On Space Efficiency of Algorithms Working on Structural Decompositions of Graphs*

Michał Pilipczuk[1] and Marcin Wrochna[2]

1 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
`michal.pilipczuk@mimuw.edu.pl`

2 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
`m.wrochna@mimuw.edu.pl`

Abstract

Dynamic programming on path and tree decompositions of graphs is a technique that is ubiquitous in the field of parameterized and exponential-time algorithms. However, one of its drawbacks is that the space usage is exponential in the decomposition's width. Following the work of Allender et al. [Theory of Computing, '14], we investigate whether this space complexity explosion is unavoidable. Using the idea of reparameterization of Cai and Juedes [J. Comput. Syst. Sci., '03], we prove that the question is closely related to a conjecture that the Longest Common Subsequence problem parameterized by the number of input strings does not admit an algorithm that simultaneously uses **XP** time and **FPT** space. Moreover, we complete the complexity landscape sketched for pathwidth and treewidth by Allender et al. by considering the parameter *tree-depth*. We prove that computations on tree-depth decompositions correspond to a model of non-deterministic machines that work in polynomial time and logarithmic space, with access to an auxiliary stack of maximum height equal to the decomposition's depth. Together with the results of Allender et al., this describes a hierarchy of complexity classes for polynomial-time non-deterministic machines with different restrictions on the access to working space, which mirrors the classic relations between treewidth, pathwidth, and tree-depth.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.2.3 Tradeoffs between Complexity Measures, G.2.2 Graph Theory

Keywords and phrases tree decomposition, LCS, tree-depth, NAuxSA, Savitch's theorem

Digital Object Identifier 10.4230/LIPIcs.STACS.2016.57

1 Introduction

Treewidth is a parameter that measures how easily a graph can be decomposed into a tree-like structure, called a *tree decomposition*. While initially introduced by Robertson and Seymour in their Graph Minors project [41], treewidth has found numerous applications in the field of algorithms, because many problems that are intractable on general graphs, become efficiently solvable on graphs of small treewidth. Theorems of Courcelle [14] and of Arnborg et al. [5] explain that every problem expressible in Monadic Second Order logic can be solved in time $f(s)\cdot n$ on graphs of treewidth s and size n, for some function f. While f can be non-elementary in general, for many classic problems, like Vertex Cover, 3Coloring, or Dominating

* Research supported by Polish National Science Centre grant DEC-2013/11/D/ST6/03073. During the work on these results, Michał Pilipczuk held a post-doc position at Warsaw Center of Mathematics and Computer Science, and was supported by Foundation for Polish Science (FNP) via START stipend programme.

33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016).
Editors: Nicolas Ollinger and Heribert Vollmer; Article No. 57; pp. 57:1–57:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Set, the natural dynamic programming approach yields a running time of $\mathcal{O}(c^s \cdot n)$ for a small constant c. Dynamic programming procedures working on tree decompositions are important for applications, as they often serve as critical subroutines in more complex techniques, e.g., subexponential parameterized algorithms derived using *bidimensionality* [16], or approximation schemes obtained via Baker's approach [7]. Algorithms working on tree decompositions are usually analyzed in the paradigm of *parameterized complexity*, where treewidth is the considered parameter. We refer to textbooks [15, 17, 22] for a broad introduction, and to a recent survey of Langer et al. [30] for more specific results.

A certain limitation of dynamic programming on a tree decomposition is that it uses space exponential in its width, which is often a prohibitive factor in practical applications. Therefore, recently there is much focus on reducing the space complexity of exponential-time algorithms to polynomial, even at the cost of slightly worsening the time complexity [6, 9, 23, 24, 34, 37]. Here, the usage of algebraic tools proved to be an extremely useful approach. Unfortunately, algorithms working on treewidth remain a family where virtually no progress has been achieved in this matter. Therefore, a natural question arises: Can we reduce the space complexity of algorithms working on tree decompositions while keeping (or moderately worsening) their time complexity? This was first asked explicitly by Lokshtanov et al. [33], who sketched how a simple tradeoff achieves polynomial space complexity while increasing the time complexity to $2^{\mathcal{O}(s^2)} + \mathcal{O}(n^2)$. The question was reiterated later by Langer et al. [30].

Following early completeness results of Monien and Sudborough [36] on bandwidth-constrained problems and of Gottlob et al. [26] on conjunctive queries of bounded treewidth, Allender et al. [4] recently initiated the systematic study of satisfaction complexity in variable path- and treewidth. Essentially, they observe that CSP-like problems—say, 3Coloring for concreteness[1]—when limited to instances of small treewidth or pathwidth, are complete for certain complexity classes under logspace reductions. More precisely, when the input graph is equipped with a path decomposition of width at most $s(n) \geq \log n$, for some fixed function s of the input size, then 3Coloring (denoted in this case as pw-3Coloring$[s]$) is complete for the class $\mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{s(\text{poly})}]$: problems that admit non-deterministic algorithms working simultaneously in polynomial time and space $\mathcal{O}(s(\text{poly}(n)))$. Similarly tw-3Coloring$[s]$, where $s(n)$ bounds the width of a given tree decomposition, is complete for the class $\mathbf{NAuxPDA}[\underset{time}{\text{poly}}, \underset{space}{s(\text{poly})}]$; the difference with $\mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{s(\text{poly})}]$ is that the algorithm can use an auxiliary push-down of unlimited size, to which read/write access is only from the top. Allender et al. also describe the class in terms of *semi-unbounded fan-in* (SAC) circuits. They assume $s(n) = \log^k n$, but the proof works in the more general setting given below.

▶ **Theorem 1** ([4]). *Let $s(n) \geq \log n$ be a nice function*[2]*.*
Then pw-3Coloring$[s]$ *is complete for* $\mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{s(\text{poly})}]$ *under logspace reductions, whereas* tw-3Coloring$[s]$ *is complete for* $\mathbf{NAuxPDA}[\underset{time}{\text{poly}}, \underset{space}{s(\text{poly})}]$ *under logspace reductions.*

Thus, the feasibility of various space-time tradeoffs when working on tree/path decompositions is equivalent to inclusions of corresponding complexity classes. For instance (assuming for conciseness $\forall_c\ s(n^c) = \mathcal{O}(s(n))$, e.g., $s(n) = \log^k n$ for $k \geq 1$), pw-3Coloring$[s]$ is solvable:

[1] Allender et al. use SAT parameterized by treewidth/pathwidth of its primal graph as an exemplary problem, but SAT and 3Coloring can be easily seen to be equivalent under logspace reductions; see Lemma 10. In this paper, we prefer to use 3Coloring as an exemplary hard CSP-like problem.

[2] By a *nice* function we mean a function s that is constructible and such that $s(n)/\lg n$ is non-decreasing.

- in time $2^{o(s(n)\log n)}$ and space $2^{o(s(n))}$ if and only if $\mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{s}] \subseteq \mathbf{D}[\underset{time}{2^{o(s\cdot\log)}}, \underset{space}{2^{o(s)}}]$;
- in time $2^{\mathcal{O}(s(n))}$ and space $\text{poly}(n)$ if and only if $\mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{s}] \subseteq \mathbf{D}[\underset{time}{2^{\mathcal{O}(s)}}, \underset{space}{\text{poly}}]$.

