THE PARAMETERIZED SPACE COMPLEXITY OF
MODEL-CHECKING BOUNDED VARIABLE FIRST-ORDER LOGIC *

YIJIA CHEN [a,1], MICHAEL ELBERFELD [b], AND MORITZ MÜLLER [c,2]

[a] School of Computer Science, Fudan University, China
  e-mail address: yijiachen@fudan.edu.cn

[b] Lehrstuhl für Informatik 7, RWTH Aachen, Germany
  e-mail address: elberfeld@informatik.rwth-aachen.de

[c] Department of Computer Science, Universitat Politècnica de Catalunya, Spain
  e-mail address: moritz@cs.upc.edu

ABSTRACT. The parameterized model-checking problem for a class of first-order sentences (queries) asks to decide whether a given sentence from the class holds true in a given relational structure (database); the parameter is the length of the sentence. We study the parameterized space complexity of the model-checking problem for queries with a bounded number of variables. For each bound on the quantifier alternation rank the problem becomes complete for the corresponding level of what we call the tree hierarchy, a hierarchy of parameterized complexity classes defined via space bounded alternating machines between parameterized logarithmic space and fixed-parameter tractable time. We observe that a parameterized logarithmic space model-checker for existential bounded variable queries would allow to improve Savitch's classical simulation of nondeterministic logarithmic space in deterministic space $O(\log^2 n)$. Further, we define a highly space efficient model-checker for queries with a bounded number of variables and bounded quantifier alternation rank. We study its optimality under the assumption that Savitch's Theorem is optimal.

1. INTRODUCTION

The model-checking problem MC(FO) for first-order logic FO asks whether a given first-order sentence φ holds true in a given relational structure $\mathbf{A}$. This problem is of paramount importance throughout computer science, and especially in database theory [SSS10]. The problem is PSPACE-complete in general and even its restriction to primitive positive sentences and two-element structures stays NP-hard (cf. [CM77]). Hence neither syntactic nor

structural restrictions seem to allow to get a handle on this problem. An important exception is the observation that the natural bottom-up evaluation algorithm takes only polynomial time on sentences with a bounded number of variables (see [Var95, Proposition 3.1], or the proof of [Imm82, Theorem B.5]). Indeed, this algorithm runs in time $O(|\varphi| \cdot |\mathbf{A}|^s)$ on instances $(\mathbf{A}, \varphi)$ where φ is in FO^s, i.e. contains at most s many variables.

Following Chandra and Merlin's seminal paper [CM77, Section 4], it has repeatedly been argued in the literature (see e.g. [SSS10, FG01]) that measuring computational resources needed to solve MC(FO) by functions in the length of the input only is unsatisfactory. It neglects the fact that in typical situations in database theory we are asked to evaluate a relatively short φ (the query) against a relatively large $\mathbf{A}$ (the database). Parameterized complexity theory measures computational resources by functions taking as an additional argument a *parameter* associated to the problem instance. For the parameterized model-checking problem p-MC(FO) one takes the length $|\varphi|$ of φ as parameter and asks for fixed-parameter tractable restrictions of p-MC(FO), i.e. restrictions decidable in *fpt time* $f(|\varphi|) \cdot |\mathbf{A}|^{O(1)}$ where $f : \mathbb{N} \to \mathbb{N}$ is an arbitrary computable function.

This relaxed tractability notion allows for effective but inefficient formula manipulations and thereby the transfer of logical methods to the study of the problem (see [Gro14, Gro08, GK11] for surveys). For example, a sequence of works constructing algorithms exploiting Gaifman's locality theorem [EF99, Theorem 2.5.1] recently led to an fpt time algorithm solving p-MC(FO) on nowhere dense graph classes [GKS17]. For graphs of bounded degree one even gets an algorithm that runs in parameterized logarithmic space [FG03]. Parameterized logarithmic space, denoted by para-L, relaxes logarithmic space in much the same way as FPT relaxes polynomial time [CCDF97, FG03].

Concerning restrictions on the syntactical side one can naturally stratify the problem into subproblems p-MC(Σ_1), p-MC(Σ_2), ..., p-MC(FO) according to quantifier alternation rank. These problems stay intractable in the sense that they are well known to be complete for the levels of the A-hierarchy A[1] $\subseteq$ A[2] $\subseteq \cdots \subseteq$ AW[$*$]. The completeness stays true even in the presence of function symbols [FG06, p. 201].

One of the main research questions is to understand for which sets Φ of first-order sentences, the problem p-MC(Φ) is tractable. In this introductory exposition let us restrict attention to decidable sets Φ in a (relational) vocabulary that has bounded arity.

Today, the situation is well understood for sets Φ of primitive positive sentences, i.e. *conjunctive queries* in database terminology [CM77]. Indeed, under the assumption A[1] $\neq$ FPT, the model-checking problem p-MC(Φ) is fixed-parameter tractable if [DKV02] and only if [GSS01, Gro07] there is some constant $s \in \mathbb{N}$ such that every query in Φ is logically equivalent to a conjunctive query whose primal graph has treewidth at most $s + 1$. Now, these are precisely those conjunctive queries that can be written with at most s many variables [KV00, Lemma 5.2, Remark 5.3] (see also [DKV02, Theorem 12] and [Che14, Theorem 4] for similar statements). In other words, if p-MC(Φ) is fixed-parameter tractable at all, then it can be decided by first preprocessing the query to one with a bounded number of variables and then run the natural evaluation algorithm. As pointed out in [Che14], this is a recurring paradigm. In fact, most known sets Φ (not necessarily primitive positive) with a tractable p-MC(Φ) are contained in FO^s up to logical equivalence.

This gives special interest to the computational complexity of p-MC(FO^s). Moreover, already for $s = 2$ this problem encompasses problems of independent interest. For example, a directed graph with two vertices named by constants s and t contains a path from s to t of length at most k if and only if it satisfies the sentence $\varphi_k(s) \in \mathrm{FO}^2$ (allowing constants s, t)

where $\varphi_k(x)$ is defined as follows:

$$
\begin{aligned}
\varphi_0(x) &:= x{=}t, \\
\varphi_{k+1}(x) &:= \exists y\Big(\big(y{=}x \vee Exy\big) \wedge \exists x\big(x{=}y \wedge \varphi_k(x)\big)\Big)
\end{aligned}
\tag{1.1}
$$

There is some recent work concerning the fine-grained time complexity of $p\text{-}\mathrm{MC}(\mathrm{FO}^s)$, see [GIKW19]. But given the central importance of $p\text{-}\mathrm{MC}(\mathrm{FO}^s)$ it seems surprising that, to the best of our knowledge, its space complexity has not been thoroughly studied. It is known that $\mathrm{MC}(\mathrm{FO}^s)$ is P-complete for $s \geq 2$ under logarithmic space reductions (see [Imm82, Var95]). But, even assuming $\mathrm{P} \neq \mathrm{L}$, this leaves open the possibility that $p\text{-}\mathrm{MC}(\mathrm{FO}^s)$ could be solved in *parameterized logarithmic space*, that is, (deterministic) space $f(|\varphi|) + O(\log |\mathbf{A}|)$ for some computable $f : \mathbb{N} \to \mathbb{N}$. The central question of this paper is whether this is the case.[1]

As we shall see, answering this question either positively or negatively would have breakthrough consequences in classical complexity theory. It is one of the central open questions or, in Lipton's words [Lip10, p.137], "one of the biggest embarrassments of all complexity theory" whether Savitch's upper bound $\mathrm{NL} \subseteq \mathrm{SPACE}(\log^2 n)$ can be improved. This is open since 1969 and there is a tangible possibility that *Savitch's Theorem is optimal*, i. e.[2]

$$\mathrm{NL} \not\subseteq \mathrm{SPACE}(o(\log^2 n)).$$

See [HOT94, Pot17] for more information about this problem. We observe the following implications:

$$\mathrm{P} = \mathrm{L} \quad \Longrightarrow \quad p\text{-}\mathrm{MC}(\mathrm{FO}^s) \in \text{para-L} \quad \Longrightarrow \quad \text{Savitch's Theorem is not optimal.}$$

For the second implication, note that running the assumed model-checker on the sentences (1.1) solves the parameterized problem

$p\text{-}\mathrm{STCON}_{\leq}$	
Instance:	A directed graph $\mathbf{G}$, two vertices s and t, and $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Is there a directed path of length at most k from s to t in $\mathbf{G}$?

On n-node graphs this requires only space $f(k) + O(\log n)$ for some computable f. Then there is a logspace computable unbounded function h such that the algorithm runs in logarithmic space on instances with $k \leq h(n)$. It is well known that such an algorithm implies that Savitch's theorem is not optimal (see e.g. [Wig92]). We shall give the details of this argument, and in fact prove something stronger. More precisely, the contributions of this paper are as follows.

Our contributions. Our central question asks for the complexity of $p\text{-}\mathrm{MC}(\mathrm{FO}^s)$ up to parameterized logarithmic space reductions, i. e. *pl-reductions*. An obvious approach is to stratify the problem again according to quantifier alternation rank into subproblems $p\text{-}\mathrm{MC}(\Sigma_1^s)$, $p\text{-}\mathrm{MC}(\Sigma_2^s), \ldots$. It turns out that there exist pl-reductions from $p\text{-}\mathrm{MC}(\mathrm{FO}^s)$ to $p\text{-}\mathrm{MC}(\mathrm{FO}^2)$, and for each fixed t from $p\text{-}\mathrm{MC}(\Sigma_t^s)$ to $p\text{-}\mathrm{MC}(\Sigma_t^2)$ (see Theorem 1.1 below). On the other hand we do not know whether one can reduce $p\text{-}\mathrm{MC}(\Sigma_{t+1}^2)$ to $p\text{-}\mathrm{MC}(\Sigma_t^s)$ for any s.

---

[1] Relatedly, it has been asked by Flum and Grohe [FG03, Remark 26] whether $p\text{-}\mathrm{MC}(\mathrm{FO}^s)$ is in para-NL.

[2] By the space hierarchy theorem $\mathrm{SPACE}(s(n)) \neq \mathrm{SPACE}(\log^2 n)$ holds for the Turing machine model and for all (not necessarily space constructible) $s(n) \leq o(\log^2 n)$.

