexity class NC^1.

THEOREM 3.1 ([Elberfeld et al. 2012]). *For every $w \in \mathbb{N}$, and every* MSO*-formula φ, there is an NC^1-circuit family accepting the set of structures satisfying φ, when the inputs are given with a width-w tree decomposition in term-representation form.*

In order to to prove a histogram-based theorem for log-depth circuits, we use arithmetic circuits instead of Boolean circuits. The inputs to these circuits are (binary) strings, while their inner gates (of bounded fan-in) compute addition and multiplication and the outputs are integers. Families of log-depth circuits of this kind define the counting complexity class $\#\mathrm{NC}^1$. (A good source for information about $\#\mathrm{NC}^1$ is [Vollmer 1999].) In order to represent the output histogram h using a single number $\mathrm{num}(h) \in \mathbb{N}$, consider h to be stored in consecutive memory cells with each cell being large enough to store a single entry. Then $\mathrm{num}(h)$ is the single number that equals the content of all cells when reading them from left to right. Theorem 3.1 is a corollary of the following theorem since testing whether a function from $\#\mathrm{NC}^1$ outputs a value greater than 0 is an NC^1-computable property (this can be seen by replacing addition and multiplication gates of the arithmetic circuits with Boolean $\vee$- and $\wedge$-gates, respectively).

THEOREM 3.2 ([Elberfeld et al. 2012]). *For every $w \in \mathbb{N}$, and every* MSO*-formula $\varphi(X_1, \ldots, X_k, Y_1, \ldots, Y_\ell)$, there is a $\#\mathrm{NC}^1$-circuit family that, on input a structure $\mathcal{A}$ with its width-w tree decomposition in term-representation form, outputs* $\mathrm{histogram}(\mathcal{A}, \varphi)$.

The above two theorems can be applied when inputs are already accompanied by tree decompositions, or when these decompositions can be automatically derived. Many *evaluation* and *simulation* tasks are of the later kind, e.g., evaluating Boolean and arithmetic sentences, as well as simulating the acceptance behavior and counting the number of accepting computations of visibly pushdown automata. By encoding the problems of evaluating Boolean and arithmetic sentences as MSO-defined decision and counting problems, respectively, we see that Theorems 3.1 and 3.2 apply to problems that are complete for NC^1 and $\#\mathrm{NC}^1$, respectively.



An interesting application of the above theorems is to show how to evaluate any fixed tree automaton for unranked ordered trees in NC^1, and count the number of accepting runs in $\#\mathrm{NC}^1$. Automata on such trees, like the one shown on the right, determine a state for each node based (1) on the label of the node, and (2) the outcome of the computation of a finite automaton that works on the sequence of states assigned to the children of the node. Such tree automata are used to model specifications of XML documents [Gottlob et al. 2005] and, hence, the input is actually a string like [[[ ] [ ] [ ]] [ ] [ ]], which encodes our example tree. The NC^1-upper bound for deciding the acceptance behavior of any fixed tree automaton on such inputs was originally shown by Gottlob et al. [2005]. This result can also be obtained from the above theorems by translating the input string into a logical structure along with a tree decomposition in term representations, and using an MSO-formula to define the automaton's behavior. This also gives a $\#\mathrm{NC}^1$-upper bound for counting the number of accepting computations.

Balancing Tree Decompositions Using Constant-Depth Circuits. For proving Theorems 3.1, 3.2, no tree decompositions are constructed, but the given tree decompositions are balanced: The processed tree may have a linear depth and, thus, may contain data dependencies whose direct resolution would need the same linear circuit depth.

In the case of Theorem 3.1, we can translate the MSO-formula that defines the problem into a tree automaton and plug in a result showing that the acceptance behavior of tree automata can be simulated using log-depth Boolean circuits [Lohrey 2001]. To prove Theorem 3.2, we proceed as follows: Before translating the formula on the structure into a tree automaton, an extra layer is inserted into the proof that transforms the given tree decomposition into a tree decomposition whose underlying tree is binary and balanced and, thus, has log depth. Interestingly, for this step constant-depth Boolean circuits with threshold gates are enough to balance the tree decomposition. Evaluating tree automata can then be done with log-depth circuits for the resulting log-depth decomposition.