Similar statements can be inferred for treewidth. In contrast, the best known determinization for $\mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{s}]$ come from a brute-force approach or Savitch's theorem [43], yielding respectively (for $s(n) \geq \lg n$) $\mathbf{D}[\underset{time}{2^{\mathcal{O}(s)}}] = \mathbf{D}[\underset{time}{2^{\mathcal{O}(s)}}, \underset{space}{2^{\mathcal{O}(s)}}]$ and $\mathbf{D}[\underset{space}{s \cdot \log}] = \mathbf{D}[\underset{time}{2^{\mathcal{O}(s\cdot\log n)}}, \underset{space}{s \cdot \log}]$.

In this manner, Allender et al. conclude that, intuitively speaking, achieving better time-space tradeoffs for algorithms working on path and tree decompositions of small width would require developing a general technique of improving upon the tradeoff of Savitch. As Lipton phrased it, "one of the biggest embarrassments of complexity theory is the fact that Savitch's theorem has not been improved [...]. Nor has anyone proved that it is tight" [31].

Allender et al. argue that such an improvement would contradict certain rescaled variants of known conjectures about the containment of time- and space-constrained classes, in particular the assumption that $\mathbf{NL} \not\subseteq \mathbf{SC}$; we refer to [4] for details. We consider the study of Allender et al. not as a definite answer in the topic, but rather as an invitation to a further investigation of the introduced conjectures.

Our Contribution. In the Longest Common Subsequence problem (LCS), we are given an alphabet Σ and k strings over Σ, and ask for the longest sequence of symbols that appears as a subsequence in each input string. The applicability of the LCS problem in, e.g., computational biology, motivated many to search for faster, more space-efficient algorithms, as the classical dynamic programming solution, running in time and space $\mathcal{O}(n^k)$ (where n is the length of each string) is often far from practical. From the point of view of parameterized complexity, LCS parameterized by k is W[t]-hard for every level t [11], remains W[1]-hard for a fixed-sized alphabet [39], and is W[1]-complete when parameterized jointly by k and ℓ, the target length of the subsequence [27]. In a recent breakthrough, Abboud et al. [1] proved that the existence of an algorithm with running time $\mathcal{O}(n^{k-\varepsilon})$, for any $\varepsilon > 0$, would contradict the Strong Exponential Time Hypothesis. As far as the space complexity is concerned, only modest progress has been achieved: The best known result, by Barsky et al. [8], improves the space complexity to $\mathcal{O}(n^{k-1})$. This motivates us to formulate the following conjecture.

▶ **Conjecture 2.** *There is no algorithm for LCS that works in time $n^{f(k)}$ and space $f(k)\text{poly}(n)$ for a computable function f, where k is the number of input strings and n their total length.*

Quite surprisingly, we show that Conjecture 2 is closely related to the question of time-space tradeoffs for algorithms working on small pathwidth, as detailed in Theorem 15. There, the conjecture is sandwiched between a weaker statement that it is impossible to achieve subexponential space while keeping single exponential time complexity, and a stronger statement that this holds even if we allow the time complexity exponent to increase by an arbitrarily slowly growing function of the width. To prove this, we use a completeness result of Elberfeld et al. [21] for LCS, which allows to formulate Conjecture 2 as an equivalent statement in parameterized complexity about the impossibility of determinization results improving upon Savitch's theorem. Using the ideas of Cai and Juedes [12] connecting subexponential complexity to fixed-parameter tractability, we consider a reparameterized version of pw-3Coloring. This allows us to compare questions concerning time-space tradeoffs for pw-3Coloring and determinization of $\mathbf{N}[\underset{time}{t}, \underset{space}{s}]$ classes to those concerning parameterized classes and the complexity of LCS. In particular, we show that Conjecture 2 implies $\mathbf{NL} \not\subseteq \mathbf{D}[\underset{time}{\text{poly}}, \underset{space}{\text{poly}\log}]$ (the latter class being usually called **SC**) and is implied by a rescaled version of the following stronger variant: $\mathbf{NL} \not\subseteq \mathbf{D}[\underset{time}{2^{o(\log^2 n)}}, \underset{space}{n^{o(1)}}]$.

In the second part of this work, we complement the findings of Allender et al. [4] by considering the graph parameter *tree-depth*. Tree-depth of a graph is lower bounded by its pathwidth and upper bounded by its treewidth times $\lg n$. Our motivation for considering this parameter is two-fold. First, recent advances have uncovered a wide range of topics where tree-depth appears naturally. For instance, it plays an important role in the theory of sparse graphs of Nešetřil and Ossona de Mendez [38], it is the key factor in classifiyng homomorphism problems that can be solved in logspace [13], and characterizes classes of graphs where the expressive power of First-Order and Monadic Second-Order logic coincides [19]. It was rediscovered several times under different names: *minimum elimination tree height* [40], *ordered chromatic number* [28], *vertex ranking* [10], or the maximum number of introduce nodes on a root-to-leaf path of a tree decomposition [24].

Second, algorithms working on tree-depth decompositions model generic exponential-time Divide&Conquer algorithms. In this approach, after finding a small, balanced separator S in the graph, the algorithm tries all possible ways a solution can interact with S, and solves connected components of $G - S$ recursively. This naturally gives rise to a tree-depth decomposition of the graph, where S is placed on top, and decompositions of the components of $G - S$ are attached below it as subtrees. The maximum total number of separator vertices handled at any moment in the recursion corresponds to the depth of the decomposition. Thus, many classic Divide&Conquer algorithms, including the ones derived for planar graphs using the Lipton-Tarjan separator theorem [32], can be reinterpreted as first building a tree-depth decomposition of the graph using a separator theorem, and then running the algorithm on it.

Most importantly for us, recursive algorithms working on tree-depth decompositions run in polynomial space. For instance, such an algorithm for 3Coloring on a tree-depth decomposition of depth s runs in time $3^s \cdot \text{poly}(n)$ and space $\mathcal{O}(s + \log n)$ (see Lemma 20), which places td-3Coloring$[s]$ in $\mathbf{D}[\underset{time}{2^{\mathcal{O}(s)}\text{poly}}, \underset{space}{s + \log}] = \mathbf{D}[\underset{space}{s + \log n}]$. This is immediate for CSP-like problems like 3Coloring, but recently Fürer and Yu [24] showed that algebraic transforms can be used to reduce the space usage to polynomial in n also for other problems, like counting perfect matchings or dominating sets. We describe how this approach gives an $3^s \cdot \text{poly}(n)$-time $\text{poly}(n)$-space algorithm for Dominating Set in more detail in the full version of the article. This means that the reduction of space complexity that is conjectured to be impossible for treewidth and pathwidth, actually is possible for tree-depth. Therefore, we believe that it is useful to study the computation model standing behind low tree-depth decompositions, in order to understand how it differs from the models for treewidth and pathwidth.

Consequently, mirroring Theorem 1, we prove that computations on tree-depth decompositions exactly correspond to the class $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s}]$: problems that can be decided by a non-deterministic Turing Machine that uses polynomial time and logarithmic space, but also has access to an auxiliary stack of maximum height s. The stack can be freely read by the machine, just as the input tape, but write access is only via `push`/`pop` operations.