Clearly, all these problems lie between para-L and FPT. But unfortunately not much is known about this complexity landscape, and only recently there is a slowly emerging picture [EST15, CM15, CM17, CM14]. In particular, [EST15] introduced a parameterized analogue of NL, called PATH in [CM15], and [CM15] introduced the class TREE, a parameterized analogue of LOGCFL. Unlike their classical counterparts [Sze87, Imm88, BCD$^+$89], PATH and TREE are not known to be closed under complementation (this is a question from [CM15]). We thus face the naturally defined alternation hierarchy built above TREE (see Definition 3.1). We call it the *tree hierarchy*:

$$\text{para-L} \subseteq \text{PATH} \subseteq \text{TREE} = \text{TREE}[1] \subseteq \text{TREE}[2] \subseteq \cdots \subseteq \text{TREE}[*] \subseteq \text{FPT}. \qquad (1.2)$$

Our first result reads as follows:

Theorem 1.1. *Let $s \geq 2$ and $t \geq 1$.*
(1) *$p\text{-MC}(\Sigma_t^s)$ is complete for TREE$[t]$ under pl-reductions.*
(2) *$p\text{-MC}(\text{FO}^s)$ is complete for TREE$[*]$ under pl-reductions.*

We shall also prove that these results stay true in the presence of function symbols (Theorem 4.6). This situation within FPT is thus fully analogous to the situation with unboundedly many variables and the A-hierarchy. The proofs, however, are quite different. A further difference is that the tree hierarchy satisfies a collapse theorem (Corollary 4.2) like the polynomial hierarchy while such a theorem is unknown for the A-hierarchy.

The connection to the classical question whether Savitch's Theorem is optimal reads, more precisely, as follows:

Theorem 1.2. *If Savitch's Theorem is optimal, then PATH $\not\subseteq$ para-L.*

Concerning our central question, these results together with (1.2) imply that, if Savitch's Theorem is optimal, then already the lowest level $p\text{-MC}(\Sigma_1^2)$ cannot be solved in parameterized logarithmic space. In fact, we show something stronger:

Theorem 1.3. *Let f be an arbitrary function from $\mathbb{N}$ to $\mathbb{N}$. If Savitch's Theorem is optimal, then $p\text{-MC}(\Sigma_1^2)$ is not decidable in space $o(f(|\varphi|) \cdot \log |\mathbf{A}|)$.*

A straightforward algorithm solves $p\text{-MC}(\text{FO}^s)$ in space (cf. Lemma 6.2)

$$O(|\varphi| \cdot (\log |\varphi| + \log |\mathbf{A}|)).$$

Theorem 1.3 shows that there might be not too much room for improvement. However, building on ideas of Ruzzo [Ruz80], we still manage to give a significant improvement for bounded quantifier alternation rank:

Theorem 1.4. *For all $s, t \in \mathbb{N}$ the problem $p\text{-MC}(\Sigma_t^s)$ is decidable in space*

$$O(\log |\varphi| \cdot (\log |\varphi| + \log |\mathbf{A}|)).$$

It is unlikely that the bound on the quantifier alternation rank can be omitted in this statement. Indeed, it follows from the P-completeness result mentioned above that $p\text{-MC}(\text{FO}^2)$ cannot be decided in the displayed space unless P $\subseteq$ SPACE$(\log^2 n)$.

2. Preliminaries

For a natural number $n \in \mathbb{N}$ we set $[n] = \{1, \ldots, n\}$ understanding $[0] = \emptyset$.

2.1. **Structures.** A *vocabulary* τ is a finite set of relation and function symbols. Relation and function symbols have an associated *arity*, a natural number. A τ-*structure* $\mathbf{A}$ consists of a finite nonempty set A, its *universe*, and for each r-ary relation symbol $R \in \tau$ an *interpretation* $R^{\mathbf{A}} \subseteq A^r$ and for each r-ary function symbol $f \in \tau$ an *interpretation* $f^{\mathbf{A}} : A^r \to A$. A *constant* is a function symbol c of arity 0. We identify its interpretation $c^{\mathbf{A}}$ with its unique value, an element of A.

A *directed graph* is an $\{E\}$-structure $\mathbf{G} = (G, E^{\mathbf{G}})$ for the binary relation symbol E. We refer to elements of G as *vertices* and to elements $(a, b) \in E^{\mathbf{G}}$ as *(directed) edges (from a to b)*. Note this allows $\mathbf{G}$ to have self-loops, i.e. edges from a to a. A *graph* is a directed graph $\mathbf{G} = (G, E^{\mathbf{G}})$ with irreflexive and symmetric $E^{\mathbf{G}}$. A *(directed) path* in a (directed) graph $\mathbf{G}$ is a sequence $(a_1, \ldots, a_{k+1})$ of pairwise distinct vertices such that $(a_i, a_{i+1}) \in E^{\mathbf{G}}$ for all $i \in [k]$; the path is said to have *length k* and to be *from a_1 to a_k*. Note that there is a path of length 0 from every vertex to itself.

The *size* of a τ-structure $\mathbf{A}$ is

$$|\mathbf{A}| := |\tau| + |A| + \sum_R |R^{\mathbf{A}}| \cdot ar(R) + \sum_f |A|^{ar(f)},$$

where R, f range over the relation and function symbols of τ respectively, and $ar(R), ar(f)$ denote the arities of R, f respectively. For example, the size of a (directed) graph with n vertices and m edges is $O(n+m)$. Note that a reasonable binary ("sparse" or "list") encoding of $\mathbf{A}$ has length $O(|\mathbf{A}| \cdot \log |A|)$. The difference between the size as defined and the length of the binary encoding of a structure plays no role in this paper.

2.2. **Formulas.** Let τ be a vocabulary. A τ-*term* is a variable, or of the form $ft_1 \cdots t_r$ where f is an r-ary function symbol and $t_1, \ldots, t_r$ are again τ-terms. *Atomic τ-formulas*, i.e. τ-*atoms*, have the form $t{=}t'$ or $R(t_1, \ldots, t_r)$ where R is an r-ary relation symbol in τ and $t, t', t_1, \ldots, t_r$ are τ-terms. General τ-*formulas* are built from atomic ones by $\wedge, \vee, \neg$ and universal and existential quantification $\forall x, \exists x$. The vocabulary τ is *relational* if it contains only relation symbols. For a tuple of variables $\bar{x} = (x_1, \ldots, x_k)$ we write $\varphi = \varphi(\bar{x})$ to indicate that the free variables of φ are among $\{x_1, \ldots, x_k\}$. If $\mathbf{A}$ is a τ-structure and $\bar{a} = (a_1, \ldots, a_k) \in A^k$, then $\mathbf{A} \models \varphi(\bar{a})$ means that the assignment that maps x_i to a_i for $i \in [k]$ satisfies φ in $\mathbf{A}$. A *sentence* is a formula without free variables. The *size* $|\varphi|$ of a formula φ is the length of a reasonable binary encoding of it.

For $s \in \mathbb{N}$ let FO^s denote the class of (first-order) formulas *over a relational vocabulary* containing at most s variables (free or bound). For $t \in \mathbb{N}$ we define the classes Σ_t and Π_t as follows. Both Σ_0 and Π_0 are the class of quantifier free formulas; Σ_{t+1} (resp. Π_{t+1}) is the closure of Π_t (resp. Σ_t) under positive Boolean combinations (i.e. applying $\vee, \wedge$) and existential (resp. universal) quantification. We set

- $\Sigma_t^s := \mathrm{FO}^s \cap \Sigma_t$,
- $\Pi_t^s := \mathrm{FO}^s \cap \Pi_t$.

Example 2.1. The formulas (1.1) in the introduction do not qualify as FO^2 because they use constants s, t. They are evaluated in structures $\mathbf{G} = (G, E^{\mathbf{G}}, s^{\mathbf{G}}, t^{\mathbf{G}})$ where $(G, E^{\mathbf{G}})$ is a directed graph and $s^{\mathbf{G}}, t^{\mathbf{G}} \in G$ are two vertices. Let S, T be unary relation symbols and consider the $\{E, S, T\}$-structure $\mathbf{G}' = (G, E^{\mathbf{G}'}, S^{\mathbf{G}'}, T^{\mathbf{G}'})$ with $E^{\mathbf{G}'} := E^{\mathbf{G}}, S^{\mathbf{G}'} := \{s^{\mathbf{G}}\}$ and $T^{\mathbf{G}'} := \{t^{\mathbf{G}}\}$. Define formulas $\varphi_k'(x)$ as $\varphi_k(x)$ but with $\varphi_0'(x) := T(x)$. Then $\exists x(S(x) \wedge \varphi_k'(x))$ is in Σ_1^2 and

$$\mathbf{G} \models \varphi_k(s) \iff \mathbf{G}' \models \exists x(S(x) \wedge \varphi_k'(x)).$$

2.3. **Parameterized complexity.** A *(classical) problem* is a subset $Q \subseteq \{0,1\}^*$, where $\{0,1\}^*$ is the set of finite binary strings; the length of a binary string x is denoted by $|x|$. As model of computation we use Turing machines $\mathbb{A}$ with a (read-only) input tape and several worktapes. We shall consider Turing machines with nondeterminism and co-nondeterminism. For definiteness, let us agree that a *nondeterministic* Turing machine has special states $c_\exists, c_0, c_1$ and can nondeterministically move from state $c_\exists$ to state c_b with $b \in \{0,1\}$, and we say $\mathbb{A}$ *existentially guesses the bit b*. An *alternating* Turing machine additionally has a state $c_\forall$ allowing to *universally guess* a bit b. For a function $c : \{0,1\}^* \to \mathbb{N}$, the machine is said to *use c many (co-)nondeterministic bits* if for every $x \in \{0,1\}^*$ every run[3] of $\mathbb{A}$ on x contains at most $c(x)$ many configurations with state $c_\exists$ (resp. $c_\forall$).

A *parameterized problem* is a pair (Q, κ) of a classical problem Q and a *parameterization κ*, i.e. a logarithmic space computable function $\kappa : \{0,1\}^* \to \mathbb{N}$ mapping any $x \in \{0,1\}^*$ to its *parameter $\kappa(x) \in \mathbb{N}$*.

We exemplify how we present parameterized problems. The *model-checking problem for a class of first-order sentences* Φ is the parameterized problem

p-MC(Φ)	
Instance:	A first-order sentence φ and a structure $\mathbf{A}$.
Parameter:	$|\varphi|$.
Problem:	$\varphi \in \Phi$ and $\mathbf{A} \models \varphi$?

More formally, this is the classical problem MC(Φ) containing all (binary strings encoding) pairs $(\varphi, \mathbf{A})$ with $\varphi \in \Phi$ and $\mathbf{A} \models \varphi$, together with a parameterization that maps binary strings encoding pairs of formulas and structures to the length of the binary string encoding the formula, and all other strings to 0.