4. VARIANTS OF COURCELLE'S THEOREM FOR CONSTANT-DEPTH CIRCUITS

For *structures of bounded tree depth*, the complexity of MSO-definable problems drops to constant-depth circuit classes. In order to solve MSO-definable decision, counting, and optimization problems on tree-depth-bounded structures, different kinds of constant-depth circuits are used: Boolean circuits for solving decision problems, arithmetic circuits for solving counting problems, and Boolean circuits with threshold gates for solving optimization problems. Depending on its type, a problem lies in one of the unbounded-fanin, bounded-depth circuit classes: AC^0 (Boolean gates), GapAC^0 (arithmetic $\{+,-,\cdot\}$-gates) or TC^0 (threshold gates). We start with MSO-definable decision problems.

THEOREM 4.1. *For every $d \in \mathbb{N}$, and every* MSO*-formula φ, there is an AC^0-circuit family that, on input a structure $\mathcal{A}$ of tree depth at most d, decides whether $\mathcal{A} \models \varphi$.*

An example application of Theorem 4.1 is to put the MSO-definable decision problem of whether a graph of bounded tree depth has a perfect matching, into AC^0. In contrast, the same problem for graphs of bounded tree width is L-complete. In case of input structures of bounded tree depth, we can state the following theorem for counting problems:

THEOREM 4.2 ([Elberfeld et al. 2010]). *For every $d \in \mathbb{N}$, and every* MSO*-formula $\varphi(X_1, \ldots, X_k, Y_1, \ldots, Y_\ell)$, there is a GapAC^0-circuit family that, on input a structure $\mathcal{A}$ of tree depth at most d, outputs $\mathrm{histogram}(\mathcal{A}, \varphi)$ as a sequence of numbers.*

Applying Theorem 4.2 to a formula that defines perfect matchings on the incidence representation of graphs (as described above) proves that we can count the number of perfect matchings of tree-depth-bounded graphs in GapAC^0. As in the logspace case, the solution histogram subsumes many optimization problems that ask for the existence of a solution of a certain size. For this, we need to look up individual bits in the string representation of the solution histogram. Using the result of Hesse et al. [2002] that $\mathrm{DLOGTIME}$-uniform GapAC^0-circuit families, which compute numbers, can be simulated by $\mathrm{DLOGTIME}$-uniform functional TC^0-circuit families that compute the binary string representations of these numbers, we derive the following as a corollary of Theorem 4.2.

THEOREM 4.3. *For every $d \in \mathbb{N}$, and every* MSO*-formula $\varphi(X_1, \ldots, X_k, Y_1, \ldots, Y_\ell)$, there is a TC^0-circuit family that, on input a structure $\mathcal{A}$ of tree depth at most d, outputs $\mathrm{histogram}(\mathcal{A}, \varphi)$ as a string.*

Like Theorem 2.2, which concerns computing histograms in logspace, Theorem 4.3 can be used to solve a wide range of problems: Elberfeld et al. [2012] apply it to show that the problems SUBSETSUM and KNAPSACK with input numbers that are encoded in unary lie in TC^0. The quest to determine the complexity of this kind of problem has a long history: If the input numbers are encoded in binary, SUBSETSUM is known

to be NP-complete. It becomes polynomial-time solvable if the numbers are encoded in unary; in fact, it can be shown to lie in NL in this setting via solving a reachability problem related to dynamic programming tables. Inspired by a conjecture of Cook [1985] that "a problem in NL which is probably not complete is the knapsack problem with unary weights," a line of research began to capture the complexity of SUBSETSUM and KNAPSACK with unary weights using specialized complexity classes lying between L and NL [Monien 1980; Ibarra et al. 1988; Cho and Huynh 1988; Jenner 1995]. Using Theorem 4.3 it is possible to show that SUBSETSUM is TC^0-complete: map an instance with input numbers a_1 to a_n and b encoded in unary as $1^{a_1}0\,1^{a_2}0\ldots1^{a_n}00\,1^b$ of SUBSETSUM to a forest $\mathcal{F} = (V, E^{\mathcal{F}})$ consisting of n stars where the ith star has a_i vertices, and use an MSO-formula $\varphi(S)$ that forces solution sets $S \subseteq V$ to cover each star either completely or not at all. Since such forests have bounded tree depth, applying Theorem 4.3 puts SUBSETSUM with unary weights into TC^0; the input $1^{a_1}0\,1^{a_1}0\ldots1^{a_n}00\,1^b$ has a solution exactly if $\mathrm{histogram}(\mathcal{F}, \varphi)[b] > 0$. A similar idea applies to KNAPSACK with unary weights and related number problems.

