Algorithmic Meta Theorems for Circuit Classes of Constant and Logarithmic Depth

Michael Elberfeld, Andreas Jakoby, and Till Tantau

Institut für Theoretische Informatik, Universität zu Lübeck
D-23538 Lübeck, Germany
`{elberfeld,jakoby,tantau}@tcs.uni-luebeck.de`

Abstract

An algorithmic meta theorem for a logic and a class C of structures states that all problems expressible in this logic can be solved efficiently for inputs from C. The prime example is Courcelle's Theorem, which states that monadic second-order (MSO) definable problems are linear-time solvable on graphs of bounded tree width. We contribute new algorithmic meta theorems, which state that MSO-definable problems are (a) solvable by uniform constant-depth circuit families (AC^0 for decision problems and TC^0 for counting problems) when restricted to input structures of bounded tree depth and (b) solvable by uniform logarithmic-depth circuit families (NC^1 for decision problems and $\#\mathrm{NC}^1$ for counting problems) when a tree decomposition of bounded width in term representation is part of the input. Applications of our theorems include a TC^0-completeness proof for the unary version of integer linear programming with a fixed number of equations and extensions of a recent result that counting the number of accepting paths of a visible pushdown automaton lies in $\#\mathrm{NC}^1$. Our main technical contributions are a new tree automata model for unordered, unranked, labeled trees; a method for representing the tree automata's computations algebraically using convolution circuits; and a lemma on computing balanced width-3 tree decompositions of trees in TC^0, which encapsulates most of the technical difficulties surrounding earlier results connecting tree automata and NC^1.

1998 ACM Subject Classification F.1.3

Keywords and phrases algorithmic meta theorem, monadic second-order logic, circuit complexity, tree width, tree depth

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Courcelle's Theorem [6] states that every monadic second-order (MSO) definable problem can be decided in linear time on graphs of bounded tree width. Since many important graph properties are expressible in this logic, Courcelle's Theorem yields a unified framework for showing that numerous problems on graphs of bounded tree width are solvable in linear time. Recently we showed that both Courcelle's Theorem as well as its later extensions [1] also hold when "linear time" is replaced by "logarithmic space" [9], making the power of MSO-definability available for the study of logarithmic space.

The present paper furthers this line of research and transfers the idea of unified MSO-based problem definitions to circuit classes inside logarithmic space. During the course of this paper we identify MSO-based algorithmic meta theorems that place problems in the circuit classes AC^0, GapAC^0, TC^0, NC^1, and $\#\mathrm{NC}^1$. The classes AC^0, GapAC^0, and TC^0 are defined via Boolean (AC^0), arithmetic (GapAC^0), and threshold (TC^0) circuit families of constant depth and unbounded fan-in. The classes NC^1 and $\#\mathrm{NC}^1$ are defined via Boolean and arithmetic circuits, respectively, of logarithmic depth and bounded fan-in. All our results concerning circuit classes hold for a strict form of uniformity, namely DLOGTIME-uniformity.

The inputs for Courcelle's Theorem are graphs of bounded tree width and many MSO-definable problems on such graphs are complete for logarithmic space, including even the question of whether the graph has a certain tree width [9], but also the reachability problem for trees. Thus, algorithmic meta theorems that place problems inside subclasses of L either need to restrict the logic or the kinds of inputs allowed. In the present paper, we consider the latter case: For the constant-depth circuit classes, we only allow input graphs that have *bounded tree depth* (a restriction of bounded tree width). For the logarithmic-depth circuit classes we allow arbitrary graphs of bounded tree width as input, but require that the graphs are *accompanied by tree decompositions in term representation.*

Bounded Tree Depth Structures and Constant-Depth Circuits Our first contribution is a set of meta theorems that place problems in constant-depth circuit classes. The inputs for these theorems are structures that have bounded tree *depth*, a measure on graphs that was introduced by Nešetřil and Ossona de Mendez [16] to quantify the similarity of graphs to star graphs (in opposition to tree width, which quantifies the similarity of graphs to trees). Characterizations of when a class C of graphs has bounded tree depth include: (a) All graphs in C have a tree decomposition of both bounded width and depth; or alternatively (b) all graphs in C have bounded longest path length. The tree depth of a logical structure is the tree depth of its Gaifman graph.

▶ **Theorem 1** (Decision Using Boolean Constant-Depth Circuits). *For every* MSO*-formula ϕ over some signature τ and every $d \in \mathbb{N}$, there is a* DLOGTIME*-uniform* AC^0*-circuit family that, on input of an arbitrary τ-structure $\mathcal{S}$, outputs* 1 *if, and only if, the tree depth of $\mathcal{S}$ is at most d and $\mathcal{S} \models \phi$ holds.*

As an example application, consider the problem of deciding whether a graph has a perfect matching. The complexity of this MSO-definable problem has been studied in detail and its complexity varies in dependence of the class of graphs under consideration. By the above theorem, deciding whether a graph of bounded tree depth has a perfect matching lies in AC^0. In contrast, it is known that the same problem for graphs of bounded tree *width* is L-complete [7, 9].