The class FPT contains those parameterized problems (Q, κ) that can be decided in *fpt time with respect to κ*, i.e. in time $f(\kappa(x)) \cdot |x|^{O(1)}$ for some computable $f : \mathbb{N} \to \mathbb{N}$. The class para-L (para-NL) contains those parameterized problems (Q, κ) such that Q is decided (accepted) by some (nondeterministic) Turing machine $\mathbb{A}$ that runs in *parameterized logarithmic space with respect to κ*, i.e. in space $f(\kappa(x)) + O(\log |x|)$ for some computable function $f : \mathbb{N} \to \mathbb{N}$. We remark that the class XL is defined using space bound $f(\kappa(x)) \cdot \log |x|$ instead $f(\kappa(x)) + O(\log |x|)$. This class is not known to be contained in FPT. We shall omit the phrase "with respect to κ" if κ is clear from context.

Parameterized logarithmic space reductions have been introduced in [FG03]. We use the following equivalent definition: a *pl-reduction* from (Q, κ) to (Q', κ') is a reduction $R : \{0,1\}^* \to \{0,1\}^*$ from Q to Q' such that $\kappa'(R(x)) \leq f(\kappa(x))$ and $|R(x)| \leq f(\kappa(x)) \cdot |x|^{O(1)}$ for some computable function $f : \mathbb{N} \to \mathbb{N}$, and R is *implicitly pl-computable*, that is, the following problem is in para-L:

p-BITGRAPH(R)	
Instance:	(x, i, b) with $x \in \{0,1\}^*$, $i \geq 1$, and $b \in \{0,1\}$.
Parameter:	$\kappa(x)$.
Problem:	Does $R(x)$ have length $|R(x)| \geq i$ and i-th bit b?

If there is such a reduction, (Q, κ) is *pl-reducible* to (Q', κ'), written $(Q, \kappa) \leq_{pl} (Q', \kappa')$. If also $(Q', \kappa) \leq_{pl} (Q, \kappa)$, then the problems are *pl-equivalent*, written $(Q, \kappa) \equiv_{pl} (Q', \kappa')$.

It is routine to verify that $\leq_{pl}$ is transitive and $\equiv_{pl}$ an equivalence relation.

---

[3]By a *run* of an alternating Turing machine we mean a sequence of configurations such that each, except the first, is a successor configuration of the previous one.

2.4. **The classes PATH and TREE.** The class PATH has been introduced in [EST15] (where it is called paraβL) and can be viewed as a parameterized analogue of NL:

Definition 2.2. A parameterized problem (Q, κ) is in PATH if and only if there exists a nondeterministic Turing machine that accepts Q, runs in parameterized logarithmic space, and uses $f(\kappa(x)) \cdot \log |x|$ many nondeterministic bits for some computable $f : \mathbb{N} \to \mathbb{N}$.

Example 2.3. To gain some intuition for this definition consider the *homomorphism problem for directed paths*:[4] the input is a directed path **P** and a directed graph **G**; the question is whether there is a homomorphism from **P** into **G**; the parameter is $k := |P|$.

Note that $k \cdot \log |G|$ nondeterministic bits are enough to guess a solution, they can however not be stored in parameterized logarithmic space. However, the guess and check algorithm can nevertheless be implemented in such space by observing that, intuitively, a solution can be verified locally. More precisely, the algorithm guesses ($\log |G|$ bits to determine) $b \in G$ and writes (a, b) on some tape where a is the source of **P**. It then repeatedly updates this tape as follows: accept if a is the sink of **P**; else guess $b' \in G$ and check $(b, b') \in E^{\mathbf{B}}$; if this fails, reject; otherwise replace (a, b) by (a', b') where a' is the successor of a in **P**.

One similarly verifies that $p\text{-STCON}_\leq$ is in PATH. In fact, as has been shown in [EST15, Theorem 3.14]:

Theorem 2.4. *The parameterized problem* $p\text{-STCON}_\leq$ *is* PATH*-complete under pl-reductions.*

The class TREE has been introduced in [CM15] and can be viewed as a parameterized analogue of LOGCFL:[5]

Definition 2.5. A parameterized problem (Q, κ) is in TREE if and only if there exists an alternating Turing machine that accepts Q, runs in parameterized logarithmic space, and for some computable $f : \mathbb{N} \to \mathbb{N}$ uses $f(\kappa(x)) \cdot \log |x|$ many nondeterministic bits and $f(\kappa(x))$ many co-nondeterministic bits.

Example 2.6. Again to gain some intuition consider the *homomorphism problem for directed binary trees*: the input is a full binary tree **T** with edges directed away from the root, and a directed graph **G**; the question is whether there is a homomorphism from **T** into **G**; the parameter is $k := |T|$.

To see that this problem belongs to TREE we note that a solution can be locally verified with the help of universal guesses, namely with $h \cdot \log k$ many co-nondeterministic bits where $h := \log k - 1$ is the height of **T**. As in the previous example the algorithm maintains a pair $(a, b) \in T \times G$ on some tape, starting with a being the root of **T**. The tape is updated as follows: if a is a leaf of **T**, accept; otherwise universally guess ($\log k$ bits to determine) $a' \in T$ and check $(a, a') \in E^{\mathbf{T}}$; if this fails, accept; else existentially guess $b' \in G$ and check $(b, b') \in E^{\mathbf{G}}$; if this fails, reject; else replace (a, b) by (a', b').

As outlined in the introduction the classes PATH and TREE play a central role in this article. For this reason we provide more background on these classes in the following by explaining their key role to understanding the parameterized space complexity of the

---

[4]The homomorphism problems for directed paths and binary trees (see Example 2.6) are known to be PATH- and TREE-complete under pl-reductions, respectively [CM15].

[5]The analogy is based on the characterization of LOGCFL as logspace uniform SAC_1 [Ven91].

model-checking problem for structurally restricted primitive positive sentences. We restate the relevant results reformulated in our context. They are not needed later on.

Recall that a sentence φ in a relational vocabulary is *primitive positive* if it is built from atoms by means of conjunctions and existential quantifications. To define the *primal graph* $\mathbf{G}(\varphi)$ first write φ in prenex form (introducing new variables) and then delete each subformula of the form $x=y$ replacing all occurrences of y by x; the graph $\mathbf{G}(\varphi)$ has the variables of the resulting sentence as vertices, and an edge between two distinct variables if they occur both in some atomic subformula of φ. A class Φ of primitive positive sentences has *bounded arity* if there is a constant $r \in \mathbb{N}$ such that all relation symbols in all sentences in Φ have arity at most r. Let us say, Φ has *bounded tree-depth up to logical equivalence* if there is a constant $c \in \mathbb{N}$ such that every sentence in Φ is logically equivalent to a primitive positive sentence whose primal graph has tree-depth at most c. We use a similar mode of speech for path-width and tree-width.

Then the complexity classification of problems $p\text{-}\mathrm{MC}(\Phi)$ reads as follows.

Theorem 2.7. *Let Φ be a decidable class of primitive positive sentences of bounded arity.*
(1) *If Φ has bounded tree-depth up to logical equivalence, then $p\text{-}\mathrm{MC}(\Phi) \in$ para-L.*
(2) *If Φ does not have bounded tree-depth but bounded path-width up to logical equivalence, then $p\text{-}\mathrm{MC}(\Phi)$ is PATH-complete under pl-reductions.*
(3) *If Φ does not have bounded path-width but bounded tree-width up to logical equivalence, then $p\text{-}\mathrm{MC}(\Phi)$ is TREE-complete under pl-reductions.*
(4) *If Φ does not have bounded tree-width up to logical equivalence, then $p\text{-}\mathrm{MC}(\Phi)$ is not in FPT unless $\mathrm{A}[1] = \mathrm{FPT}$.*

Statement (4) is Grohe's famous classification [Gro07] (building on [GSS01]). The other statements are from [CM15]. An even finer classification (including (4)) appears in [CM17]. Note statements (3) and (4) state a complexity gap (assuming $\mathrm{A}[1] \neq \mathrm{FPT}$): if $p\text{-}\mathrm{MC}(\Phi)$ is in FPT at all, then it is already in TREE.

Originally these results have been phrased for homomorphism problems. The equivalence of the problems is well known since Chandra and Merlin's seminal paper [CM77]. Instead of logical equivalence of sentences one talks about homomorphic equivalence of relational structures, that is, structures with isomorphic cores.

Finally, the connection to bounded variable logics mentioned in the introduction follows from [KV00, Lemma 5.2, Remark 5.3] (see also [DKV02, Theorem 12] and [Che14, Theorem 4] for similar statements):

Theorem 2.8. *Let $s \in \mathbb{N}$ and φ be a primitive positive sentence. Then φ is logically equivalent to a primitive positive sentence whose primal graph has treewidth less than s if and only if φ is logically equivalent to a primitive positive sentence in FO^s.*

3. The tree hierarchy

Theorem 2.7 and the following discussion give some special interest to the class TREE. It has been asked in [CM15] whether TREE is closed under complementation. If not, then we get an alternation hierarchy above TREE defined in the usual way (that still might collapse to a level further up). In this section we define this *tree hierarchy* and make some initial observations.

3.1. **Definitions.** Following [CM15] we consider machines $\mathbb{A}$ with *mixed nondeterminism*. Additionally to the binary nondeterminism embodied in the states $c_\exists, c_\forall, c_0, c_1$ from Section 2.3 they use *jumps* explained as follows. Recall that our Turing machines have an input tape. During a computation on an input x of length $n := |x| > 0$ the cells numbered 1 to n of the input tape contain the n bits of x. The machine has an *existential* and a *universal jump state* $j_\exists$ resp. $j_\forall$. A successor configuration of a configuration in a jump state is obtained by changing the state to the initial state and placing the input head on an arbitrary cell holding an input bit; the machine is said to *existentially resp. universally jump to* the cell.

Acceptance is defined as usual for alternating machines. Call a configuration *universal* if it has state $j_\forall$ or $c_\forall$, and otherwise *existential*. The machine $\mathbb{A}$ accepts $x \in \{0, 1\}^*$ if its initial configuration on x is *accepting*. The set of accepting configurations is the smallest set that contains all accepting halting configurations, that contains an existential configuration if it contains at least one of its successor configurations, and that contains a universal configuration if it contains all of its successor configurations.

Observe that the number of the cell to which the machine jumps can be computed in logarithmic space by moving the input head stepwise to the left. Intuitively, a jump should be thought as a guess of a number in $[n]$.

Each run of $\mathbb{A}$ on some input x contains a subsequence of jump configurations (i. e. with state $j_\exists$ or $j_\forall$). For a natural number $t \geq 1$ the run is *t-alternating* if this subsequence consists of t blocks, the first consisting of existential configurations, the second in universal configurations, and so on. The machine $\mathbb{A}$ is *t-alternating* if every run of $\mathbb{A}$ on any input is t-alternating.