► **Theorem 3.** *Let $s(n) \geq \log^2 n$ be a nice function. Then* td-3Coloring$[s]$ *is complete for* $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s(\text{poly})}]$ *under logspace reductions.*

Thus, computations on tree-depth and path decompositions differ by the access restrictions to $\mathcal{O}(s)$ space used by the machine. While for pathwidth this space can be accessed freely, for tree-depth all except an $\mathcal{O}(\log n)$ working buffer has to be organized in a stack.

The proof of Theorem 3 largely follows the approach of Akatov and Gottlob [3], who proved a different completeness result for the class $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{\log^2}]$, which they call $\mathbf{DC}^1$. The main idea is to regularize the run of the machine so that the push-pop tree has the rigid shape of a full binary tree. Then we can use this concrete structure to "wrap around" gadgets encoding an accepting run of a regularized NAuxSA machine. However, the motivation in the work of Akatov and Gottlob was answering conjunctive queries in a hypergraph by exploiting a kind of balanced decomposition, and hence the problem proven to be complete for $\mathbf{DC}^1$ is a quite general and expressive problem originating in database motivations; see [2, 3] for details. In our setting, in order to get a reduction to 3Coloring, we need to work more to encode an accepting run. In particular, to encode each part of the computation where no push or pop is performed, instead of producing a single atom in a conjunctive query, we use computation gadgets that originate in Cook's proof of the NP-completeness of SAT. The assumption that the computation has a polynomial number of steps is essential here for bounding the tree-depth of each such gadget. This way, Theorem 3 presents a more natural complete problem for $\mathbf{DC}^1$.

Another difference is that Theorem 3 works for any well-behaved function $s(n) \geq \log^2 n$, as opposed to the bound $s(n) = \log^2 n$ inherent to the problem considered by Akatov and Gottlob. For this, the crucial new idea is to increase the working space of the machine to $s(n)/\log n$ in order to be able to perform regularization – a move that looks dangerous at first glance, but turns out not to increase the expressive power of the computation model. This proves the following interesting by-product of our work.

▶ **Theorem 4.** *Let $s(n) \geq \log^2 n$ be a nice function. Then*

$$\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s(\text{poly})}] = \mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{s(\text{poly})/\log}, \underset{height}{s(\text{poly})}].$$

The following determinization follows from the $\mathcal{O}(s + \log n)$ algorithm for td-3Coloring.

▶ **Theorem 5.** *Let $s(n) \geq \log^2(n)$ be a nice function. Then*

$$\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s(\text{poly})}] \subseteq \mathbf{D}[\underset{space}{s(\text{poly})}].$$

Theorem 5 for $s(n) = \log^2 n$ also follows from the work of Akatov and Gottlob [3]. Observe that now the justification for the assumption $s(n) \geq \log^2 n$ becomes apparent: for, say, $s(n) = \log n$, the theorem would state that $\mathbf{L} = \mathbf{NL}$, a highly unexpected outcome.

We find Theorem 5 interesting, because a naive simulation of the whole configuration space for NAuxSA would require space exponential in s. It appears, however, that the exponential blow-up of the space complexity can be avoided. We do not see any significantly simpler way to prove this result other than going through the td-3Coloring$[s]$ problem, and hence it seems that the tree-depth view gives a valuable insight into the computation model of NAuxSA. The classic relations between treewidth, pathwidth and tree-depth are, through completeness results, mirrored in a hierarchy between NAuxPDA, N, and NAuxSA classes, as detailed in the concluding section. In particular, this answers a question of Akatov and Gottlob [2, 3] about the relation of $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{\text{poly}\log}]$ to other classes in $\mathbf{NP}$.

Finally, using Theorem 3 we also give an alternative view on NAuxSA computations using alternating Turing machines in Theorem 21, answering another question of Akatov and Gottlob. From this point of view, Theorem 5 is immediate.

2 Preliminaries

Reductions and complexity classes. For two languages P, Q, we write $P \leq_L Q$ when P is logspace reducible to Q. Most of the complexity classes we consider are closed under logspace reductions. Because we handle various measures of complexity and compare a wide array of classes that bound two measures simultaneously, we introduce the following notation. A complexity class is first described by the machine model: **D**, **N**, **A** denote deterministic, non-deterministic, and alternating (see [42]) Turing machines, respectively. Then bounds on complexity measures are described (up to constant factors) as a list of functions with the measure's name underneath. All functions except the symbol f (which we reserve for classes in parameterized complexity) are functions of the input size n. For example, $\mathbf{N}[\underset{time}{t}, \underset{space}{s}]$ is the class often known as $\mathrm{NTiSp}(t(n), s(n))$. We write $\mathrm{poly}(n)$ for $n^{\mathcal{O}(1)}$, e.g., $\mathbf{D}[\underset{time}{\mathrm{poly}}] = \mathbf{P}$. An auxiliary push-down or stack is denoted as AuxPDA or AuxSA, respectively: the difference is that a push-down can only be read at the top, while a stack can be read just as a tape (both can be written to only by pushing and popping symbols at the top), see e.g. [44]. The measure named *height* is the maximum height of the push-down or stack.

We write lg for the logarithm with base 2 and log when the base is irrelevant. We say a function $s : \mathbb{N} \to \mathbb{N}$ is *constructible* if there is a Turing machine which given a number n in unary outputs $s(n)$ in unary using logarithmic space; in particular, this implies $s(n) \leq \mathrm{poly}(n)$. A function s is *nice* if it is constructible and $\frac{s(n)}{\lg n}$ is non-decreasing. For simplicity, we will assume all functions $s : \mathbb{N} \to \mathbb{N}$ describing complexity bounds to be nice.

Note that logspace reductions can blow-up instance sizes polynomially, hence the closure of $\mathbf{N}[\underset{time}{\mathrm{poly}}, \underset{space}{s}]$ under such reductions is $\mathbf{N}[\underset{time}{\mathrm{poly}}, \underset{space}{s(\mathrm{poly})}]$, for example. These are equal for functions $s(n)$ such that $s(\mathrm{poly}(n)) \leq \mathcal{O}(s(n))$ (that is, if for every $c > 0$ there is a $d > 0$ such that $s(n^c) \leq d \cdot s(n)$). This includes $\lg^k(n)$ for any $k \geq 1$ and $\lg n \lg \lg n$, for example.

Structural parameters. We recall the definition of tree-depth. See e.g. [15, 41] for definitions of treewidth and pathwidth. For technical reasons, we assume that in all given tree and path decompositions $\mathcal{T}$, $|\mathcal{T}| \leq 2|V(G)|^2$; standard methods allow to prune any decomposition to this size in logspace, see e.g. [29, Lemma 13.1.2]. For conciseness, we will refer to the certifying structures as *decompositions* for all three parameters.