Instead of just deciding whether a formula is satisfied by a logical structure, when the formula has a free second-order variable, we can try to count the number of assignments of sets to the free variable that make the formula true. Moreover, if we count the number of solutions with respect to the sizes of these sets, this leads to *cardinality versions* of Courcelle's Theorem. These cardinality versions allow a much broader range of applications than the decision version and we will show how both known results from the literature and also new results can be proved in an elegant manner using these versions. To formulate the cardinality versions, we need a bit of terminology: Let $\phi(X_1, \dots, X_\ell, Y_1, \dots, Y_k)$ be an MSO-formula with two sets of free set variables, namely the X_i and the Y_j, and let $\mathcal{S}$ be a logical structure with universe S. The *solution histogram* of ϕ and $\mathcal{S}$, denoted by $\mathrm{hist}(\mathcal{S}, \phi)$, is an ℓ-dimensional integer array that tells us "how many solutions of a certain size exist". In detail, let $s = (s_1, \dots, s_\ell) \in \{0, \dots, |S|\}^\ell$ be an index vector that prescribes sizes for the sets that are substituted for the X_i. Then $\mathrm{hist}(\mathcal{S}, \phi)[s]$ equals the number of subsets $S_1, \dots, S_\ell, S'_1, \dots, S'_k \subseteq S$ with $|S_1| = s_1, \dots, |S_\ell| = s_\ell$ and $\mathcal{S} \models \phi(S_1, \dots, S_\ell, S'_1, \dots, S'_k)$. In other words, we count how often ϕ can be satisfied when the sets assigned to the X_i-variables have certain sizes, but impose no restrictions on the sizes of the Y_j. As a first example, let

$\phi_{\text{dom}}(X_1) = \forall x\big(X_1(x) \lor \exists y(X_1(y) \land E(y,x))\big)$, which expresses that X_1 is a dominating set in a graph with edge relation E. Then $\text{hist}(\mathcal{G}, \phi_{\text{dom}})[s_1]$ is the number of dominating sets of size s_1 in the graph $\mathcal{G}$. As a second example, let $\phi_{\text{matching}}(Y_1)$ be the formula expressing that Y_1 is an edge set that is a perfect matching in G. Then, since $\ell = 0$, the histogram $\text{hist}(\mathcal{G}, \phi_{\text{matching}})$ is just a scalar value that tells us how many perfect matchings G has.

In order to represent a histogram h using a single number $\text{num}(h) \in \mathbb{N}$, imagine h to be stored in computer memory with a word size large enough so that each of its entries fits into one memory cell (choosing the word size as $k|S|$ will suffice). Then $\text{num}(h)$ is the single number representing the whole of the memory contents (a formal definition of $\text{num}(h)$ will be given later). In particular, the bits of any single entry of h can easily be obtained from the bits of $\text{num}(h)$.

▶ **Theorem 2** (Histogram Computation Using Arithmetic Constant-Depth Circuits). *For every* MSO*-formula $\phi(X_1, \dots, X_\ell, Y_1, \dots, Y_k)$ over some signature τ and every $d \in \mathbb{N}$, there is a* DLOGTIME*-uniform* GapAC^0*-circuit family that, on input of a τ-structure $\mathcal{S}$ of tree depth at most d, outputs* $\text{num}(\text{hist}(\mathcal{S}, \phi))$.

By Theorem 1 we can check in AC^0 whether an input structure $\mathcal{S}$ has tree depth d and, if not, we could output an error value like -1. Applying Theorem 2 to the formula $\text{hist}(\mathcal{G}, \phi_{\text{matching}})$ shows that counting the number of perfect matchings in graphs of bounded tree depths lies in GapAC^0. Since GapAC^0 is contained in FTC^0, the functional version of the class TC^0 of constant-depth circuits with threshold gates, computing a particular bit of the number $\text{num}(\text{hist}(\mathcal{S}, \phi))$ can be done using a TC^0-circuit:

▶ **Corollary 3.** *For every* MSO*-formula $\phi(X_1, \dots, X_\ell, Y_1, \dots, Y_k)$ over some signature τ and every $d \in \mathbb{N}$, there is a* DLOGTIME*-uniform* TC^0*-circuit family that, on input of a τ-structure $\mathcal{S}$ of tree depth at most d, an ℓ-dimensional index s, and a bit position i, outputs the ith bit of* $\text{hist}(\mathcal{S}, \phi)[s]$.

We cannot hope to place the computation of solution histograms in any complexity class smaller than FTC^0 since the TC^0-complete problem MAJORITY is easily expressible using an MSO-formula and the histogram: Turning a string s into a logical structure $\mathcal{S} = (\{1, \dots, |s|\}, P_1^{\mathcal{S}})$ in the usual manner by setting $i \in P_1^{\mathcal{S}} \Leftrightarrow s[i] = 1$, for the MSO-formula $\phi(X_1) = \forall x(X_1(x) \to P_1(x))$ more than half of the input bits are 1 if, and only if, $\text{hist}(\mathcal{S}, \phi)\big[\lfloor |s|/2 \rfloor + 1\big] > 0$. In Section 3 we use extensions of this idea to prove the TC^0-completeness of the unary version of integer linear programming with a constant number of equations.

Bounded Tree Width, Term Representations, and Logarithmic-Depth Circuits Our second contribution are algorithmic meta theorems for NC^1 and its arithmetic companion class $\#\text{NC}^1$. For these theorems the input structure is equipped with a tree decomposition of bounded width (no longer of bounded depth, though) given in term representation. The term representation of a tree like is the string [[ ] [[ ] [ ]]], which exhibits the tree's ancestor relation.

▶ **Theorem 4** (Decision Using Boolean Logarithmic-Depth Circuits). *For every* MSO*-formula ϕ over some signature τ and every $w \in \mathbb{N}$, there is a* DLOGTIME*-uniform* NC^1*-circuit family that, on input of a τ-structure $\mathcal{S}$ along with a width-w tree decomposition in term representation for $\mathcal{S}$, decides whether $\mathcal{S} \models \phi$ holds.*

As an example application, consider the problem of deciding the language accepted by a tree automaton. It is well known that every such language lies in NC^1 [15]. The above theorem allows us to reprove this fact succinctly: an MSO-formula can easily check (using existential second-order quantifiers) whether there is an assignment of states to the nodes of the tree that is locally consistent and that makes the automaton accept.