Note that a 1-alternating machine can existentially jump but not universally; it may however use universal (and existential) bits; in fact, the use of nondeterministic bits is completely neglected by the above definition.

For $f : \{0, 1\}^* \to \mathbb{N}$, we say $\mathbb{A}$ *uses f jumps (bits)* if for every $x \in \{0, 1\}^*$ every run of $\mathbb{A}$ on x contains at most $f(x)$ many jump configurations (resp. configurations with state $c_\exists$ or $c_\forall$). As for a more general notation, note that every run of $\mathbb{A}$ on x contains a (possibly empty) sequence of *nondeterministic configurations*, i. e. with state in $\{j_\exists, j_\forall, c_\exists, c_\forall\}$. The *nondeterminism type* of the run is the corresponding word over the alphabet $\{j_\exists, j_\forall, c_\exists, c_\forall\}$. For example, being $2t$-alternating means having nondeterminism type in $(\{j_\exists, c_\exists, c_\forall\}^* \{j_\forall, c_\exists, c_\forall\}^*)^t$. Here and below, we use regular expressions to denote languages over $\{j_\exists, j_\forall, c_\exists, c_\forall\}$.

Definition 3.1. A parameterized problem (Q, κ) is in TREE[∗] if there are a computable $f : \mathbb{N} \to \mathbb{N}$ and a machine $\mathbb{A}$ with mixed nondeterminism that accepts Q, runs in parameterized logarithmic space (with respect to κ) and uses $f \circ \kappa$ jumps and $f \circ \kappa$ bits. If additionally $\mathbb{A}$ is t-alternating for some $t \geq 1$, then (Q, κ) is in TREE[t].

The definition of TREE[t] is due to Hubie Chen (personal communication).

3.2. **Observations.** The following two propositions are straightforward (cf. [CM15, Lemmas 4.5, 5.4]):

Proposition 3.2. *A parameterized problem (Q, κ) is in PATH if and only if there are a computable $f : \mathbb{N} \to \mathbb{N}$ and a 1-alternating machine with mixed nondeterminism that accepts Q, runs in parameterized logarithmic space (with respect to κ) and uses $f \circ \kappa$ jumps and 0 bits.*

Proposition 3.3. TREE = TREE[1].

Hence, all inclusions displayed in (1.2) in the introduction are trivial except possibly the last one TREE[∗] ⊆ FPT. We prove it in Corollary 4.4. It is likely to be strict:

Proposition 3.4.
(1) para-NL ⊆ TREE[∗] *if and only if* NL ⊆ L.
(2) FPT ⊆ TREE[∗] *if and only if* P ⊆ L.

Proof. We prove (1), the proof of (2) is similar. For the backward direction, note that NL ⊆ L implies para-NL ⊆ para-L by general results of Flum and Grohe [FG03, Theorem 4, Proposition 8]. The same results also imply the forward direction noting that TREE[∗] is contained in XL. We include a direct argument: assume para-NL ⊆ TREE[∗], let Q be a classical problem which is NL-complete under logarithmic space reductions and let κ_0 be the parameterization which is constantly 0. Then $(Q, \kappa_0) \in$ para-NL ⊆ TREE[∗], so there are a function f and machine $\mathbb{A}$ with mixed nondeterminism that accepts Q and on input x uses $f(0) + O(\log|x|)$ space and $f(0)$ jumps and bits. Thus, Q can be decided in logarithmic space by simulating $\mathbb{A}$ for all possible outcomes of these constantly many jumps and bits. $\square$

Remark 3.5. [EST15] observed that para-NL ⊆ PATH is equivalent to NL ⊆ L.

The following technical lemma will prove useful in the next section.

Lemma 3.6 (Normalization). *Let $t \geq 1$ and (Q, κ) be a parameterized problem.*
(1) $(Q, \kappa) \in \textsc{Tree}[t]$ *if and only if there are a computable $f : \mathbb{N} \to \mathbb{N}$ and a t-alternating machine $\mathbb{A}$ with mixed nondeterminism that accepts Q, runs in parameterized logarithmic space and such that for all $x \in \{0,1\}^*$ every run of $\mathbb{A}$ on x has nondeterminism type:*

$$\left(\ (j_\exists c_\forall)^{f(\kappa(x))} (j_\forall c_\exists)^{f(\kappa(x))}\ \right)^{\lfloor t/2 \rfloor} (j_\exists c_\forall)^{f(\kappa(x)) \cdot (t \bmod 2)}. \tag{3.1}$$

(2) $(Q, \kappa) \in \textsc{Tree}[*]$ *if and only if there are a computable $f : \mathbb{N} \to \mathbb{N}$ and machine $\mathbb{A}$ with mixed nondeterminism that accepts Q, runs in parameterized logarithmic space and such that for all $x \in \{0,1\}^*$ every run of $\mathbb{A}$ on x has nondeterminism type:*

$$(j_\exists j_\forall)^{f(\kappa(x))}. \tag{3.2}$$

Proof. We only show (1), the proof of (2) is similar. The backward direction of (1) is obvious. To prove the forward direction, assume $(Q, \kappa) \in \textsc{Tree}[t]$ and choose a parameterized logarithmic space t-alternating machine $\mathbb{A}$ with mixed nondeterminism accepting Q and a computable function $f : \mathbb{N} \to \mathbb{N}$ such that $\mathbb{A}$ on input $x \in \{0,1\}^*$ uses $f(\kappa(x))$ many jumps and bits.

Every run of $\mathbb{A}$ on x consists of at most t *blocks*. The first block is the sequence of configurations from the starting configuration until the first configuration in state $j_\forall$, the second block is the sequence starting from this configuration until the first following configuration in state $j_\exists$, and so on. We can assume that the first nondeterministic configuration is a jump configuration and thus has state $j_\exists$.

Define the machine $\mathbb{A}'$ as follows. On an input x of length at least 2, $\mathbb{A}'$ simulates $\mathbb{A}$ on x and additionally stores the parity of (the number of) the current block of the run. In an odd (even) block $\mathbb{A}'$ replaces $\mathbb{A}$'s existential (universal) guesses of a bit b by existential (universal) jumps. Namely, it first computes the number m of the cell currently read on the input tape, then performs an existential (universal) jump, then computes the parity b of the cell it jumps to, then moves the input head back to cell m and then continues the run of $\mathbb{A}$ with guessed bit b (i.e. in state c_b).

Then $\mathbb{A}'$ accepts Q and every run of $\mathbb{A}'$ on x does not have any configuration with state $c_\exists$ ($c_\forall$) in an odd (even) block. More precisely, and assuming $t = 3$ for notational simplicity: every run of $\mathbb{A}'$ on x has a nondeterminism type which is a prefix of a word in

$$j_\exists\{c_\forall, j_\exists\}^{\leq f(\kappa(x))} j_\forall\{c_\exists, j_\forall\}^{\leq f(\kappa(x))} j_\exists\{c_\forall, j_\exists\}^{\leq f(\kappa(x))}. \tag{3.3}$$

Since κ is computable in logarithmic space, the number $k := f(\kappa(x))$ can be computed in parameterized logarithmic space. Define a machine $\mathbb{A}''$ which on x first computes $k := f(\kappa(x))$ and then simulates $\mathbb{A}'$ keeping record of the nondeterminism type of the sofar simulated run. Moreover, $\mathbb{A}''$ uses the record to do appropriate dummy nondeterministic steps to ensure its nondeterminism type to be

$$j_\exists(c_\forall j_\exists)^{2k} c_\forall j_\forall (c_\exists j_\forall)^{2k} c_\exists j_\exists (c_\forall j_\exists)^{2k} c_\forall. \qquad \square$$

4. Model-checking problems and the tree hierarchy

In Section 4.1 we prove Theorem 1.1 (1) as Theorem 4.1, and draw some corollaries to its proof in Section 4.2. In particular, Theorem 1.1 (2) is proved as Corollary 4.3, and the collapse theorem announced in the introduction is proved as Corollary 4.2. In Section 4.3 we prove that the completeness results stay true when function symbols are allowed.

4.1. **Completeness results.** It is easy to see that $p\text{-}\mathrm{MC}(\mathrm{FO}^s) \in \mathrm{para\text{-}L}$ for $s = 1$. We prove completeness results for $s \geq 2$.

Theorem 4.1. Let $t \geq 1$ and $s \geq 2$. Then $p\text{-}\mathrm{MC}(\Sigma_t^s)$ is complete for $\mathrm{TREE}[t]$ under pl-reductions.

Proof. (Containment) We first show that $p\text{-}\mathrm{MC}(\Sigma_t^s) \in \mathrm{TREE}[t]$. This is done by a straightforward algorithm $\mathbb{A}$ as follows. At any moment it keeps in memory a formula ψ and an assignment α to its free variables. For simplicity we assume that all negation symbols of ψ appear only in front of atoms.

In case ψ is an atom or a negated atom, $\mathbb{A}$ accepts if α satisfies ψ in $\mathbf{A}$ and rejects otherwise. If ψ is a disjunction $(\psi_0 \vee \psi_1)$ $\big($conjunction $(\psi_0 \wedge \psi_1)\big)$, the algorithm $\mathbb{A}$ existentially (universally) guesses a bit b and recurses replacing ψ by ψ_b and α by its restriction to the free variables in ψ_b. If ψ is $\exists x \psi_0$ (resp. $\forall x \psi_0$), then $\mathbb{A}$ makes an existential (resp. universal) jump to guess $a \in A$ and recurses replacing ψ by ψ_0 and α by the assignment which extends α mapping x to a. Here we assume x occurs freely in ψ_0 — otherwise $\mathbb{A}$ simply recurses on ψ_0 with α unchanged.

When started on a Σ_t^s-sentence φ and the empty assignment, the formulas occurring during the recursion are subformulas of φ and thus contain as most s variables, so each of the assignments computed during the recursion can be stored in space roughly $s \cdot \log |A|$.

(Hardness) We now show that $p\text{-}\mathrm{MC}(\Sigma_t^2)$ is hard for $\mathrm{TREE}[t]$ under pl-reductions. Given a problem decided by a t-alternating machine $\mathbb{B}$ in small space, we construct a structure akin to the configuration graph of $\mathbb{B}$. Its universe consists of all small *nondeterministic* configurations of $\mathbb{B}$. We draw a directed edge from one such configuration to another if there exists a deterministic computation of $\mathbb{B}$ leading from the first configuration to the second. We then express acceptance by a short (parameter bounded) sentence which is constructed in two steps: first we give a direct and intuitive construction using function symbols. Then, in a second step, we show how to eliminate these functions symbols. Details follow.

Let $(Q, \kappa) \in \text{Tree}[t]$ be given and choose a computable f and a t-alternating machine $\mathbb{B}$ with $f \circ \kappa$ jumps and $f \circ \kappa$ bits such that $\mathbb{B}$ accepts Q and runs in space $f(\kappa(x)) + O(\log |x|)$.