▶ **Definition 6** (tree-depth)**.** A *tree-depth decomposition* of an undirected graph G is a rooted forest $\mathcal{T}$ (disjoint union of rooted trees) together with a bijection μ from the vertices of G to the nodes of $\mathcal{T}$, such that for every edge uv of G, $\mu(u)$ is an ancestor of $\mu(v)$ or vice-versa in $\mathcal{T}$. The *depth* of $\mathcal{T}$ is the largest number of nodes on a path between a root and a leaf. The *tree-depth* of G is the minimum depth over all possible tree-depth decompositions of G.

The following lemma describes well-known inequalities between the three parameters.

▶ **Lemma 7** (♠)**.** [3] *There is a constant $c \in \mathbb{N}$ such that for any graph G, $\mathtt{td}(G) \geq \mathtt{pw}(G) \geq \mathtt{tw}(G) \geq \mathtt{td}(G)/(c \cdot \log|V(G)|)$. Furthermore, each inequality is certified by an algorithm that transforms the respective graph decompositions in logspace.*

For a graph problem, such as 3Coloring, a structural parameter $\pi \in \{\mathtt{td}, \mathtt{pw}, \mathtt{tw}\}$, and a nice function $s : \mathbb{N} \to \mathbb{N}$, we define π-3Coloring$[s]$ to be the decision problem where given

[3] Proofs of statements marked with ♠ are deferred to the full version of the article (in the appendix).

an instance G of 3Coloring and a π-decomposition of G, we ask whether the decomposition has width at most $s(|V(G)|)$ and G is a yes-instance of 3Coloring. The assumption that a decomposition is given on input is to factor away the complexity of finding it, which is a problem not directly relevant to our work. Note that the validity and width/depth of a decomposition given in any natural encoding can easily be checked in logarithmic space.

Observe also that for any $c > 0$, π-3Coloring$[s(n)]$ is equivalent to π-3Coloring$[s(n^c)]$ under logspace reductions. A reduction to π-3Coloring$[s(n^c)]$ is trivial, while the reverse reduction follows easily by padding: adding isolated vertices up to size n^c that do not change the answer nor the value of π. Also, as we assume s to be nice, we have $\frac{s(n)}{\lg n} \leq \frac{s(n^c)}{\lg n^c}$, hence $c \cdot s(n) \leq s(n^c)$ for any $c \geq 1$. This implies that π-3Coloring$[c \cdot s(n)]$ is equivalent to π-3Coloring$[s(n)]$. Thus, the hierarchy of Lemma 7 takes the following form.

▶ **Corollary 8.** *Let $s : \mathbb{N} \to \mathbb{N}$ be a nice function. Then*

$$\text{td-3Coloring}[s] \leq_L \text{pw-3Coloring}[s] \leq_L \text{tw-3Coloring}[s] \leq_L \text{td-3Coloring}[s \cdot \log].$$

Equivalence of problems. A reduction between two graph problems *preserves structural parameters* if for each parameter $\mathtt{tw}, \mathtt{pw}, \mathtt{td}$ and any instance with graph G, a decomposition of G of width/depth at most s can be transformed in logspace into a decomposition of the graph H produced by the reduction of width/depth at most $\mathcal{O}(s)$. Many NP-hardness reductions have this property, in particular those that replace each vertex or edge with a constant-size gadget (see the 'local replacement' and 'component design' methods in Garey and Johnson [25]). For example, 3Coloring and variants of SAT are equivalent in all our theorems, while Vertex Cover or Dominating Set (defined in [25]) are at least as hard.

▶ **Definition 9.** Let ϕ be a CNF formula. The *primal (Gaifman) graph of ϕ* is the graph with a vertex for each variable of ϕ and an edge between every pair of variables that appear together in some clause. The *incidence graph of ϕ* is the bipartite graph with a vertex for each clause and each variable of ϕ, where every clause is adjacent to variables contained in it.

▶ **Lemma 10 (♠).** *The following problems are equivalent under logspace reductions preserving structural parameters:* 3Coloring*, CNF-SAT (with the primal graph), k-SAT (with either the primal or incidence graph) for each $k \geq 3$. Furthermore,* Vertex Cover*,* Independent Set *and* Dominating Set *each admit such a reduction from the above problems.*

Cook's theorem with bounded space. A common element in our reductions is the description of Turing machine computations using CNF formulas, as in Cook's theorem. Already Monien and Sudborough [36] observed that Cook's reduction applied to machines with bounded space yields formulas of bounded width. The difference is that machine's worktape space bound can be much smaller than the input word—access to the read-only input tape has to be implemented differently. To later handle stack machines, we also need to consider a second input tape separately. We informally state the version of Cook's construction we need.

▶ **Lemma 11** (Computation gadget, ♠)**.** *Let M be an NTM over alphabet Σ with two read-only input tapes and one work tape. Given an input word α of length n and integers s, t, h in unary such that $\lg n, \lg h \leq \mathcal{O}(s)$, one can in logspace output a CNF formula such that:*

- *The formula has $\mathrm{poly}(n, t, s, h)$ variables, including named variables $w_1, \ldots, w_{h \cdot \lceil |\Sigma| \rceil}$, $u_1, \ldots, u_{\Theta(s)}$, $v_1, \ldots, v_{\Theta(s)}$, describing: a word $\bar{w}$ and two configurations $\mathbf{u}$, $\mathbf{v}$ of M (up to s symbols of the working tape, heads' positions encoded in binary, and the state).*
- *Any assignment to the named variables can be extended to a satisfying assignment iff M on inputs α and $\bar{w}$ has a run from $\mathbf{u}$ to $\mathbf{v}$, using at most t steps and s space.*
- *The formula's primal graph has pathwidth $\mathcal{O}(s+h)$ and tree-depth $\mathcal{O}(s \cdot \log(n+s+t+h)+h)$.*

3 Connections with Tradeoffs for LCS

In this section we relate Conjecture 2 to statements of varying strength concerning different time-space tradeoffs. The results are summarized in Figure 1.

A *pl-reduction* between parameterized problems is an algorithm that transforms an instance of one problem with parameter k into an equivalent instance of another problem with parameter $k' \leq f(k)$, working in space $f(k)+\mathcal{O}(\log n)$, for some computable f. Following Elberfeld et al. [21] we define[4] $\mathbf{N}[f\text{poly}, f\log]$ as the class of parameterized problems that can be solved in non-deterministic time $f(k)\text{poly}(n)$ and space $f(k)\log(n)$ for some computable function f, where k is the parameter. Deterministic classes $\mathbf{D}[t, s]$ are defined analogously for various expressions t, s. All those mentioned in the article are closed under pl-reductions. We do not use the better known fpt-reductions because $\mathbf{N}[f\text{poly}, f\log]$ is not expected to be closed under them; its closure under fpt-reductions has been called WNL by Guillemot [27].

We use $o_{\text{eff}}(h(n))$ as an effective variant of $o(h(n))$: for $f, h : \mathbb{N} \to \mathbb{N}$ we write $f = o_{\text{eff}}(h)$ if there is a non-decreasing, unbounded, computable function $g(n)$ such that $f = \mathcal{O}(\frac{h}{g})$. The *inverse* of a function f is the function $f^{-1}(n) := \max\{i \mid f(i) \leq n\}$; observe that $f(f^{-1}(n)) \leq n \leq f^{-1}(f(n))$. Conjecture 2 concerns the following parameterized problem.