Arithmetic-Circuit-Based Evaluation of Automata for Unbounded Degree Trees. For the proof of Theorem 4.3, tree decompositions of bounded width, whose underlying trees have bounded depth, are constructed. Once such tree decompositions are available, the proof of Theorem 4.2 proceeds to evaluate the MSO-formula from the problem definition along the tree decomposition, which is done by (1) transforming the formula on the input structure into an equivalent tree automaton on the tree decomposition, and (2) evaluating the tree automaton using an arithmetic circuit. This basic structure is often used to prove Courcelle's Theorem and its variants, but for implementing it in constant-depth, additional techniques needed to be developed. Commonly, proofs of Courcelle's Theorem and its variants are based on translations into automata on degree-bounded trees. The nodes of the depth-bounded trees underlying the tree decompositions that are involved in the proof of Theorem 4.2 necessarily have an unbounded degree since, otherwise, they would only be able to cover constant-size structures. The main step in proving Theorem 4.2 is the development of an automaton model for unbounded-degree labeled trees that can simulate MSO-formulas and whose computations can be represented algebraically using arithmetic circuits.

5. CONCLUSION

Summary. This column reviewed variants of Courcelle's theorem for complexity classes inside P. That means, variants where the "linear-time" resource bound is replaced by other bounds (like logspace or bounds on circuit families) and "bounded tree width" is kept or replaced by other conditions (like given tree decompositions of bounded width in term representation or bounded tree depth). In each case, we reviewed results that can be seen as tailoring Courcelle's theorem to be used for a certain complexity class. Figure 1 gives an overview of the theorems, the ideas for their proofs, and their range of applications.

Outlook. Theorem 2.1 shows that all MSO problems are in logspace for inputs of bounded tree width. This contrasts with Courcelle's Theorem which says that these problems are in linear time, but potentially requiring linear space. The runtime for the logspace algorithms provided by Theorem 2.1 is typically far from linear (the current proof provides a polynomial runtime where the polynomial degree grows with a linear function in the tree width). It would be extremely interesting to find algorithms that have both a low memory footprint and a polynomial runtime with a low (maybe even fixed) polynomial degree; the main motivation for this question comes from the wide range of applications of MSO-definable problems.