Given $x \in \{0, 1\}^*$ compute an upper bound $s = f(\kappa(x)) + O(\log |x|)$ on the space needed by $\mathbb{B}$ on x. Since κ is computable in logarithmic space, such a number s can be computed in parameterized logarithmic space. We can assume that $\mathbb{B}$ on x always halts after at most $m = 2^{f(\kappa(x))} \cdot |x|^{O(1)}$ steps. Note the binary representation of m can be computed in parameterized logarithmic space.

For two space s configurations c, c' of $\mathbb{B}$ on x, we say that $\mathbb{B}$ *reaches c' from c* if there is a computation of $\mathbb{B}$ leading from c to c' of length at most m that does neither pass through a nondeterministic configuration nor through a configuration of space larger than s. In particular, c cannot be nondeterministic but c' can. We assume that $\mathbb{B}$ reaches a nondeterministic configuration from the initial configuration, i. e. the computation of $\mathbb{B}$ on x is not deterministic.

We define a structure $\mathbf{A}$ whose universe A comprises all $\left(\text{length } O(s) \text{ binary codes of}\right)$ nondeterministic space s configurations of $\mathbb{B}$ on x. It interprets a binary relation symbol E, unary function symbols s_0, s_1 and unary relation symbols $S, F, J_\exists, J_\forall, C_\exists, C_\forall$ as follows.

A pair $(c, c') \in A^2$ is in $E^{\mathbf{A}}$ if there exists a successor configuration c'' of c such that $\mathbb{B}$ reaches c' from c''. The relation symbol S is interpreted by $S^{\mathbf{A}} = \{c_{\text{first}}\}$ where c_{first} is the (unique) first configuration in A reached by $\mathbb{B}$ from the initial configuration of $\mathbb{B}$ on x. The relation symbols $J_\exists, J_\forall, C_\exists$ and $C_\forall$ are interpreted by the sets of configurations in A with states $j_\exists, j_\forall, c_\exists$ and $c_\forall$ respectively. Obviously these sets partition A.

The relation symbol F is interpreted by the set $F^{\mathbf{A}}$ of those $c \in A$ such that one of the following holds:

- $c \in C_\exists^{\mathbf{A}} \cup J_\exists^{\mathbf{A}}$ and $\mathbb{B}$ reaches a space s accepting halting configuration from at least one successor configuration of c;
- $c \in C_\forall^{\mathbf{A}} \cup J_\forall^{\mathbf{A}}$ and $\mathbb{B}$ reaches a space s accepting halting configuration from all successor configurations of c.

Finally, the function symbols s_0 and s_1 are interpreted by any functions $s_0^{\mathbf{A}}, s_1^{\mathbf{A}} : A \to A$ such that for every $c \in C_\exists^{\mathbf{A}} \cup C_\forall^{\mathbf{A}}$ with $\{d \mid (c, d) \in E^{\mathbf{A}}\} \neq \emptyset$ we have:

$$\left\{s_0^{\mathbf{A}}(c), s_1^{\mathbf{A}}(c)\right\} = \left\{d \in A \mid (c, d) \in E^{\mathbf{A}}\right\}.$$

It is easy to check that $\mathbf{A}$ is computable from x in parameterized logarithmic space. For example, to check whether a given pair $(c, c') \in A^2$ is in $E^{\mathbf{A}}$ we simulate $\mathbb{B}$ starting from c for at most m steps; if the simulation wants to visit a configuration of space larger than s or a nondeterministic configuration $\neq c'$, then we stop the simulation and reject.

For a word w of length $|w| \geq 1$ over the alphabet $\{j_\exists, j_\forall, c_\exists, c_\forall\}$ we define a formula $\varphi_w(x)$ with (free or bound) variables x, y as follows. We proceed by induction on the length $|w|$.

If $|w| = 1$, define $\varphi_w(x) := F(x)$. For $|w| \geq 1$ define:

$$\varphi_{c_\forall w}(x) := C_\forall(x) \wedge \left(\varphi_w(s_0(x)) \wedge \varphi_w(s_1(x))\right),$$

$$\varphi_{c_\exists w}(x) := C_\exists(x) \wedge \left(\varphi_w(s_0(x)) \vee \varphi_w(s_1(x))\right),$$

$$\varphi_{j_\exists w}(x) := J_\exists(x) \wedge \exists y\left(E(x, y) \wedge \exists x(x = y \wedge \varphi_w(x))\right),$$

$$\varphi_{j_\forall w}(x) := J_\forall(x) \wedge \forall y\left(\neg E(x, y) \vee \forall x(\neg x = y \vee \varphi_w(x))\right).$$

Let $|w| \geq 1$ and assume that $c \in A$ is a configuration such that every run of $\mathbb{B}$ on x starting at c has nondeterminism type w and consists of space s configurations; then

$$c \text{ is accepting} \iff \mathbf{A} \models \varphi_w(c). \tag{4.1}$$

This follows by a straightforward induction on $|w|$. Now we look for $\mathbf{A}'$ and $\varphi_w' \in \Sigma_t^2$ with this property but in a relational vocabulary.

By the Normalization Lemma 3.6 we can assume that all runs of $\mathbb{B}$ on x have nondeterminism type w of the form (3.1). For such a w we observe that $\varphi_w(x)$ has the required quantifier structure: in the notation of the next Section 4.3 it is in func-Σ_t^2, i.e. the class of formulas defined as Σ_t^2 but allowing function symbols.

Furthermore, $\varphi_w(x)$ does not contain nested terms, in fact, all its atomic subformulas containing some function symbol are of the form $E(s_b(x), y)$, $J_\exists(s_b(x))$, or $J_\forall(s_b(x))$. For $b \in \{0,1\}$ we introduce binary relation symbols E_b and unary relation symbols $J_{\forall b}$ and $J_{\exists b}$, and then replace the atomic subformulas $E(s_b(x), y)$, $J_\exists(s_b(x))$, $J_\forall(s_b(x))$ in $\varphi_w(x)$ by $E_b(x, y)$, $J_{\exists b}(x)$, $J_{\forall b}(x)$ respectively. This defines the formula $\varphi_w'(x)$. Since $\varphi_w(x) \in$ func-Σ_t^2, we have $\varphi_w'(x) \in \Sigma_t^2$.

To define $\mathbf{A}'$ we expand $\mathbf{A}$ by interpreting the new symbols by:

$$E_b^{\mathbf{A}'} := \left\{ (c, d) \mid (s_b^{\mathbf{A}}(c), d) \in E^{\mathbf{A}} \right\},$$
$$J_{\exists b}^{\mathbf{A}'} := \left\{ c \mid s_b^{\mathbf{A}}(c) \in J_\exists^{\mathbf{A}} \right\},$$
$$J_{\forall b}^{\mathbf{A}'} := \left\{ c \mid s_b^{\mathbf{A}}(c) \in J_\forall^{\mathbf{A}} \right\}.$$

We have for all $c \in A$:

$$\mathbf{A} \models \varphi_w(c) \iff \mathbf{A}' \models \varphi_w'(c).$$

As the assumption of (4.1) is satisfied for c_{first}, and c_{first} is accepting if and only if $\mathbb{B}$ accepts x, that is, if and only if $x \in Q$, we get

$$x \in Q \iff \mathbf{A}' \models \varphi_w'(c_{\text{first}}).$$

Setting $\psi := \exists x (S(x) \wedge \varphi_w'(x))$ we get a reduction as desired by mapping x to $(\psi, \mathbf{A}')$. $\square$

4.2. **Corollaries.** The easy elimination of function symbols in the proof of Theorem 4.1 rests on the fact that it is applied only to formulas where function symbols are not nested. We shall treat the general case in the next subsection. Before that we draw some consequences of the last theorem and its proof.

Corollary 4.2. *Let $t' > t \geq 1$. If* TREE$[t]$ *is closed under complementation, then*

$$\text{TREE}[t'] = \text{TREE}[t].$$

Proof. It is sufficient to show this for $t' = t + 1$. Assume TREE$[t]$ is closed under complementation. By the previous theorem it suffices to show p-MC$(\Sigma_{t+1}^2) \in$ TREE$[t]$, and we know p-MC$(\Sigma_t^2) \in$ TREE$[t]$. By assumption we find a computable f and a t-alternating machine $\mathbb{A}'$ with $f \circ \kappa$ jumps and $f \circ \kappa$ bits that runs in parameterized logarithmic space and accepts the complement of p-MC(Σ_t^2).

We describe a machine $\mathbb{B}$ accepting p-MC(Σ_{t+1}^2). On input $(\varphi, \mathbf{A})$ with a sentence $\varphi \in \Sigma_t^2$ we first eliminate all void universal quantifiers, i.e. replace all subformulas $\forall x \psi$ by ψ whenever x does not appear free in ψ. The machine $\mathbb{B}$ starts simulating the recursive algorithm $\mathbb{A}$ for p-MC(Σ_{t+1}^2) described in the previous proof. The simulation is stopped

when $\mathbb{A}$ recurses to a subformula of φ of the form $\forall x \psi$. In this case, $\forall x \psi$ has at most one free variable y and the current assignment α maps it, say, to $a \in A$. The machine $\mathbb{B}$ simulates $\mathbb{A}'$ on $(\chi, \mathbf{A}')$ where $\mathbf{A}'$ expands $\mathbf{A}$ interpreting a new unary relation symbol C by $\{a\}$ and where χ is a Σ_t^2-sentence logically equivalent to $\neg \forall xy(C(y) \to \psi)$.

It is clear that $\mathbb{B}$ accepts $p\text{-MC}(\Sigma_{t+1}^2)$. Observe that $\mathbb{B}$ does not make universal jumps before it starts simulating $\mathbb{A}'$. Hence, since $\mathbb{A}'$ is t-alternating, so is $\mathbb{B}$. The number of jumps and guesses in a run of $\mathbb{B}$ before the simulation of $\mathbb{A}'$ is clearly bounded by $|\varphi|$, the parameter. Furthermore, $\mathbb{B}$ can be implemented in parameterized logarithmic space: for the simulation of $\mathbb{A}'$ it stores (ψ, a) and relies on the implicit pl-computability of $(\chi, \mathbf{A}')$ from $(\psi, a, \mathbf{A})$. Thus, $\mathbb{B}$ witnesses that $p\text{-MC}(\Sigma_{t+1}^2) \in \text{TREE}[t]$. $\qquad\square$

Corollary 4.3. *Let $s \geq 2$. Then $p\text{-MC}(\text{FO}^s)$ is complete for* TREE[∗] *under pl-reductions.*

Proof. That $p\text{-MC}(\text{FO}^s) \in \text{TREE}[*]$ can be seen as in the proof of Theorem 4.1. Hardness of $p\text{-MC}(\text{FO}^2)$ also follows as in this proof, but the argument is actually simpler: let $(Q, \kappa) \in \text{TREE}[*]$ and choose a machine accepting it according to the Normalization Lemma 3.6 (2). Observe that the formula φ_w does not contain the function symbols s_0, s_1 for w as in (3.2). Hence the reduction can simply map $x \in \{0, 1\}^*$ to $(\exists x(S(x) \wedge \varphi_w(x)), \mathbf{A}'')$ where $w = (j_\exists j_\forall)^{f(\kappa(x))}$ and $\mathbf{A}''$ is obtained from $\mathbf{A}$ by forgetting the interpretations of s_0, s_1. It follows from (4.1) that this defines a pl-reduction from (Q, κ) to $p\text{-MC}(\text{FO}^2)$. $\qquad\square$

Corollary 4.4. TREE[∗] ⊆ FPT.