LCS	**Parameter:** k
Input: A finite alphabet Σ, k strings $s_1, s_2, \ldots, s_k$ over Σ, and an integer ℓ.	
Question: Is there a common subsequence of $s_1, s_2, \ldots, s_k$ of length at least ℓ?	

Elberfeld et al. [21], drawing on the work of Guillemot [27], pinpointed the complexity of LCS, allowing Conjecture 2 to be phrased as a general statement in parameterized complexity.

▶ **Theorem 12** ([21]). *LCS is complete for* $\mathbf{N}[f\text{poly}, f\log]$ *under pl-reductions.*

▶ **Corollary 13.** *Conjecture 2 holds if and only if* $\mathbf{N}[f\text{poly}, f\log] \not\subseteq \mathbf{D}[n^f, f\text{poly}]$.

Similarly as described in the introduction, the best known determinization results can only place $\mathbf{N}[f\text{poly}, f\log]$ in $\mathbf{D}[n^f, n^f]$ (commonly known as $\mathbf{XP}$) and $\mathbf{D}[n^{f(k)\cdot\log n}, f(k)\cdot\log^2 n]$.

Following Cai and Juedes [12], to relate parameterized tractability bounds to subexponential bounds, we define a reparameterized version of pw-3Coloring.

pw-3Coloring$^{\log n}$	**Parameter:** $s/\lg n$
Input: A graph G and a path decomposition of G of width s	
Question: Is G 3-colorable?	

Similarly as in Theorem 1, pathwidth-constrained problems turn out to be complete for non-deterministic computation with simultaneous time and space bounds.

▶ **Theorem 14** (♠). pw-3Coloring$^{\log n}$ *is complete for* $\mathbf{N}[f\text{poly}, f\log]$ *under pl-reductions.*

Containement follows from a poly-time, $\mathcal{O}(s + \log n)$-space non-deterministic algorithm that proceeds on consecutive bags of the decomposition, guessing each vertex color and remembering only those in the current bag. Completeness is proved with a direct application of Lemma 11 to a problem with parameter k solved in space $\mathcal{O}(f(k)\log n)$, yielding through Lemma 10 a pw-3Coloring instance of width $s = \mathcal{O}(f(k)\log n)$.

Conjecture 2 is thus equivalent to the statement that pw-3Coloring$^{\log n}$ is not in $\mathbf{D}[n^f, f\text{poly}]$, which gives Theorem 15.1. To contrast pathwidth with tree-depth, Lemma 20

[4] In this section, we only use time-space-bounded classes, hence we drop the subscripts for readability.

(introduced later) places td-3Coloring$^{\log n}$ in $\mathbf{D}[n^f, f\log]$, a class known as **XL**. Similarly as in the work of Cai and Juedes [12], we show that also subexponential bounds on the complexity of pw-3Coloring are related to the parameterized complexity of pw-3Coloring$^{\log n}$. This gives the sandwiching of Conjecture 2 between two similar statements in Theorem 15.2, 15.3. An even weaker statement is proved equivalent to $\mathbf{NL} \not\subseteq \mathbf{SC}$ by a simple padding argument in Theorem 15.4. For a somewhat less natural, stronger variant of Conjecture 2, we can show a similar, but exact correspondence in 15.5 (note the quasi-polynomial factor on both sides).

▶ **Theorem 15 (♠).** *Consider deterministic algorithms for* pw-3Coloring *working on instances of size n with a given path decomposition of width s (uniformly for all values of s).*

1. *There is no such algorithm working in time $n^{f(s/\lg n)}$ and space $f(s/\lg n)\mathrm{poly}(n)$ for any computable f if and only if Conjecture 2 holds.*
2. *Assuming Conjecture 2, there is no such algorithm working in time $2^{\mathcal{O}(s)}\mathrm{poly}(n)$ and space $2^{o_{\mathrm{eff}}(s)}\mathrm{poly}(n)$.*
3. *If Conjecture 2 fails, then for every unbounded, computable function g, there is such an algorithm working in time $2^{s\cdot g(s)}\mathrm{poly}(n)$ and space $2^{o_{\mathrm{eff}}(s)}\mathrm{poly}(n)$.*
4. *There is no such algorithm working in time $2^{\mathcal{O}(s)}\mathrm{poly}(n)$ and space $\mathrm{poly}(s, \log n)$ if and only if $\mathbf{NL} \not\subseteq \mathbf{SC}$.*
5. *There is no such algorithm working in time $2^{o_{\mathrm{eff}}(s^2)}n^{\mathcal{O}(\log n)}$ and space $2^{o_{\mathrm{eff}}(s)}\mathrm{poly}(n)$ if and only if $\mathbf{N}[f\mathrm{poly}, f\log] \not\subseteq \mathbf{D}[n^{f+\log}, f\mathrm{poly}]$.*

Proof of Theorem 15(2). Suppose to the contrary that pw-3Coloring can be solved in time $2^{\mathcal{O}(s)}\mathrm{poly}(n)$ and space $2^{o_{\mathrm{eff}}(s)}\mathrm{poly}(n)$. We show that $\mathbf{N}[f\mathrm{poly}, f\log] \subseteq \mathbf{D}[n^f, f\mathrm{poly}]$, contradicting Conjecture 2. The assumption implies that pw-3Coloring$^{\log n}$ can be solved in time $2^{\mathcal{O}(k\cdot\lg n)} = n^{\mathcal{O}(k)}$ and space $2^{k\cdot\lg n/g(k\cdot\lg n)}$ for some unbounded and non-decreasing computable function $g(\cdot)$. If $k \leq g(k\cdot\lg n)$, then the bound on space is bounded by n. Otherwise, if $k > g(k\lg n) \geq g(\lg n)$, then $n \leq 2^{g^{-1}(k)}$. In this case the bound on space is bounded by a computable function of k, namely $2^{k\cdot g^{-1}(k)}$. Hence in each case, the same algorithm solves pw-3Coloring$^{\log n}$ in time $n^{\mathcal{O}(k)}$ and space $n + 2^{k\cdot g^{-1}(k)}$. By Theorem 14, this implies $\mathbf{N}[f\mathrm{poly}, f\log] \subseteq \mathbf{D}[n^f, f\mathrm{poly}]$. ◀

We summarize the relationships around Conjecture 2 in Figure 1. The weakest statement there is $\mathbf{NL} \not\subseteq \mathbf{SC}$, a widely explored hypothesis in complexity theory. Since Directed (s,t)-Reachability (asking given a directed graph and two nodes s, t, whether is t reachable from s) is an **NL**-complete problem, this is also equivalent to the question of whether this problem can be decided in polynomial time and polylogarithmic space. However, even this weakest statement is not known to be implied by better known conjectures such as the Exponential Time Hypothesis. It seems that the simultaneous requirement on bounding two complexity measures—time and space—has a nature independent of the usual time complexity considerations. Hence, new assumptions may be needed to explore this paradigm, and we hope that Conjecture 2 may serve as a transparent and robust example of such.