▶ **Theorem 5** (Histogram Computation Using Arithmetic Logarithmic-Depth Circuits). *For every* MSO*-formula $\phi(X_1, \dots, X_\ell, Y_1, \dots, Y_k)$ over some signature τ and every $w \in \mathbb{N}$, there is a* DLOGTIME*-uniform $\#\mathrm{NC}^1$-circuit family that, on input of a τ-structure $\mathcal{S}$ along with a width-w tree decomposition in term representation for $\mathcal{S}$, outputs* $\mathrm{num}(\mathrm{hist}(\mathcal{S}, \phi))$.

An application of this theorem is to count the number of accepting paths of nondeterministic visible pushdown automata.

Technical Contributions The proofs of the algorithmic meta theorems for constant-depth circuits rest on two new technical tools. First, we introduce a new model of automata, which we call *multiset automata*, that exactly captures the MSO-definable problems on unordered unranked labeled trees. Standard automata-theoretic approaches to proving meta theorems cannot be applied in the context of constant-depth circuits: all known approaches include preprocessing steps that enlarge the depth of the input trees by at least a logarithmic factor, making them infeasible for simulation by constant-depth circuits. Second, we develop algebraic representations of the computations of multiset automata using arithmetic circuits that keep track of the number of ways in which states can be reached.

In the context of research on logarithmic-depth circuits, trees in term representation are a natural form of input. In many papers (including the present), a central problem is that a logarithmic-depth circuit cannot work on the term representation directly when it has large depth. The standard approach is to recursively divide the tree into parts smaller by some constant factor, but doing so uniformly is an involved problem. We present a new algorithm for dealing with trees of arbitrary depth: It takes a tree T as input and outputs a width-3 tree decomposition of T that is perfectly balanced and, hence, has logarithmic depth. The bags of this decomposition form a hierarchical separation of T into subtrees along which a recursive algorithm can work. A key property of our construction is that it can be performed in TC^0.

Related Work Algorithmic meta theorems for monadic second-order logic have been studied intensively from the perspective of achieving a low runtime (see [12] for an overview), but there is less work on meta theorems that lead to exact classifications in complexity theoretic terms. Two exceptions are Wanke's paper [18], which shows that all problems that are captured by Courcelle's Theorem are in LOGCFL, and our paper [9], which places these problems further down into L.

Tree automata-based techniques are routinely used to prove time- and space-efficient variants of Courcelle's Theorem [1, 9]. The problem of deciding whether a fixed tree automaton accepts a given tree in term representation lies in NC^1 both in the ranked [15] and the unranked case [11].

Buss [3] used pebbling-based strategies to evaluate Boolean sentences in uniform NC^1. His method was later adopted to evaluate arithmetic sentences [2] and, more recently, to prove that the number of accepting computations of nondeterministic visible pushdown automata can be counted in $\#\mathrm{NC}^1$ [13].

Organization of This Paper After discussing the logical, graph theoretic and complexity theoretic background of our work in Section 2, in Sections 3 and 4 we sketch the proofs and applications of the algorithmic meta theorems for constant-depth and logarithmic-depth circuits, respectively. Due to lack of space, formal proofs are omitted from the present conference paper; they can be found in its technical report version [10].

2 Background

A detailed review of the notations on logic, graphs, tree decompositions, and complexity classes that we use in the present paper can be found in our technical report [10]. In the following, we just point out less common notations that we will use: For a graph G, we write $V(G)$ for its vertex set and $E(G) \subseteq V(G) \times V(G)$ for its edge set. We consider trees as special cases of directed graphs, where edges point from the root towards the leaves, but call their vertices *nodes*. A tree decomposition is a pair (T_D, B_D) where T_D is a tree and B_D is a labeling function $B_D \colon V(T_D) \to \mathcal{P}(V(G))$, where $\mathcal{P}(X)$ is the power set of X, that satisfies standard connectedness and edge covering conditions. The *closure* $\operatorname{clos}(T)$ of a directed tree T is the graph with vertex set $V(\operatorname{clos}(T)) = V(T)$ and edge set $E(\operatorname{clos}(T)) = \{(v, w) \in V(T) \times V(T) \mid \text{there is a } v\text{-to-}w \text{ or a } w\text{-to-}v \text{ path in } T\}$. The *tree depth* [16] of a connected graph G, denoted by $\operatorname{td}(G)$, is 1 plus the minimum depth of a rooted tree T with $V(G) = V(T)$ and $E(G) \subseteq E(\operatorname{clos}(T))$. The tree depth of a graph with components $C_1, \dots, C_m$ is $\max_{i \in \{1,\dots,m\}} \operatorname{td}(C_i)$. The tree width and tree depth of a logical structure are those of its Gaifman graph. The *longest path length* $\operatorname{lpl}(G)$ of a graph is the length of the longest path in the graph. Nešetřil and Ossona de Mendez [17] showed that $\operatorname{lpl}(G) \leq 2^{\operatorname{td}(G)} - 2$ holds for each undirected graph G.