Proof. By Corollary 4.3 and the fact that $\text{MC}(\text{FO}^2)$ is in P. $\qquad\square$

The introduction mentioned the result that the classical problem $\text{MC}(\text{FO}^s)$ is P-complete for $s \geq 2$. As a further corollary we get that the parameterized analogue of this completeness result is likely false:

Corollary 4.5. *Unless* P = L, *there is no $s \in \mathbb{N}$ such that $p\text{-MC}(\text{FO}^s)$ is FPT-complete under pl-reductions.*

Proof. If $p\text{-MC}(\text{FO}^s)$ is FPT-complete, then (we can assume $s \geq 2$ and thus) TREE[∗] = FPT by Corollary 4.3. This implies P = L by Proposition 3.4. $\qquad\square$

4.3. Function symbols.
Let func-Σ_t be defined as Σ_t except that function symbols are allowed. It is not hard to show (see e.g. [FG06, Example 8.55]) that $p\text{-MC}(\text{func-}\Sigma_t)$ is equivalent to $p\text{-MC}(\Sigma_t)$ under fpt-reductions (even pl-reductions). An analogous statement for the model-checking problems characterizing the classes of the W-hierarchy is not known. In fact, allowing function symbols gives problems complete for the presumably larger classes of the W$^{\text{func}}$-hierarchy. We refer to [CFG05] for more information.

Let func-Σ_t^s and func-FO^s be defined as Σ_t^s and FO^s, respectively, except that function symbols are allowed.

Theorem 4.6. *Let $s \geq 2$ and $t \geq 1$. Then*
(1) $p\text{-MC}(\Sigma_t^s) \equiv_{pl} p\text{-MC}(\text{func-}\Sigma_t^s)$.
(2) $p\text{-MC}(\text{FO}^s) \equiv_{pl} p\text{-MC}(\text{func-FO}^s)$.

Proof. The second statement will be an easy corollary to the proof of the first. We first give a rough, informal sketch of this proof. The idea is, as usual, to replace functions by their graphs and translate formulas to the resulting relational vocabulary using formulas $value_t(x, \bar{x})$ expressing that x is the value of the term t at $\bar{x}$. This formula implements the bottom-up evaluation of t at $\bar{x}$ in a straightforward way. However, to do so, the formula needs to quantify over tuples of intermediate values. To do this using a single variable we extend the structure to contain tuples of appropriate lengths. This enables us to write $value_t(x, \bar{x})$ and the whole translation with only a constant overhead of new variables. To preserve the quantifier alternation rank we write two versions of $value_t(x, \bar{x})$, an existential and a universal one. Details follow.

To prove statement (1) it suffices, by Theorem 4.1, to show

$$p\text{-MC}(\text{func-}\Sigma_t^s) \leq_{pl} p\text{-MC}(\Sigma_t^{s+3}).$$

Let $\mathbf{A}$ be a structure of a vocabulary τ with $|A| \geq 2$. We define a relational vocabulary τ' which depends on τ only, and a τ'-structure $\mathbf{A}'$. The universe A' is the union
- of A;
- for each relation symbol $R \in \tau$ of arity r, of the set

$$\big\{(a_1, \ldots, a_i) \mid 2 \leq i \leq r \text{ and } (a_1, \ldots, a_i, a_{i+1}, \ldots, a_r) \in R^{\mathbf{A}} \text{ for some } a_{i+1}, \ldots, a_r \in A\big\},$$

  i.e. the set of all 'partial' tuples that can be extended to some tuple in $R^{\mathbf{A}}$; note that the size of this set is bounded by $(r-1) \cdot |R^{\mathbf{A}}|$;
- for each function symbol $f \in \tau$ of arity r, of the set

$$\textstyle\bigcup_{2 \leq i \leq r} A^i.$$

We identify A^1 with A and it is therefore that the union $\bigcup_{2 \leq i \leq r} A^i$ starts at $i = 2$. Recall we assumed $|A| \geq 2$, so $\bigcup_{2 \leq i \leq r} A^i$ has size at most $|A|^{r+1}$. Altogether, $|A'| \leq |\mathbf{A}|^2$.

Now we define the vocabulary τ' and the τ'-structure $\mathbf{A}'$ in parallel:
- τ' contains a unary relation symbol U interpreted by the original universe of $\mathbf{A}$, that is,

$$U^{\mathbf{A}'} := A.$$

- For every constant symbol $c \in \tau$, the vocabulary τ' contains a unary relation symbol U_c which is interpreted as

$$U_c^{\mathbf{A}'} := \big\{c^{\mathbf{A}}\big\}.$$

- For every r-ary relation symbol $R \in \tau$, the vocabulary τ' contains a unary relation symbol U_R and we set

$$U_R^{\mathbf{A}'} := \big\{b \in A' \mid b = (a_1, \ldots, a_r) \text{ for some } (a_1, \ldots, a_r) \in R^{\mathbf{A}}\big\}.$$

- For every r-ary function symbol $f \in \tau$, the vocabulary τ' contains a binary relation symbol F_f and we define

$$F_f^{\mathbf{A}'} := \big\{((a_1, \ldots, a_r), a) \in A' \mid (a_1, \ldots, a_r) \in A^r \text{ and } f^{\mathbf{A}}(a_1, \ldots, a_r) = a\big\}.$$

- τ' contains a ternary relation symbol R_e; the 'tuple *extending* relation' $R_e^{\mathbf{A}'}$ is defined by

$$R_e^{\mathbf{A}'} := \big\{(b, a, b') \in A' \times A \times A' \mid \text{there are } i \geq 1 \text{ and } a_1, \ldots, a_i \in A \text{ such that}$$
$$b = (a_1, \ldots, a_i) \in A' \text{ and } b' = (a_1, \ldots, a_i, a) \in A'\big\},$$

Now, for any sentence $\varphi \in$ func-Σ_t^s, we construct two sentences $trans_\varphi^\exists$ and $trans_\varphi^\forall$ such that

$$\mathbf{A} \models \varphi \quad \Longleftrightarrow \quad \mathbf{A}' \models trans_\varphi^\exists$$
$$\Longleftrightarrow \quad \mathbf{A}' \models trans_\varphi^\forall. \tag{4.2}$$

We start by defining, for every term $m(\bar{x})$ with $\bar{x} = x_1, \ldots, x_s$, two formulas

$$value_m^\exists(x, \bar{x}) \quad \text{and} \quad value_m^\forall(x, \bar{x})$$

which respectively are in Σ_1^{s+3} and Π_1^{s+3} up to logical equivalence. Furthermore, for every $a \in A'$ and every $a_1, \ldots, a_s \in A$ it holds that

$$a = m^{\mathbf{A}}(a_1, \ldots, a_s) \iff \mathbf{A}' \models value_m^\exists(a, a_1, \ldots, a_s)$$
$$\iff \mathbf{A}' \models value_m^\forall(a, a_1, \ldots, a_s).$$

If m is a variable x_i, then $value_m^\exists := value_m^\forall := x = x_i$. If m is a constant c, then

$$value_m^\exists := value_m^\forall := U_c(x).$$

If m is the composed term $f(m_1, \ldots, m_r)$, then

$$value_m^\exists(x, \bar{x}) := \exists y \Big(\exists x \big(x = y \wedge tuple_{m_1, \ldots, m_r}(x, \bar{x}) \big) \wedge F_f(y, x) \Big),$$

$$value_m^\forall(x, \bar{x}) := \forall y \Big(\neg \exists x \big(x = y \wedge tuple_{m_1, \ldots, m_r}(x, \bar{x}) \big) \vee F_f(y, x) \Big),$$

where $tuple_{m_1, \ldots, m_r}$ is defined inductively on r as follows.

$$tuple_{m_1}(x, \bar{x}) := value_{m_1}^\exists(x, \bar{x}),$$

$$tuple_{m_1, \ldots, m_{i+1}}(x, \bar{x}) := \exists y \exists z \Big(R_e(y, z, x) \wedge \exists x \big(x = y \wedge tuple_{m_1, \ldots, m_i}(x, \bar{x}) \big)$$
$$\wedge \exists x \big(x = z \wedge value_{m_i}^\exists(x, \bar{x}) \big) \Big).$$

It is easy to verify that $tuple_{m_1, \ldots, m_r} \in \Sigma_1^{s+3}$ and for every $b \in A'$ and every $a_1, \ldots, a_s \in A$

$$b = \Big(m_1^{\mathbf{A}}(a_1, \ldots, a_s), \ldots, m_r^{\mathbf{A}}(a_1, \ldots, a_s) \Big) \quad \Longleftrightarrow \quad \mathbf{A}' \models tuple_{m_1, \ldots, m_r}(b, a_1, \ldots, a_s).$$

For formulas φ we define $trans_\varphi^\exists$ and $trans_\varphi^\forall$ by induction as follows:

$$trans_{m_1 = m_2}^\exists := \exists x \big(value_{m_1}^\exists(x, \bar{x}) \wedge value_{m_2}^\exists(x, \bar{x}) \big),$$
$$trans_{m_1 = m_2}^\forall := \forall x \big(\neg value_{m_1}^\exists(x, \bar{x}) \vee value_{m_2}^\forall(x, \bar{x}) \big),$$
$$trans_{Rm_1 \ldots m_r}^\exists := \exists x \big(U_R(x) \wedge tuple_{m_1, \ldots, m_r}(x, \bar{x}) \big),$$
$$trans_{Rm_1 \ldots m_r}^\forall := \forall x \big(\neg tuple_{m_1, \ldots, m_r}^\exists(x, \bar{x}) \vee U_R(x) \big).$$