In a certain restricted computation model (allowing operations on graph nodes only, not on individual bits), unconditional tight lower bounds have been proved by Edmonds et al. [18]: it is impossible to decide Directed (s,t)-Reachability in time $2^{o(\log^2 n)}$ and space $\mathcal{O}(n^{1-\varepsilon})$ (for any $\varepsilon > 0$), even if randomization is allowed. Essentially all known techniques for solving Directed (s,t)-Reachability are known to be implementable in this model [35] (including DFS, BFS, theorems of Savitch, of Immerman and Szelepcsényi, as well as Reingold's breakthrough), therefore this strongly suggests that no algorithm running in time $2^{o_{\mathrm{eff}}(\log^2 n)}$ and space $n^{o_{\mathrm{eff}}(1)}$ is possible, that is, $\mathbf{NL} \not\subseteq \mathbf{D}[\underset{time}{2^{o_{\mathrm{eff}}(\log^2 n)}}, \underset{space}{n^{o_{\mathrm{eff}}(1)}}]$.

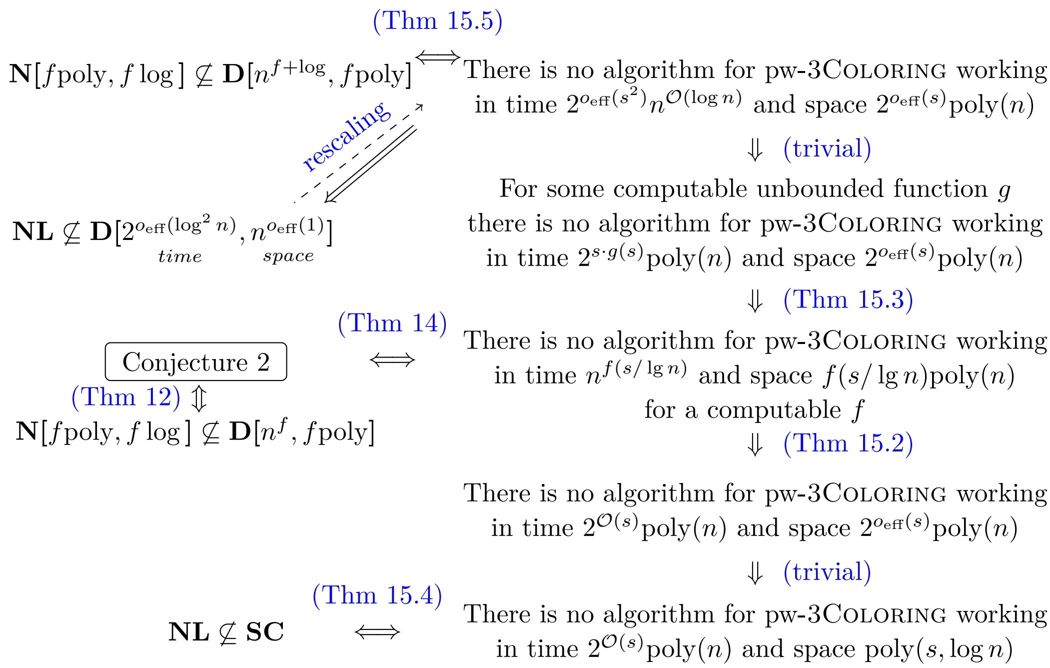

Figure 1 A summary of the relationships between various statements related to Conjecture 2.

By Theorem 1, this is equivalent to saying that pw-3Coloring[log] cannot be solved in these time and space bounds. The strongest statement on Figure 1 is a rescaling of this, that is, it implies $\mathbf{NL} \not\subseteq \mathbf{D}[\underset{time}{2^{o_{\mathrm{eff}}(\log^2 n)}}, \underset{space}{n^{o_{\mathrm{eff}}(1)}}]$ by a trivial padding argument, but the reverse implication is also probable in the sense that any proof of the latter would likely scale to prove the former. However, it is still possible that an algorithm working in polynomial space refutes the stronger statement even though $\mathbf{NL} \not\subseteq \mathbf{D}[\underset{time}{2^{o_{\mathrm{eff}}(\log^2 n)}}, \underset{space}{n^{o_{\mathrm{eff}}(1)}}]$.

4 Treedepth

In this section we sketch the proof of Theorem 3. Let $s : \mathbb{N} \to \mathbb{N}$ be a nice function. First, we discuss more precisely the model of machines used to define class $\mathbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s}]$. The machine has three tapes, each using a fixed, finite alphabet Σ: a read-only input tape, a working tape of length $\mathcal{O}(\log n)$, and a stack tape of length $s(n)$. On each of the tapes there is a head; the transitions of the machine depend on its state and the triple of symbols under the heads. The input tape is read-only. The stack tape can be read but not freely written on; instead, the transitions of the machine may contain instructions `push` σ or `pop`, working naturally. Since s is nice, $s(n) \le \mathrm{poly}(n)$ so within the working tape the machine can keep track of the current height of the stack and the indices on which the heads are positioned.

We start the proof of Theorem 3 by showing containment, exemplifying how the resources are used. The idea is to perform a depth-first search of the tree-depth decomposition, guessing the color of each entered vertex, pushing it onto the stack and popping it when withdrawing from the vertex. Thus, the stack maintains the guessed colors on the path from the current vertex to the root, allowing correctness to be checked.

▶ **Lemma 16 (♠).** *For any nice $s(n)$,* td-3COLORING$[s]$ *is in* $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s}]$.

Clearly if $s(n) \geq \log^2 n$, $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s}] \subseteq \mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{s/\log}, \underset{height}{s}]$. The next step is to show how the stack operations of the latter class' machines can be regularized. This idea originates in the approach of Akatov and Gottlob [3]. Following [3], we define a regular stack machine in the following way. For any valid sequence S of push/pop operations that starts and ends with an empty stack, define the corresponding push-pop tree $\tau(S)$ to be the ordered tree (a rooted tree with an order imposed on the children of each node) in which a depth-first search would result in the sequence S, where entering/withdrawing from a vertex corresponds to a push/pop operation. We say that a language is in reg-$\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{s/\log}, \underset{height}{s}]$ if it is recognized by an NTM M with $s(n)/\log(n)$ working space and an auxiliary stack of height $s(n)$ that has the following properties:

(1) M pushes and pops blocks of $\mathbf{b} = \lceil s(n)/\lg(n) \rceil$ symbols at a time, say, simultaneously from/to the first $\mathbf{b}$ positions of the worktape.

(2) Whenever M decides to push or pop, it can only change its state. Moreover, the decision about using a push or pop transition depends only on the machine's state.

(3) If M accepts input α, then there is a run on α where the push-pop tree (where pushing/popping a block is considered atomic) is the full binary tree of depth exactly $c\lceil \lg n \rceil$, for some fixed integer c. In particular, at the moment of accepting the stack is empty.

Restriction (2) is a technical adjustment. Restriction (1) is easily achieved by simulating the top symbols from the stack in a length $\mathbf{b}$ buffer on the working tape, pushing and popping a full buffer when needed. The most important restriction is (3): the push-pop tree has a fixed shape of a full binary tree. For this, we use the following observation of Akatov and Gottlob [3, 2], used also by Elberfeld et al. [20]. The *traversal ordering* of the nodes of an ordered tree is the linear ordering which places a parent before its children and, for children a, b of a node, a occurring before b, places all descendants of a before all descendants of b.