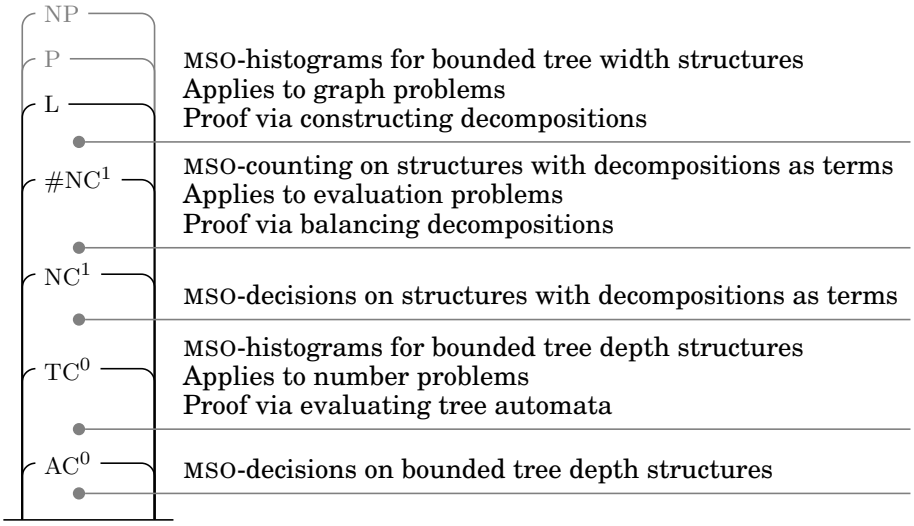


Fig. 1. The algorithmic meta theorems described apply to MSO-definable decision problems (MSO-decisions) or to MSO-definable histograms (MSO-histograms). For the latter it is important to observe the difference between histograms encoded as strings and histograms encoded as number.

One of the applications of the algorithmic meta theorem for logspace is constructing tree decompositions of every constant width $w \in \mathbb{N}$ in logspace, which implies that the problem of deciding whether a graph has tree width at most w is L-complete for every $w \geq 1$. A similar result was recently obtained for the related question of finding graph embeddings with a constant Euler genus. It thus follows that the problem of deciding whether a graph has Euler genus at most g is L-complete for every $g \in \mathbb{N}$ [Elberfeld and Kawarabayashi 2014]. It would be interesting to generalize these insights to more general classes of graphs, for example, graphs that exclude a fixed minor. It is known that the problem of deciding whether a graph does not contain H as a minor can be solved in polynomial time for every fixed H [Robertson and Seymour 1995]. In the light of the above results, it seems plausible that this upper bound might be lowered to a complexity class inside P; perhaps even L.

REFERENCES

Hans L. Bodlaender. 1989. NC-algorithms for graphs with small treewidth. In *Proceedings of the 14th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 1988) (Lecture Notes in Computer Science)*, Vol. 344. Springer, 1–10. DOI: http://dx.doi.org/10.1007/3-540-50728-0_32

Hans L. Bodlaender. 1990. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms* 11, 4 (1990), 631–643. DOI: http://dx.doi.org/10.1016/0196-6774(90)90013-5

Hans L. Bodlaender and Torben Hagerup. 1998. Parallel Algorithms with Optimal Speedup for Bounded Treewidth. *SIAM J. Comput.* 27, 6 (1998), 1725–1746. DOI: http://dx.doi.org/10.1137/S0097539795289859

Sang Cho and Dung T. Huynh. 1988. On a complexity hierarchy between L and NL. *Inform. Process. Lett.* 29, 4 (1988), 177–182. DOI: http://dx.doi.org/10.1016/0020-0190(88)90057-9

Stephen A. Cook. 1985. A taxonomy of problems with fast parallel algorithms. *Information and Control* 64, 1–3 (1985), 2–22. DOI: http://dx.doi.org/10.1016/S0019-9958(85)80041-3

Stephen A. Cook and Pierre McKenzie. 1987. Problems complete for deterministic logarithmic space. *Journal of Algorithms* 8, 5 (1987), 385–394. DOI:http://dx.doi.org/10.1016/0196-6774(87)90018-6

Bruno Courcelle. 1990. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, Jan van Leeuwen (Ed.). Elsevier and MIT Press, 193–242.

Michael Elberfeld. 2012. *Space and Circuit Complexity of Monadic Second-Order Definable Problems on Tree-Decomposable Structures*. Universitt zu Lübeck. Dissertation.

Michael Elberfeld, Andreas Jakoby, and Till Tantau. 2010. Logspace Versions of the Theorems of Bodlaender and Courcelle. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010)*. IEEE, 143–152. DOI:http://dx.doi.org/10.1109/FOCS.2010.21

Michael Elberfeld, Andreas Jakoby, and Till Tantau. 2012. Algorithmic Meta Theorems for Circuit Classes of Constant and Logarithmic Depth. In *Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012) (Leibniz International Proceedings in Informatics)*, Vol. 14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 66–77. DOI:http://dx.doi.org/10.4230/LIPIcs.STACS.2012.66

Michael Elberfeld and Ken-ichi Kawarabayashi. 2014. Embedding and Canonizing Graphs of Bounded Genus in Logspace. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*. ACM, New York, NY, USA, 383–392. DOI:http://dx.doi.org/10.1145/2591796.2591865

Michael Elberfeld and Pascal Schweitzer. 2015. Canonizing Graphs of Bounded Tree Width in Logspace. In *Proceedings of the 33nd International Symposium on Theoretical Aspects of Computer Science (STACS 2016) (Leibniz International Proceedings in Informatics)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. to appear.