3 Algorithmic Meta Theorems For Constant-Depth Circuit Classes

In the present section we prove the algorithmic meta theorems that relate monadic second-order properties of graphs of bounded tree depth to constant-depth circuit classes (Theorems 1 and 2 from the introduction). The route toward proving them is the following: (1) First, we show how a tree decomposition of a logical structure of bounded tree depth can be computed using first-order reductions. Once available, we show how to adjust the original MSO-formula to an equivalent formula for the computed tree. This first step allows us to replace the task of computing solution histograms for structures of any signature by the more manageable problem of computing solution histograms for trees. (2) Second, we introduce the notion of multiset automata for unordered unranked labeled trees, prove standard closure properties for these automata, and show that they capture exactly the MSO-definable properties of unordered unranked labeled trees. This turns the problem of deciding formulas into the problem of evaluating multiset automata. (3) After that we explain how to reduce computing the number of ways in which multiset tree automata accept an input tree to evaluating arithmetic circuits of constant depth. In the course of this step, we address the problem of how histograms can be encoded as numbers. As we will see, by using an appropriate encoding, we may assume that our formulas ϕ are all of the form $\phi(X_1, \dots, X_k)$, that is, we may assume that no variables Y_i are present. This is why the lemmas and theorems of the present section are all formulated without references to any Y_j. (4) At the end, we apply the algorithmic meta theorems to concrete problems. We show, in particular, that the unary version of integer linear programming with a constant number of equations is complete for TC^0.

Turning Tree-Depth-Bounded Structures into Depth-Bounded Tree Structures The first step toward our goal of proving Theorems 1 and 2 is to compute tree decompositions of bounded width and depth for input structures of bounded tree depth using first-order reductions.

▶ **Lemma 6.** *Let τ be a signature and $d \in \mathbb{N}$. There is a first-order computable function that, on input of the encoding $\operatorname{code}(\mathcal{S})$ of a τ-structure $\mathcal{S}$, outputs either (a) a tree decomposition D of $\mathcal{S}$ of width at most $2^d - 3$ and depth at most $2^d - 1$, or (b) "no" and $\operatorname{td}(\mathcal{S}) > d$ holds in this case.*

The following lemma uses a first-order reduction to transform the task of computing histograms for input structure of any signature to the task of computing histograms for tree structures. For the formulation of the lemma, we use the following terminology: An *s-tree structure* is a structure $\mathcal{T} = (V, E^{\mathcal{T}}, P_1^{\mathcal{T}}, \dots, P_s^{\mathcal{T}})$ over the signature $\tau_{s\text{-tree}} = \{E^2, P_1^1, \dots, P_s^1\}$ where $(V, E^{\mathcal{T}})$ is a directed tree.

▶ **Lemma 7.** *Let $\phi(X_1, \dots, X_\ell)$ be an MSO-formula over some signature τ and $w \in \mathbb{N}$. There is an $s \in \mathbb{N}$, a MSO-formula $\psi(X_1, \dots, X_\ell)$ over $\tau_{s\text{-tree}}$, and a first-order computable function that, on input of any τ-structure $\mathcal{S}$ with universe S and a width-w tree decomposition $D = (T_D, B_D)$ for $\mathcal{S}$, produces an s-tree structure $\mathcal{T}$, such that (a) the depth of $\mathcal{T}$ equals the depth of T_D plus 1, and (b) for all indices $i \in \{0, \dots, |S|\}^\ell$ we have $\operatorname{hist}(\mathcal{S}, \phi)[i] = \operatorname{hist}(\mathcal{T}, \psi)[i]$ and all other entries in the array $\operatorname{hist}(\mathcal{T}, \psi)$ are 0.*

Lemma 6 and Lemma 7 together provide a transformation from evaluating MSO-formulas on logical structures of bounded tree depth to evaluating them on s-tree structures of bounded depth.

Turning Formulas on Tree Structures into Tree Automata The trees underlying the s-tree structures that are produced by Lemma 7 do not impose an order on sibling nodes and nodes may have an unbounded number of children. Such trees, with the s unary predicates represented by binary strings, are known as *unordered, unranked* labeled trees in the literature [14]. "Unordered" means that there is no total order on sibling nodes and "unranked" stands for unbounded degree. In this section we introduce a new notion of automata that is appropriate for unordered labeled trees and prove that it exactly captures the MSO-definable properties of unordered labeled trees, resulting in a theorem which can be seen as an extension of the classical Büchi–Elgot–Trakhtenbrot Theorem. Moreover, the translation between MSO-formula and automata will preserve the sizes and number of solutions, thereby establishing a reduction from computing solution histograms for MSO-formulas on s-tree structures to evaluating tree automata.

Tree-automata-based proofs of time and space efficient variants of Courcelle's Theorem transform input structures into trees where the underlying tree has bounded degree. Then, in these proofs MSO-formulas on bounded degree trees are transformed into the classical tree automata for ranked labeled trees that were developed in the 1970's. Adopting this strategy and transforming s-tree structures with unbounded degree into tree structures of bounded degree would come at the cost of increasing the depth of the tree by at least a logarithmic factor and this would imply vertical data dependencies in the tree that we cannot hope to handle with constant-depth circuits. Due to this reason, we need an automaton model that does not force us to change the topology of the tree. For a similar reason, we cannot use some order on the children and translate to the tree automata for ordered unranked trees that are studied in the context of XML processing [11]; here the horizontal data dependencies on sibling subtrees are too high. In fact, such automata are able to decide any regular property on the ordered children of a node and, thus, cannot be simulated by constant-depth circuits.

The only automaton model from the literature that does not introduce dependencies between nodes that cannot be handled by constant-depth circuits is due to Libkin [14] who defined *counting unranked tree automata*, which are equivalent to MSO on unordered trees. The transition functions of these automata are defined in terms of Boolean functions: they allow us to assign a state q' to a node with symbol σ if a Boolean function $\delta(\sigma, q')$, which depends on the number of occurrences of states at the children, evaluates to 1. However, it is unclear (at least to us) how these automata could be used to compute solution histograms since we need to relate the states assigned to the subtrees of a node with the state that is assigned to the whole tree in a transparent way, without "hiding it inside a Boolean function."