▶ **Lemma 17** (Lemma 3.3 of [2]; Theorem 3.14 of [20])**.** *Given an ordered tree T with n nodes and depth at most $\lg n$, one can in logarithmic space compute an embedding (an injection that preserves the ancestor relation and traversal ordering) into a full binary tree of depth $4 \lg n$.*

As in [3], this allows us to regularize our machines, as dummy pushes/pops can be non-deterministically guessed so that the push-pop tree of at least one run is a full binary tree.

▶ **Lemma 18 (♠).** $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{s/\log}, \underset{height}{s}] \subseteq$ *reg-*$\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{s/\log}, \underset{height}{s}]$.

Knowing that computations for NAuxSA can be conveniently regularized, we can describe the existence of such a computation by a CNF formula "wrapped around" the rigid shape of the full binary tree that encodes the push-pop tree of the run. We think of the computation as starting at the root node, moving down an edge whenever a push is made and moving up an edge whenever a pop is made. This was also the idea of Akatov and Gottlob [3], but our reduction needs to introduce many more elements, in particular copies of the gadget of Lemma 11 for every fragment between two push/pop operations. Each part of the computation depends only on symbols pushed onto the stack on the path to the root. This, together with the $\mathcal{O}(\frac{s(n)}{\log n} \cdot \log n) = \mathcal{O}(s(n))$ bound on the tree-depth of the computation gadget, will give rise to a tree-depth decomposition of depth $\mathcal{O}(s(n))$ of the obtained formula's primal graph.

▶ **Lemma 19** (♠). *If* $L \in$ *reg*-$\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{s/\log}, \underset{height}{s}]$, *then* $L \leq_L$ *td-CNF-SAT*$[s]$.

Lemmas 18 and 19 show that td-CNF-SAT$[s]$ is hard for $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{s/\log}, \underset{height}{s}]$, and by Lemma 10 so is td-3Coloring$[s]$. Since the closure of $\mathbf{NAuxSA}[\text{poly}, \log, s]$ under logspace reductions is $\mathbf{NAuxSA}[\text{poly}, \log, s(\text{poly})]$, Lemmas 16, 18, 19 give the following chain of containments (here $[A]^L$ denotes the class of problems reducible to A in logspace):

$$\begin{aligned}[\text{td-3Coloring}[s(\text{poly})]]^L &\subseteq [\text{td-3Coloring}[s]]^L \subseteq \mathbf{NAuxSA}[\text{poly}, \log, s(\text{poly})] \\ &\subseteq \mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{s(\text{poly})/\log}, \underset{height}{s(\text{poly})}] \\ &\subseteq [\text{td-3Coloring}[s(\text{poly})]]^L\end{aligned}$$

Therefore, all containments must be equalities, which concludes the proof of Theorems 3 and 4. Now, to prove the determinization of Theorem 5, we only need an algorithm for td-3Coloring$[s]$. The following lemma implies td-3Coloring$[s] \in \mathbf{D}[\underset{space}{s}]$ (for nice s), hence Theorem 3, and the fact that $\mathbf{D}[\underset{space}{s(\text{poly})}]$ is closed under logspace, yield Theorem 5.

▶ **Lemma 20** (♠). td-3Coloring$[s]$ *can be solved in time* $3^s \cdot \text{poly}(n)$ *and space* $\mathcal{O}(s + \log n)$.

Characterization via alternating machines. In the full version of this paper, we use Theorem 3 to give another characterization in terms of alternating Turing machines with polynomial size of an accepting tree, i.e. *treesize*. Both the notions of ATMs and that of treesize later introduced by Ruzzo [42] gave a new unified view on various complexity classes, simplifying a few containment proofs. Ruzzo showed that $\mathbf{NAuxPDA}[\underset{time}{\text{poly}}, \underset{space}{s}] = \mathbf{A}[\underset{space}{s}, \underset{treesize}{\text{poly}}]$. We show that bounding the time (as opposed to space) of a polynomial treesize ATM, leads to the classes corresponding to small tree-depth, as opposed to small treewidth.

▶ **Theorem 21** (♠). *Let* $s(n) \geq \log^2(n)$ *be a nice function. Then*

$$\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s(\text{poly})}] = \mathbf{A}[\underset{time}{s(\text{poly})}, \underset{treesize}{\text{poly}}].$$

5 Conclusions

Let $s(n) \geq \log^2 n$ be a nice function such that $\forall_c\ s(n^c) = \mathcal{O}(s(n))$ (e.g. $s(n) = \lg^k n$, $k \geq 2$). The hierarchy of graph parameters of Corollary 8 together with Theorems 1, 3, and 21 implies the following hierarchy of complexity classes between **NL** and **NP**.

$$\begin{array}{ccccccc}
\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s}] & = & [\text{td-3Coloring}[s]]^L & = & \mathbf{A}[\underset{time}{s}, \underset{treesize}{\text{poly}}] & \subseteq & \mathbf{D}[\underset{space}{s}] \\
 & & \cap\!\!| & & & & \\
\mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{s}] & = & [\text{pw-3Coloring}[s]]^L & = & \mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{s}] & & \\
 & & \cap\!\!| & & & & \\
\mathbf{NAuxPDA}[\underset{time}{\text{poly}}, \underset{space}{s}] & = & [\text{tw-3Coloring}[s]]^L & = & \mathbf{A}[\underset{space}{s}, \underset{treesize}{\text{poly}}] & \subseteq & \mathbf{D}[\underset{time}{2^{\mathcal{O}(s)}}] \\
 & & \cap\!\!| & & & & \\
\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s \cdot \log}] & = & [\text{td-3Coloring}[s \cdot \log]]^L & = & \mathbf{A}[\underset{time}{s \cdot \log}, \underset{treesize}{\text{poly}}] & \subseteq & \mathbf{D}[\underset{space}{s \cdot \log}]
\end{array}$$

For $s(n) = \log^k(n)$, the classes have been considered under different names:

- $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{\log^k}]$ was named $\mathbf{DC}^{k-1}$ (for *divide and conquer*) in [3, 2],
- $\mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{\log^k}]$ are known as $\mathbf{NSC}^k$ (the non-deterministic variant of *Steve's Class*),
- $\mathbf{NAuxPDA}[\underset{time}{\text{poly}}, \underset{space}{\log^k}]$ is shown equal to a class named $\mathbf{SAC}^k_{quasi}$ in [4].

This yields the following hierarchy:

$$\mathbf{L} \subseteq \begin{array}{c}\mathbf{NL}\\ \| \\ \mathbf{NSC}^1\end{array} \subseteq \begin{array}{c}\mathbf{SAC}^1\\ \| \\ \mathbf{SAC}^1_{quasi}\end{array} \subseteq \mathbf{DC}^1 \subseteq \cdots \subseteq \mathbf{DC}^{k-1} \subseteq \mathbf{NSC}^k \subseteq \mathbf{SAC}^k_{quasi} \subseteq \mathbf{DC}^k \subseteq \cdots \subseteq \mathbf{NP}$$

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78. IEEE Computer Society, 2015.

2 Dmitri Akatov. *Exploiting parallelism in decomposition methods for constraint satisfaction*. PhD thesis, University of Oxford, 2010.