If $\varphi = \neg \psi$, then $trans_\varphi^\exists := \neg trans_\psi^\forall$ and $trans_\varphi^\forall := \neg trans_\psi^\exists$. If $\varphi = (\psi_1 \vee \psi_2)$, then $trans_\varphi^\exists := \big(trans_{\psi_1}^\exists \vee trans_{\psi_2}^\exists \big)$ and $trans_\varphi^\forall := \big(trans_{\psi_1}^\forall \vee trans_{\psi_2}^\forall \big)$. The case for $(\psi_1 \wedge \psi_2)$ is similar. If $\varphi = \exists x_i \psi$, we define

$$trans_\varphi^\exists := \exists x_i \big(U_A(x_i) \wedge trans_\psi^\exists \big) \quad \text{and} \quad trans^\forall(\varphi) := \exists x_i \big(U_A(x_i) \wedge trans_\psi^\forall \big).$$

Similarly, for $\varphi = \forall x \psi$,

$$trans_\varphi^\exists := \forall x_i \big(\neg U_A(x_i) \vee trans_\psi^\exists \big) \quad \text{and} \quad trans_\varphi^\forall := \forall x_i \big(\neg U_A(x_i) \vee trans_\psi^\forall \big).$$

It is routine to verify (4.2). Moreover, if t is odd, then $trans_\varphi^\exists$ is equivalent to a Σ_t^{s+3}-sentence; and if t is even, then $trans_\varphi^\forall$ is equivalent to a Σ_t^{s+3}-sentence. For simplicity, we denote the corresponding Σ_t^{s+3}-sentence and Σ_t^{s+3}-sentence by $trans_\varphi^\exists$ and $trans_\varphi^\forall$ again. Therefore, for every structure $\mathbf{A}$ and $\varphi \in \text{func-}\Sigma_t^s$

$$(\mathbf{A}, \varphi) \mapsto \begin{cases} (\mathbf{A}', trans_\varphi^\exists) & \text{if } t \text{ is odd,} \\ (\mathbf{A}', trans_\varphi^\forall) & \text{if } t \text{ is even} \end{cases}$$

gives the desired pl-reduction from $p\text{-MC}(\text{func-}\Sigma_t^s)$ to $p\text{-MC}(\Sigma_t^{s+3})$. This finishes the proof of statement (1).

The proof of statement (2) is now easy. By Corollary 4.3, it is enough to show

$$p\text{-MC}(\text{func-FO}^s) \leq_{pl} p\text{-MC}(\text{FO}^{s+3}).$$

This is witnessed by the reduction mapping an instance $(\mathbf{A}, \varphi)$ of $p\text{-MC}(\text{func-FO}^s)$ to the instance $(\mathbf{A}', trans_\varphi^\exists)$, defined as above. $\square$

5. PATH AND OPTIMALITY OF SAVITCH'S THEOREM

Savitch's Theorem is a milestone result linking nondeterministic space to deterministic space.

> STCON
> *Instance:* A directed graph $\mathbf{G}$ and two vertices $s, t \in G$.
> *Problem:* Is there a (directed) path from s to t in $\mathbf{G}$?

Theorem 5.1 (Savitch [Sav70])**.** STCON $\in \text{SPACE}(\log^2 n)$. *In particular,*

$$\text{NL} \subseteq \text{SPACE}(\log^2 n).$$

The second statement follows from the first via the following proposition, itself a direct consequence of the fact that STCON is complete for NL under logarithmic space reductions (see e.g. [AB09, Theorem 4.18]).

Proposition 5.2. *Let* $s : \mathbb{N} \to \mathbb{N}$ *and assume* STCON $\in \text{SPACE}(s)$. *Then*

$$\text{NL} \subseteq \text{SPACE}\left(s(n^{O(1)}) + \log n\right).$$

In this section we prove a stronger version of Theorem 1.2 and Theorem 1.3. Additionally, we explain what a collapse of the parameterized classes PATH and para-L means in terms of classical complexity classes.

5.1. **Proof of Theorem 1.2.** Recall we say *Savitch's Theorem is optimal* if

$$\text{NL} \not\subseteq \text{SPACE}(o(\log^2 n)).$$

In this subsection we prove:

Theorem 1.2. *If Savitch's Theorem is optimal, then* PATH $\not\subseteq$ para-L.

In fact, we shall prove something stronger, namely Theorem 5.3 below: its assumption for *computable f* is equivalent to PATH $\subseteq$ para-L by Theorem 2.4; its conclusion implies that Savitch's Theorem is not optimal via Proposition 5.2. We shall use this stronger statement when proving Theorem 1.3 in the next subsection.

Theorem 5.3. *Assume there is an algorithm deciding p-STCON$_\leq$ that on instance $(\mathbf{G}, s, t, k)$ runs in space*

$$f(k) + O(\log |G|) \tag{5.1}$$

for some $f : \mathbb{N} \to \mathbb{N}$ (not necessarily computable). Then STCON $\in$ SPACE$(o(\log^2 n))$.

Proof. Choose an algorithm $\mathbb{A}$ and a function f according to the assumption. Without loss of generality, assume that $f(k) \geq k$ for every $k \in \mathbb{N}$. Let $\iota : \mathbb{N} \to \mathbb{N}$ be a non-decreasing and unbounded function such that for all $n \in \mathbb{N}$

$$f(\iota(n)) \leq \log n, \tag{5.2}$$

and hence

$$\iota(n) \leq \log n. \tag{5.3}$$

Note that we might not know how to compute $\iota(n)$.

Now let $\mathbf{G}$ be a directed graph, $s, t \in G$, $n := |G|$, and $k \geq 2$. We compute in space $O(\log k + \log n)$ the *minimum* $\ell := \ell(k) \in \mathbb{N}$ with

$$k^\ell \geq n - 1, \tag{5.4}$$

which implies

$$\ell \leq O\left(\frac{\log n}{\log k}\right). \tag{5.5}$$

Then we define a sequence of directed graphs $\left(\mathbf{G}_i^k\right)_{i \in \{0,\dots,\ell\}}$ with self-loops, as follows. For every $i \geq 0$ the vertex set G_i^k of $\mathbf{G}_i^k$ is G, the vertex set of $\mathbf{G}$. There is an edge in $\mathbf{G}_i^k$ from a vertex u to a vertex v if and only if there is a directed path from u to v in $\mathbf{G}$ of length at most k^i. In particular, $E^{\mathbf{G}_0^k}$ is the reflexive closure of $E^{\mathbf{G}}$. By (5.4)

there is a path from s to t in $\mathbf{G}$ $\Longleftrightarrow$ there is an edge from s to t in $\mathbf{G}_\ell^k$.   (5.6)

Furthermore, for every $i \in [\ell]$ and $u, v \in G_i^k = G_{i-1}^k = G$

there is an edge from u to v in $\mathbf{G}_i^k$

$$\Longleftrightarrow \text{ there is a path from } u \text{ to } v \text{ in } \mathbf{G}_{i-1}^k \text{ of length at most } k.$$

This can be decided by the following *recursive* algorithm:

Algorithm $\mathbb{C}$
input: a directed graph $\mathbf{G}$, $k, i \in \mathbb{N}$, and $u, v \in G$
output: decide whether there is an edge in $\mathbf{G}_i^k$ from u to v.
  1. **if** $i = 0$ **then** output whether $\left(u = v \text{ or } (u,v) \in E^{\mathbf{G}}\right)$ and return
  2. simulate $\mathbb{A}$ on $\left(\mathbf{G}_{i-1}^k, u, v, k\right)$
  3.   **if** in the simulation of $\mathbb{A}$ queries "$(u', v') \in E^{\mathbf{G}_{i-1}^k}$?"
  4.   **then** call $\mathbb{C}(\mathbf{G}, k, i-1, u', v')$.

For every $k \geq 2$ let $\mathbb{C}^k$ be the algorithm which on every directed graph $\mathbf{G}$ and $s, t \in G$ first computes $\ell = \ell(k)$ as in (5.4) and then simulates $\mathbb{C}(\mathbf{G}, k, \ell, s, t)$. Thus, $\mathbb{C}^k$ decides whether there is a path from s to t in $\mathbf{G}$ by (5.6). We analyse its space complexity. First, the depth of the recursion tree is ℓ, as $\mathbb{C}^k$ recurses on $i = \ell, \ell - 1, \dots, 0$. As usual, $\mathbb{C}^k$ has to maintain a stack of intermediate configurations for the simulations of

$$\mathbb{A}(\mathbf{G}_\ell^k, \text{-}, \text{-}, k), \mathbb{A}(\mathbf{G}_{\ell-1}^k, \text{-}, \text{-}, k), \dots, \mathbb{A}(\mathbf{G}_0^k, \text{-}, \text{-}, k).$$

For the simulation of each $\mathbb{A}(\mathbf{G}_i^k, \_, \_, k)$, the size of the configuration is linearly bounded by $f(k) + O(\log n)$ because of (5.1). Therefore, the total space required is

$$O\Big(\log k + \log n + \ell \cdot \big(f(k) + \log n\big)\Big) \leq O\left(\log k + \frac{f(k) \cdot \log n + \log^2 n}{\log k}\right) \qquad (5.7)$$

by (5.5). As a consequence, in case $k = \iota(n)$, (5.2) and (5.3) imply that (5.7) is bounded by $o(\log^2 n)$. So if $\iota(n)$ would be computable, we could replace it with a logspace computable function, and then the result would follow. In particular, under the assumption PATH = para-L of Theorem 1.2, we can assume f is computable, and hence find a computable ι.

In order to circumvent the possible uncomputability of $\iota(n)$ we adopt the strategy underlying Levin's optimal inverters [Lev73, CF14]. Namely, we simulate all the algorithms $\mathbb{C}^2, \mathbb{C}^3, \ldots$ in a diagonal fashion, while slowly increasing the allowed space.

Algorithm $\mathbb{S}$
input: a graph $\mathbf{G}$ and $s, t \in G$
output: decide whether there is a path in $\mathbf{G}$ from s to t.
  1. $S \leftarrow 2$
  2. **for all** $i = 2$ **to** S **do**
  3.   simulate $\mathbb{C}^i$ on $(\mathbf{G}, s, t)$ in *space* S
  4.   **if** the simulation accepts or rejects in space S
  5.   **then** accept or reject accordingly
  6. $S \leftarrow S + 1$
  7. goto 2.