In this section, we develop multiset automata as a notion that exactly captures the MSO-definable properties of unordered labeled trees (unranked or ranked) and that allows us to control the assignment of states to the children of a node such that we can later establish a cardinality-preserving transformation into arithmetic circuits.

A *multiset* M on a universe U is a function $\#_M \colon U \to \mathbb{N}$ that assigns a *multiplicity* to each element of U. We write $\mathcal{P}_\omega(U)$ for the class of all multisets on U and we write $\mathcal{P}_m(U)$ for the class of all multisets on U where each element has multiplicity at most m. Given a number $m \in \mathbb{N}$, let $M|_m$ be $\#_{M|_m}(e) = \min\{\#_M(e), m\}$ for $e \in U$. We call $M|_m$ the *capped version of M to multiplicity m.*

▶ **Definition 8** (Multiset Automata)**.** A *nondeterministic (bottom-up) multiset automaton* is a tuple $A = (\Sigma, Q, Q_a, \Delta)$ consisting of an *alphabet* Σ, a *state set* Q, a set $Q_a \subseteq Q$ of *accepting states*, and a *state transition relation* $\Delta \subseteq \Sigma \times \mathcal{P}_m(Q) \times Q$ for some constant *multiplicity bound* m. The automaton is *deterministic* if for every $\sigma \in \Sigma$ and every $M \in \mathcal{P}_m(Q)$ there is exactly one $q \in Q$ with $(\sigma, M, q) \in \Delta$; in this case we can view Δ as a *state transition function* $\delta \colon \Sigma \times \mathcal{P}_m(Q) \to Q$.

▶ **Definition 9** (Computation of a Multiset Automaton)**.** Let (T, l) be a labeled tree, where $l \colon V(T) \to \Sigma$ is the labelling function, and let $A = (\Sigma, Q, Q_a, \Delta)$ be a multiset automaton. A *computation* of A on (T, l) is a partial assignment $q \colon V(T) \to Q$ such that for every node $n \in V(T)$ for which $q(n)$ is defined, we have that (a) the value $q(c)$ is defined for each child c of n in T and (b) for the multiset $M = \{q(c) \mid c \text{ is a child of } n\}$ we have $(l(n), M|_m, q(n)) \in \Delta$. A computation is *accepting*, if $q(r) \in Q_a$ holds for the root node r of T. The *tree language* $L(A)$ contains all labeled trees accepted by A.

Given an s-tree structure $\mathcal{T} = (V, E^{\mathcal{T}}, P_1^{\mathcal{T}}, \dots, P_s^{\mathcal{T}})$ and sets $S_1, \dots, S_\ell \subseteq V$, let us write $T(\mathcal{T}, S_1, \dots, S_\ell)$ for the labeled tree whose node set is V, whose edge set is $E^{\mathcal{T}}$, and whose labeling function maps each node $v \in V$ to the bitstring $l_1 \dots l_s x_1 \dots x_\ell \in \{0,1\}^{s+\ell}$ with $l_i = 1 \Leftrightarrow v \in P_i^{\mathcal{T}}$ and $x_i = 1 \Leftrightarrow v \in S_i$. We write $T(\mathcal{T})$ in case $\ell = 0$.

▶ **Theorem 10.** *Let $s, \ell \in \mathbb{N}$.*

1. *For every MSO-formula $\phi(X_1, \dots, X_\ell)$ over $\tau_{s\text{-tree}}$ there is a multiset automaton A with alphabet $\{0,1\}^{s+\ell}$, such that for all s-tree structures $\mathcal{T}$ with universe V and $S_1, \dots, S_\ell \subseteq V$ we have $\mathcal{T} \models \phi(S_1, \dots, S_\ell)$ if, and only if, A accepts $T(\mathcal{T}, S_1, \dots, S_\ell)$.*
2. *For every multiset automaton A with alphabet $\{0,1\}^{s+\ell}$ there is an MSO-formula $\phi(X_1, \dots, X_\ell)$ over $\tau_{s\text{-tree}}$, such that for all s-tree structures $\mathcal{T}$ with universe V and $S_1, \dots, S_\ell \subseteq V$ we have $\mathcal{T} \models \phi(S_1, \dots, S_\ell)$ if, and only if, A accepts $T(\mathcal{T}, S_1, \dots, S_\ell)$.*

Our proof of the theorem follows Arnborg et al. [1], but modified to unranked trees rather than ranked trees and multiset automata rather than usual tree automata. It entails proofs of standard closure properties: The class of tree languages accepted by multiset automata is closed under intersection, union, complement, and for every nondeterministic multiset automaton there is a deterministic automaton accepting the same tree language.

From Automaton Evaluation to Arithmetic Circuit Evaluation Theorem 10 shows that in order to decide whether a given MSO-formula is true for a given tree, we can instead evaluate a multiset automaton. Since any logical structure of bounded tree depth can be transformed into a labeled tree of constant depth, we have all the ingredients together to prove Theorem 1 from the introduction.

Instead of just *deciding* formulas, in the remaining part of this section we turn our attention to the more challenging problem of computing the solution histograms. Our aim will be to replace the evaluation of automata by the evaluation of convolution circuits, see Lemma 11, such that the circuits's outputs are the sought solution histograms. Then we reduce the evaluation of convolution circuits to the evaluation of arithmetic circuits. Since arithmetic circuits of constant depth can be evaluated in GapAC^0, we get Theorem 2.