3 Dmitri Akatov and Georg Gottlob. Balanced queries: Divide and conquer. In *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 42–54. Springer, 2010. doi:10.1007/978-3-642-15155-2_6.

4 Eric Allender, Shiteng Chen, Tiancheng Lou, Periklis A. Papakonstantinou, and Bangsheng Tang. Width-parametrized SAT: time–space tradeoffs. *Theory of Computing*, 10:297–339, 2014. doi:10.4086/toc.2014.v010a012.

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

6 Per Austrin, Petteri Kaski, Mikko Koivisto, and Jussi Määttä. Space-time tradeoffs for subset sum: An improved worst case algorithm. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 45–56. Springer, 2013.

7 Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994. doi:10.1145/174644.174650.

8 Marina Barsky, Ulrike Stege, Alex Thomo, and Chris Upton. Shortest path approaches for the longest common subsequence of a set of strings. In *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering, BIBE 2007, October 14-17, 2007, Harvard Medical School, Boston, MA, USA*, pages 327–333. IEEE Computer Society, 2007. doi:10.1109/BIBE.2007.4375584.

9 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *CoRR*, abs/1007.1161, 2010. URL: http://arxiv.org/abs/1007.1161.

10 Hans L. Bodlaender, Jitender S. Deogun, Klaus Jansen, Ton Kloks, Dieter Kratsch, Haiko Müller, and Zsolt Tuza. Rankings of graphs. *SIAM J. Discrete Math.*, 11(1):168–181, 1998. doi:10.1137/S0895480195282550.

11 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Harold T. Wareham. The parameterized complexity of sequence alignment and consensus. *Theor. Comput. Sci.*, 147(1&2):31–54, 1995. doi:10.1016/0304-3975(94)00251-D.

12 Liming Cai and David W. Juedes. On the existence of subexponential parameterized algorithms. *J. Comput. Syst. Sci.*, 67(4):789–807, 2003. doi:10.1016/S0022-0000(03)00074-6.

13 Hubie Chen and Moritz Müller. One hierarchy spawns another: graph deconstructions and the complexity classification of conjunctive queries. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS'14, Vienna, Austria, July 14-18, 2014*, pages 32:1–32:10. ACM, 2014. doi:10.1145/2603088.2603107.

14 Bruno Courcelle. The Monadic Second-Order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

15 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

16 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.

17 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

18 Jeff Edmonds, Chung Keung Poon, and Dimitris Achlioptas. Tight lower bounds for st-connectivity on the NNJAG model. *SIAM J. Comput.*, 28(6):2257–2284, 1999. doi:10.1137/S0097539795295948.

19 Michael Elberfeld, Martin Grohe, and Till Tantau. Where first-order and monadic second-order logic coincide. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 265–274. IEEE Computer Society, 2012.

20 Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of bodlaender and courcelle. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:62, 2010. Extended abstract included in the proceedings of FOCS 2010.

21 Michael Elberfeld, Christoph Stockhusen, and Till Tantau. On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica*, 71(3):661–701, 2015. doi:10.1007/s00453-014-9944-y.

22 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1 edition, 2006.

23 Fedor V. Fomin, Petteri Kaski, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Parameterized single-exponential time polynomial space algorithm for Steiner Tree. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 494–505. Springer, 2015.

24 Martin Fürer and Huiwen Yu. Space saving by dynamic algebraization. In *Computer Science – Theory and Applications – 9th International Computer Science Symposium in Russia, CSR 2014, Moscow, Russia, June 7-11, 2014. Proceedings*, volume 8476 of *Lecture Notes in Computer Science*, pages 375–388. Springer, 2014.

25 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

26 Georg Gottlob, Nicola Leone, and Francesco Scarcello. The complexity of acyclic conjunctive queries. *J. ACM*, 48(3):431–498, 2001. doi:10.1145/382780.382783.

27 Sylvain Guillemot. Parameterized complexity and approximability of the longest compatible sequence problem. *Discrete Optimization*, 8(1):50–60, 2011. doi:10.1016/j.disopt.2010.08.003.

28 Meir Katchalski, William McCuaig, and Suzanne M. Seager. Ordered colourings. *Discrete Mathematics*, 142(1-3):141–154, 1995.

29 Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. doi:10.1007/BFb0045375.

30 Alexander Langer, Felix Reidl, Peter Rossmanith, and Somnath Sikdar. Practical algorithms for MSO model-checking on tree-decomposable graphs. *Computer Science Review*, 13-14:39–74, 2014. doi:10.1016/j.cosrev.2014.08.001.

31 Richard J Lipton. Savitch's theorem. In *The P=NP Question and Gödel's Lost Letter*, pages 135–138. Springer, 2010.

32 Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980.

33 Daniel Lokshtanov, Matthias Mnich, and Saket Saurabh. Planar k-path in subexponential time and polynomial space. In *Graph-Theoretic Concepts in Computer Science – 37th International Workshop, WG 2011, Teplá Monastery, Czech Republic, June 21-24, 2011. Revised Papers*, volume 6986 of *Lecture Notes in Computer Science*, pages 262–270. Springer, 2011. doi:10.1007/978-3-642-25870-1_24.

34 Daniel Lokshtanov and Jesper Nederlof. Saving space by algebraization. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 321–330. ACM, 2010. doi:10.1145/1806689.1806735.

35 Pinyan Lu, Jialin Zhang, Chung Keung Poon, and Jin-yi Cai. Simulating undirected *st*-connectivity algorithms on uniform jags and nnjags. In *Algorithms and Computation, 16th International Symposium, ISAAC 2005, Sanya, Hainan, China, December 19-21, 2005, Proceedings*, volume 3827 of *Lecture Notes in Computer Science*, pages 767–776. Springer, 2005. doi:10.1007/11602613_77.

36 Burkhard Monien and Ivan Hal Sudborough. Bandwidth constrained NP-complete problems. *Theor. Comput. Sci.*, 41:141–167, 1985. doi:10.1016/0304-3975(85)90068-4.

37 Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013. doi:10.1007/s00453-012-9630-x.

38 J. Nešetřil and P. Ossona de Mendez. *Sparsity: Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer, 2012.

39 Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet Shortest Common Supersequence and Longest Common Subsequence problems. *J. Comput. Syst. Sci.*, 67(4):757–771, 2003.

40 Alex Pothen. The complexity of optimal elimination trees, 1988. Technical Report CS 88-16, Pennsylvania State University.

41 Neil Robertson and Paul D. Seymour. Graph Minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

42 Walter L. Ruzzo. Tree-size bounded alternation. *J. Comput. Syst. Sci.*, 21(2):218–235, 1980. doi:10.1016/0022-0000(80)90036-7.

43 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

44 V. Vinay and V. Chandru. The expressibility of nondeterministic auxiliary stack automata and its relation to treesize bounded alternating auxiliary pushdown automata. In *Foundations of Software Technology and Theoretical Computer Science, Tenth Conference, Bangalore, India, December 17-19, 1990, Proceedings*, volume 472 of *Lecture Notes in Computer Science*, pages 104–114. Springer, 1990. doi:10.1007/3-540-53487-3_38.