Jörg Flum and Martin Grohe. 2006. *Parameterized Complexity Theory*. Springer, Berlin Heidelberg. DOI:http://dx.doi.org/10.1007/3-540-29953-X

Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman.

Georg Gottlob, Christoph Koch, Reinhard Pichler, and Luc Segoufin. 2005. The complexity of XPath query evaluation and XML typing. *Journal of the ACM* 52, 2 (2005), 284–335. DOI:http://dx.doi.org/10.1145/1059513.1059520

Martin Grohe and Stephan Kreutzer. 2011. Methods for Algorithmic Meta Theorems. In *Model Theoretic Methods in Finite Combinatorics*. Contemporary Mathematics, Vol. 558. American Mathematical Society, 181–206. https://www.lii.rwth-aachen.de/images/Mitarbeiter/pub/grohe/grokre11.pdf

William Hesse, Eric Allender, and David A. Mix Barrington. 2002. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. System Sci.* 65, 4 (2002), 695–716. DOI:http://dx.doi.org/10.1016/S0022-0000(02)00025-9

Oscar H. Ibarra, Tao Jiang, Bala Ravikumar, and Jik H. Chang. 1988. On some languages in NC^1. In *Proceedings of the Aegean Workshop on Computing: 3rd International Workshop on Parallel Computation and VLSI Theory*. Lecture Notes in Computer Science, Vol. 319. Springer, 64–73. DOI:http://dx.doi.org/10.1007/BFb0040374

Birgit Jenner. 1995. Knapsack problems for NL. *Inform. Process. Lett.* 54, 3 (1995), 169–174. DOI:http://dx.doi.org/10.1016/0020-0190(95)00017-7

Neil D. Jones. 1975. Space-bounded reducibility among combinatorial problems. *J. Comput. System Sci.* 11, 1 (1975), 68–85. DOI:http://dx.doi.org/10.1016/S0022-0000(75)80050-X

Markus Lohrey. 2001. On the Parallel Complexity of Tree Automata. In *Proceedings of 12th International Conference on Rewriting Techniques and Applications (RTA 2001) (Lecture Notes in Computer Science)*, Vol. 2051. Springer, 201–215. DOI:http://dx.doi.org/10.1007/3-540-45127-7_16

Burkhard Monien. 1980. On a subclass of pseudopolynomial problems. In *Proceedings of the 9th Symposium on Mathematical Foundations of Computer Science (MFCS 1980) (Lecture Notes in Computer Science)*, Vol. 88. Springer, 414–425. DOI:http://dx.doi.org/10.1007/BFb0022521

Omer Reingold. 2008. Undirected connectivity in log-space. *Journal of the ACM* 55, 4 (2008), 1–24. DOI:http://dx.doi.org/10.1145/1391289.1391291

N. Robertson and P. D. Seymour. 1995. Graph Minors. XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B* 63, 1 (1995), 65–110. DOI:http://dx.doi.org/10.1006/jctb.1995.1006

Carsten Thomassen. 1988. On the presence of disjoint subgraphs of a specified type. *Journal of Graph Theory* 12, 1 (1988), 101–111. DOI:http://dx.doi.org/10.1002/jgt.3190120111

Leslie G. Valiant. 1979. The complexity of computing the permanent. *Theoretical Computer Science* 8, 2 (1979), 189–201. DOI:http://dx.doi.org/10.1016/0304-3975(79)90044-6

Heribert Vollmer. 1999. *Introduction to Circuit Complexity: A Uniform Approach*. Springer, Berlin Heidelberg.

Egon Wanke. 1994. Bounded tree-width and LOGCFL. *Journal of Algorithms* 16, 3 (1994), 470–491. DOI: http://dx.doi.org/10.1006/jagm.1994.1022