Theorem 10 establishes a link between formulas and multiset automata that is "solution-preserving" in the sense that there is a one-to-one correspondence between satisfying assignments to the free variables of the formulas and labelings of the trees that make an automaton A accept. In order to talk more easily about the number of such labelings, we recall the notion of *multicolorings* from [9]: Given a set S, a *multicoloring* of S is a tuple $(S_1, \dots, S_\ell)$ of subsets $S_j \subseteq S$ for $j \in \{1, \dots, \ell\}$. Given a set X of multicolorings of S, let $\mathrm{hist}(X)$ denote the $[|S|+1]^\ell$-array whose entry at index $i = (i_1, \dots, i_\ell)$ is the number of multicolorings $(S_1, \dots, S_\ell) \in X$ with $|S_1| = i_1, \dots, |S_\ell| = i_\ell$. Given a multiset automaton $A = (\{0,1\}^{s+\ell}, Q, Q_a, \Delta)$ and an s-tree structure $\mathcal{T}$ with universe V, let us write $S_A(\mathcal{T}, P)$ for the set of tuples $(S_1, \dots, S_\ell)$ with $S_i \subseteq V$ for which A reaches a state $q \in P$ at the root of $T(\mathcal{T}, S_1, \dots, S_\ell)$. Clearly, $S_A(\mathcal{T}, P)$ is a set of multicolorings of V. In particular, for the automaton A constructed in Theorem 10 for a formula ϕ we have $\mathrm{hist}(\mathcal{T}, \phi) = \mathrm{hist}(S_A(\mathcal{T}, Q_a))$. This means that "all" we have to do is to devise a way of computing $\mathrm{hist}(S_A(\mathcal{T}, Q_a))$ for a given automaton A and a tree $\mathcal{T}$.

For the computation of $\mathrm{hist}(S_A(\mathcal{T}, Q_a))$ we use *convolution circuits*, which are similar to arithmetic circuits, only instead of numbers whole histogram arrays are passed between gates. The basic gates of a convolution circuit are addition gates (which just add the arrays componentwise), subtraction gates (if there are no subtraction gates, the circuit is called *positive*), and convolution gates. The convolution $C = A * B$ of two arrays A and B is defined by $C[k] = \sum_{i \in [r]^\ell, j \in [s]^\ell \text{ with } k=i+j} A[i]B[j]$. The addition of two histograms corresponds exactly to combining two disjoint sets of solutions for the same tree, while the convolution of the histograms corresponds to combining the solutions of two sibling subtrees.

The construction of convolution circuits for a ranked automata is already described in [9] for the logspace setting. For the unranked automata considered in the present section, the construction needs to be more involved: For a node of a tree with a large number of children, the difficult part is to combine the histograms of all of these children so that they correspond to some particular capped version of the multiset of states reached at the children. The details of the recursive construction that achieves this can be found in our technical report [10]. The main result established is the following, where $\mathrm{val}(C)$ is the output of C:

▶ **Lemma 11.** *Let $A = (\{0,1\}^{s+\ell}, Q, Q_a, \delta)$ be a deterministic multiset automaton with multiplicity bound $m \in \mathbb{N}$. Then there is a first-order computable function that maps every s-tree structure $\mathcal{T} = (V, P_1^{\mathcal{T}}, \dots, P_s^{\mathcal{T}})$ to a convolution circuit C such that (a) $\mathrm{val}(C) = \mathrm{hist}(S_A(\mathcal{T}, Q_a))$, (b) the depth of C is bounded by a function that depends on A and linearly* *on the depth of $\mathcal{T}$, and (c) the fan-in of C is bounded by a function that depends on A and linearly on the degree of $\mathcal{T}$. Furthermore, if the degree of $\mathcal{T}$ is bounded by m, then C is positive.*

The final problem is to move from convolution circuits to arithmetic circuits. This is quite easy to achieve: For a vector $b = (b_1, \dots, b_\ell)$ of large bases and an ℓ-dimensional histogram h, set $\operatorname{num}_b(h) = \sum_{(i_1,\dots,i_\ell)\in\{0,\dots,|S|\}^\ell} h[i_1, \dots, i_\ell] b_1^{i_1} \cdots b_\ell^{i_\ell}$. Then $\operatorname{num}_b(A * B) = \operatorname{num}_b(A) \cdot \operatorname{num}_b(B)$. Thus, if we replace all constants c in the circuit C from the above lemma by $\operatorname{num}_b(c)$ and replace all convolution gates by multiplication gates, we get the arithmetic circuit claimed in Theorem 2.

Application: Placing Problems in Constant-Depth Circuit Classes The algorithmic meta theorems developed in this section allow putting problems into the uniform circuit classes AC^0, GapAC^0 and TC^0 by using direct MSO-based definitions of problems on structures of bounded tree depth or reductions to MSO-definable problems on bounded tree depth structures.

▶ **Theorem 12.** *For every $d \in \mathbb{N}$, the language $\{(G, s) \mid G$ has tree depth at most d and at least s perfect matchings$\}$ is TC^0-complete under AC^0-reductions.*

▶ **Theorem 13.** *For each $\ell \in \mathbb{N}$, the problem ℓ-INTEGER-LINEAR-PROGRAMMING, the version of integer linear programming where there are at most ℓ equations and the input numbers are given in unary, is complete for TC^0 under AC^0-reductions.*

4 Algorithmic Meta Theorems For Logarithmic-Depth Circuit Classes

In the present section we prove Theorems 4 and 5, which involve circuits of *logarithmic* depth rather than constant depth as in the previous section. The inputs now consist of (an encoding of) a logical structures $\mathcal{S}$ together with a tree decomposition D of $\mathcal{S}$, where T_D is given in term representation. The proofs follow along the same lines as those of Theorems 1 and 2, which involved the following steps: (1) Compute a tree decomposition of the input structure and move from formulas on the input structures to formulas on trees, (2) move from formulas on trees to the evaluation of tree automata, (3) move from the evaluation of tree automata to convolution circuits and from convolution circuits to arithmetic circuits. Clearly, computing a tree decomposition is no longer necessary since it is already part of the input. All of the other steps are also possible when the tree depth is no longer constant, the resulting circuits then simply have arbitrary depth. Since it is known that tree automata can be evaluated in NC^1 on trees given in term representation [15, 11], Theorem 4 follows (almost) immediately from our previous arguments.

The main obstacle in proving Theorem 5 is that one can evaluate arithmetic *formulas* of arbitrary depth in $\#\mathrm{NC}^1$ [2, 5], but evaluating arithmetic *circuits* can be done in $\#\mathrm{NC}^1$ only if the circuit has logarithmic depth (evaluating arithmetic circuits of arbitrary depth is already FP-hard when we cap the numbers to enforce the outputs to have only polynomial length, which they need not have in general). This means that, at some point in the course of the proof of Theorem 5, we need to move from trees or circuits of arbitrary depth to logarithmic depth. Previous papers, such as [13], have faced a similar obstacle, namely evaluating tree-like structures of arbitrary depth whose nodes perform a complicated algebraic operation on the values of their children. In these papers, the approach was to somehow extend the ideas used in the proof that evaluating arithmetic formulas can be done in $\#\mathrm{NC}^1$ [2, 5] to more general algebraic structures.

Our approach to tackling this problem is different and may be of independent interest. Rather than trying to adapt algorithms to the convolution computations that would be needed in our setting, we attack the problem at a much earlier stage: We balance the tree decomposition. Since all of our later algorithms do not increase the depth of the considered trees, we get the desired arithmetic circuits of logarithmic depth. In detail, we show how a balanced width-3 tree decomposition of an arbitrary tree can be computed using constant-depth threshold circuits. The construction has two key properties. First, it is based on the classical tree contraction method, which is used a lot in the context of parallel random access machines, but which hitherto was not used in the context of NC^1. Using it will allow us to compute a balanced tree decomposition even in TC^0 and not only in NC^1. Second, the tree decomposition we compute does *not* have the property that the nodes of each bag form a balanced separator of some part of the tree. Normally, recursive NC^1 algorithms find sets of nodes that in each step split up the tree into components that are smaller than the current tree by a certain factor. This is not the case for the sets of nodes in our bags: While we *can* ensure that the whole tree is balanced and, hence, has logarithmic depth, we *cannot* ensure that the elements of any individual bag split the tree in some balanced way. Naturally, a lot of bags will have this balancing property (otherwise no tree decomposition of logarithmic depth would result), but we cannot say anything about where these balancing bags will lie. It seems that this more global approach (just find a tree decomposition of logarithmic depth) instead of the traditional local approach (find a balancing separator for each subtree recursively) allows us to lower the circuit complexity to a constant depth.

In the following, we first review term representations and, then, sketch the proof of Theorem 4. After that, we describe the technical result on how a width-3 tree decomposition of any tree can be computed in FTC^0; and then use this result to prove Theorem 5. At the end of this section, we sketch applications of the established meta theorems.

Background on Term and Ancestor Representations of Trees Up to now, the details of how tree decompositions are encoded as strings was not important; indeed, in the context of constant tree *depth* almost any encoding of the input graph and of tree decompositions will do since they can easily be transformed into one another. In the context of logarithmic-depth circuits, however, it is well known that it is crucial that the "ancestor relation" of the tree (for directed trees, this is exactly the transitive closure) is made accessible to the circuits, rather than just a pointer-structure or an adjacency matrix. There are two different ways of encoding this relation: Explicitly as a list of pairs or implicitly as a bracket structure. The two representations can be transformed into one another using TC^0-circuits and we will use both of them.

Decision by Logarithmic-Depth Circuits for Term Representations As mentioned earlier, the proof machinery established in Section 3 allows us already to prove Theorem 4 from the introduction. The only obstacle is that in all intermediate steps we do not only need to compute trees, but also their term representations. This is straightforward to achieve, see our technical report for details [10].

Computing Width-3 Tree Decompositions of Trees in Constant Depth We show that using only TC^0-circuits, for every tree T given in term representation we can compute a width-3 tree decomposition (T_D, B_D) of T (regarded as a graph) such that T_D is a perfectly balanced binary tree (and, hence, has logarithmic depth):

▶ **Theorem 14.** *There is a* DLOGTIME*-uniform* FTC^0*-circuit family that on input of a term representation of a tree T outputs a term representation of a width-3 tree decomposition*

(S,B) of T where S is a balanced binary tree.

The proof idea is surprisingly simple: As was already implicitly observed by Buss [4], the trees resulting from the different stages of the classical tree contraction method can be computed in TC^0. During a tree contraction step, for a leaf n, one considers its sibling s, its parent p, and its grandparent g. We call $c = (n, s, p, g)$ a *contraction tuple* and associate a set $I(c)$ of nodes with it that covers all nodes that have been "contracted away" inside this tuple. Our two key observations are the following: (a) For every two contraction tuples c and c', the sets $I(c)$ and $I(c')$ are either disjoint or one is contained in the other. From this we can derive that the contraction tuples can be arranged in a tree of logarithmic depth. (b) If we attach the bag $\{n, s, p, g\}$ to each node (n, s, p, g) of this "tree of contraction tuples," we get a width-3 tree decomposition of the original tree.

Computing Histograms by Logarithmic-Depth Circuits for Term Representations Recall that our goal for the present section is to prove Theorem 5, that our line of proof was to do the same sequence of transformations as we did in Section 3 for constant depth circuits, and that the missing building block was a procedure to turn an arbitrary tree decomposition into a tree decomposition of logarithmic depth. Theorem 14 provides us with the tools to build this missing block.

Application: Placing Problems in Logarithmic-Depth Circuit Classes We discuss some examples of how to use the algorithmic meta theorems for logarithmic-depth circuit classes to put decision and counting problems into NC^1 and $\#\mathrm{NC}^1$, respectively. A simple example is the problem of evaluating Boolean sentences that are given as terms, a problem well known to lie in DLOGTIME-uniform NC^1 [3, 2].

Buss [3] extended his NC^1-approach for the evaluation of Boolean sentences to also cover the membership problem for parenthesis languages. Later researchers adapted this approach to show that larger classes of context-free languages can be decided in NC^1, with the most general one being the result of Dymond [8] that languages recognizable by *visible pushdown automata* are in NC^1. Besides deciding whether a string is accepted by a fixed VPA, recently the problem of counting the number of accepting computation paths of nondeterministic VPAs was studied in the context of logarithmic-depth circuits and shown to be complete for $\#\mathrm{NC}^1$ by Krebs, Limaye, and Mahajan [13]. Theorems 4 and 5 can be used to reprove that these decision and counting problems are in NC^1 and $\#\mathrm{NC}^1$, respectively [10].

5 Conclusion

In the present paper we transferred the idea of unifying the study of computational problems by using MSO-based problem definitions and tree decompositions to circuit complexity classes inside logarithmic space, leading to algorithmic meta theorems for Boolean and arithmetic circuit classes of constant and logarithmic depth. Regarding constant-depth circuits, we discussed how to put the problem of solving a linear equation system that contains a constant number of equations whose coefficient are given in unary into TC^0. The most general application for logarithmic-depth circuits showed an alternative proof of a recent result that one can count the number of accepting paths of visible pushdown automata in $\#\mathrm{NC}^1$.

A natural direction of further research would be to try and use our theorems for logarithmic-depth circuits to simulate some generalization of visible pushdown automata where the height of the stack at different positions in time can be computed in advance; say, in NC^1 instead of FTC^0. Another direction would be to find algorithmic meta theorems that unify problems lying in other complexity classes around logarithmic space. Such research would need to address all three dimensions of algorithmic meta theorems: (a) the considered logic, (b) the considered class of input structures, and (c) the considered complexity class. We may go from MSO to more expressive or less expressive logics (like, for example, MSO on graphs where we can only quantify over vertex sets). Or we may consider other classes of structures that are more or less restrictive than bounded tree width (like, for example, bounded clique width).

References

1 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *J. Algorithm*, 12(2):308–340, 1991.

2 S. Buss, S. Cook, A. Gupta, and V. Ramachandran. An optimal parallel algorithm for formula evaluation. *SIAM J. Comput.*, 21(4):755–780, 1992.

3 Samuel R. Buss. The boolean formula value problem is in ALOGTIME. In *Proceedings of STOC 1987*, pages 123–131. ACM, 1987.

4 Samuel R. Buss. Algorithms for boolean formula evaluation and for tree contraction. In Peter Clote and Jan Krajíček, editors, *Arithmetic, Proof Theory, and Computational Complexity*, pages 95–115. Oxford University Press, 1993.

5 Hervé Caussinus, Pierre McKenzie, Denis Thérien, and Heribert Vollmer. Nondeterministic NC^1 computation. *J. Comput. Syst. Sci.*, 57(2):200–212, 1998.

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

7 Bireswar Das, Samir Datta, and Prajakta Nimbhorkar. Log-space algorithms for paths and matchings in k-trees. In *Proceedings of STACS 2010*, volume 5 of *LIPIcs*, pages 215–226. Schloss Dagstuhl LZI, 2010.

8 Patrick Dymond. Input-driven languages are in $\log n$ depth. *Information Processing Letters*, 26(5):247–250, 1988.

9 Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of Bodlaender and Courcelle. In *Proceedings of FOCS 2010*, pages 143–152, 2010.

10 Michael Elberfeld, Andreas Jakoby, and Till Tantau. Algorithmic meta theorems for circuit classes of constant and logarithmic depth. Technical Report ECCC-TR11-128, 2011.

11 Georg Gottlob, Christoph Koch, Reinhard Pichler, and Luc Segoufin. The complexity of XPath query evaluation and XML typing. *J. ACM*, 52(2):284–335, 2005.

12 Martin Grohe and Stephan Kreutzer. Methods for algorithmic meta theorems. In *Model Theoretic Methods in Finite Combinatorics*, volume 558 of *Contemporary Mathematics*, pages 181–206. American Mathematical Society, 2011.

13 Andreas Krebs, Nutan Limaye, and Meena Mahajan. Counting paths in VPA is complete for $\#NC^1$. In *Proceedings of COCOON 2010*, volume 6196 of *LNCS*, pages 44–53. Springer, 2010.

14 Leonid Libkin. Logics for unranked trees: An overview. *Logical Methods in Computer Science*, 2(3), 2006.

15 Markus Lohrey. On the parallel complexity of tree automata. In *Proceedings of RTA 2001*, volume 2051 of *LNCS*, pages 201–215. Springer, 2001.

16 Jaroslav Nešetřil and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Combin.*, 27(6):1022—1041, 2006.

17 Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion I. Decompositions. *Eur. J. Combin.*, 29(3):760–776, 2008.

18 Egon Wanke. Bounded tree-width and LOGCFL. *J. Algorithm*, 16(3):470–491, 1994.