Clearly, $\mathbb{S}$ decides STCON. We prove that its space complexity is $o(\log^2 n)$ on every input graph $\mathbf{G}$ with $n := |G|$. To that end, let $k := \iota(n)$ and s^* be the space needed by $\mathbb{C}^k(\mathbf{G}, s, t)$. As argued before, we have $s^* \leq o(\log^2 n)$. Observe that $\mathbb{S}$ must halt no later than S reaches the value $\max(\iota(n), s^*)$, which is again bounded by $o(\log^2 n)$ due to (5.3). This concludes the proof. $\qquad \square$

5.2. **Proof of Theorem 1.3.** For the reader's convenience, we repeat the statement of the theorem:

Theorem 1.3. *Let f be an arbitrary function from $\mathbb{N}$ to $\mathbb{N}$. If Savitch's Theorem is optimal, then p-MC(Σ_1^2) is not decidable in space $o(f(|\varphi|) \cdot \log |\mathbf{A}|)$.*

Note that neither f nor the function hidden in the $o(\ldots)$-notation is assumed to be computable. It is here where our stronger version Theorem 5.3 of Theorem 1.2 becomes useful.

Proof. Assume p-MC(Σ_1^2) is decidable in space $o(f(|\varphi|) \cdot \log |\mathbf{A}|)$ for some function $f : \mathbb{N} \to \mathbb{N}$, i.e. space $g\big(f(|\varphi|) \cdot \log |\mathbf{A}|\big)$ for some function $g : \mathbb{N} \to \mathbb{N}$ with

$$\lim_{m \to \infty} \frac{g(m)}{m} = 0. \qquad (5.8)$$

By Theorem 5.3 it suffices to show that there exist an arbitrary function $h : \mathbb{N} \to \mathbb{N}$ and an algorithm deciding p-STCON$_\leq$ that on an instance $(\mathbf{G}, s, t, k)$ runs in space

$$h(k) + O(\log |G|). \qquad (5.9)$$

Example 2.1 defines a pl-reduction from p-STCON$_\leq$ to p-MC(Σ_1^2) that maps an instance $(\mathbf{G}, s, t, k)$ of p-STCON$_\leq$ to an instance $(\varphi, \mathbf{A})$ of p-MC(Σ_1^2) such that $|\mathbf{A}| \leq O(|\mathbf{G}|)$ and $|\varphi| \leq c \cdot k$ for some constant $c \in \mathbb{N}$. Then, by the assumption and this reduction, there is an algorithm $\mathbb{A}$ which decides p-STCON$_\leq$ in space

$$g\big(f(c \cdot k) \cdot \log |\mathbf{G}|\big),$$

Here, we assume without loss of generality that f is non-decreasing, so $f(|\varphi|) \leq f(c \cdot k)$. By (5.8), for every $\ell \in \mathbb{N}$ there is $m_\ell \in \mathbb{N}$ such that for every $m \geq m_\ell$ we have

$$\frac{g(m)}{m} \leq \frac{1}{\ell}.$$

Thus, for $\ell := f(c \cdot k)$ and $m := f(c \cdot k) \cdot \log |\mathbf{G}|$

$$g\big(f(c \cdot k) \cdot \log |\mathbf{G}|\big) = g\big(m\big) \leq \max_{m < m_\ell} g(m) + \frac{m}{\ell} = \max_{m < m_\ell} g(m) + \log |\mathbf{G}|.$$

Note $\max_{m < m_\ell} g(m)$ only depends on ℓ, and hence only on the parameter k. Consequently the space required by $\mathbb{A}$ can be bounded by (5.9) for an appropriate function $h : \mathbb{N} \to \mathbb{N}$. $\square$

5.3. Bounded nondeterminism in logarithmic space.
We close this section showing that the collapse of the parameterized classes para-L and PATH can be characterized as a collapse of classical classes L and NL restricted to 'arbitrarily few but non-trivially many' nondeterministic bits.

Definition 5.4. Let $c : \mathbb{N} \to \mathbb{N}$ be a function. The class NL$[c]$ contains all classical problems Q that are accepted by some nondeterministic Turing machine which uses $c(|x|)$ many nondeterministic bits and runs in logarithmic space.

Proposition 5.5. *The following are equivalent.*
(1) para-L = PATH.
(2) *There exists a space-constructible[6] function $c(n) \geq \omega(\log(n))$ such that* NL$[c] = $ L.

Proof. To see that (1) implies (2), assume para-L = PATH. Then there is a machine $\mathbb{A}$ deciding p-STCON$_\leq$ which on an instance $(\mathbf{G}, s, t, k)$ runs in space $f(k) + O(\log |G|)$ for some computable $f : \mathbb{N} \to \mathbb{N}$. We can assume that f is increasing and space-constructible (see [FG06, Lemma 1.35] for a similar construction). Then there is an unbounded, logarithmic space computable $\iota : \mathbb{N} \to \mathbb{N}$ such that $f(\iota(n)) \leq \log n$ for all $n \in \mathbb{N}$. Then

$$c(n) := \iota(n) \cdot \log n$$

is space-constructible and $c(n) \geq \omega(\log(n))$. We claim that NL$[c] = $ L.

Let $Q \in$ NL$[c]$ be given. Choose a machine $\mathbb{B}$ accepting Q that on input x uses at most $s = O(\log |x|)$ space and at most $\iota(|x|) \cdot \log |x|$ many nondeterministic bits. We assume there is at most one accepting space s configuration c_{acc} that can possibly appear in any run of $\mathbb{B}$ on x.

The digraph $\mathbf{G}$ has as vertices all space s configurations of $\mathbb{B}$ on x and an edge from u to v if there is a computation of $\mathbb{B}$ started at u leading to v which uses at most $\log |x|$ many nondeterministic bits and space at most s. Note this can be decided in logarithmic space by simulating $\mathbb{B}$ exhaustively for all possible outcomes of guesses.

---

[6]Recall, $c : \mathbb{N} \to \mathbb{N}$ is *space-constructible* if $c(n)$ can be computed from n in space $O(c(n))$.

Since ι is logarithmic space computable, the instance $\big(\mathbf{G}, c_{start}, c_{acc}, \iota(|x|)\big)$ of p-STCON$_{\leq}$ is implicitly logarithmic space computable from x; here, c_{start} is the starting configuration of $\mathbb{B}$ on x. This is a "yes" instance if and only if $x \in Q$. Given this input, $\mathbb{A}$ needs space

$$f(\iota(|x|)) + O(\log|G|) \leq O(\log|x|).$$

To see that (2) implies (1), assume $c(n) \geq \omega(\log(n))$ is space-constructible and $\text{NL}[c] = \text{L}$. There is a logarithmic space computable, non-decreasing and unbounded function $\iota : \mathbb{N} \to \mathbb{N}$ such that $c(n) \geq \iota(n) \cdot \lceil \log n \rceil$ for all $n \in \mathbb{N}$. By Theorem 2.4 it suffices to show that p-STCON$_{\leq}$ can be decided in parameterized logarithmic space.

Define the classical problem

$$Q := \big\{(\mathbf{G}, s, t, k) \in \text{STCON}_{\leq} \mid k \leq \iota(|G|)\big\}.$$

Then $Q \in \text{NL}[c]$ by a straightforward guess and check algorithm. By assumption, Q is decided by a logarithmic space algorithm $\mathbb{A}$. Then we solve p-STCON$_{\leq}$ as follows using some arbitrary "brute force" (deterministic) algorithm $\mathbb{B}$ deciding p-STCON$_{\leq}$. Given an instance $(\mathbf{G}, s, t, k)$ we check whether $k \leq \iota(|G|)$; if this is the case, we run $\mathbb{A}$ and otherwise $\mathbb{B}$. In the first case we consume only logarithmic space. In the second case, the space is effectively bounded in k because the instance size is. Indeed, if $k > \iota(|G|)$, then $|G| < f(k)$, where f is a computable function such that $f \circ \iota(n) \geq n$ for all $n \in \mathbb{N}$. $\qquad\square$

Remark 5.6. There are similar characterizations of W[P] = FPT in [CCDF95, Theorem 3.8], EW[P] = EPT in [FGW06, Theorem 4], and BPFPT = FPT in [MM13, Theorem 5.2].

We find it worthwhile to point out explicitly the following direct corollary.

Corollary 5.7. *If Savitch's Theorem is optimal, then* $\text{L} \neq \text{NL}[c]$ *for all space-constructible functions* $c(n) \geq \omega(\log n)$.

Proof. By Proposition 5.5 and Theorem 1.2. $\qquad\square$

6. A DETERMINISTIC MODEL-CHECKER

In this section we prove

Theorem 1.4. *For all* $s, t \in \mathbb{N}$ *the problem* p-MC(Σ_t^s) *is decidable in space*

$$O(\log|\varphi| \cdot (\log|\varphi| + \log|\mathbf{A}|)).$$

This claims for each $s \in \mathbb{N}$ the existence of an algorithm. Our algorithm is going to be uniform in s, so we give a more general statement formulating the space bound using the *width* $w(\varphi)$ of a formula φ. This is the maximal number of free variables in some subformula of φ.

Being interested in the dependence of the space complexity on the parameter $|\varphi|$, we formulate all results as statements about the classical model-checking problem MC(Φ) for various classes Φ of first-order sentences. In this section we allow besides relation symbols also constants in the vocabulary of formulas in Φ. Abusing notation we continue to use notation like Σ_t and Π_t, now understanding that the formulas may contain constants.

Besides $|\varphi|$ it is technically convenient to also consider another size measure of φ, namely the number of subformulas $\|\varphi\|$ of φ counted with repetitions. In other words this is the number of nodes in the syntax tree of φ. Formally, it is defined by a recursion on the syntax:

$$
\begin{aligned}
\|\varphi\| &:= 1 & \text{for atomic } \varphi \\
\|(\varphi * \psi)\| &:= 1 + \|\varphi\| + \|\psi\| & \text{for } * \in \{\wedge, \vee\} \\
\|\neg\varphi\| &:= 1 + \|\varphi\| \\
\|Qx\varphi\| &:= 1 + \|\varphi\| & \text{for } Q \in \{\forall, \exists\}
\end{aligned}
$$

Clearly we have $\|\varphi\| \leq |\varphi|$. Hence, the following result, which is the main result of this section, implies Theorem 1.4.

Theorem 6.1. *For all $t \geq 1$ there is an algorithm deciding $\mathrm{MC}(\Sigma_t)$ that on an instance $(\varphi, \mathbf{A})$ of $\mathrm{MC}(\Sigma_t)$ runs in space*

$$
O\big(\log\|\varphi\| \cdot w(\varphi) \cdot \log|A| + \log\|\varphi\| \cdot \log|\varphi| + \log|\mathbf{A}|\big).
$$

We first give an intuitive outline of the proof. The heart of the argument is the proof for the case $t = 1$, the extension to $t \geq 1$ is straightforward. The case $t = 1$ is isolated as Proposition 6.3, and proved by a recursive divide and conquer approach. Namely, given an instance $(\mathbf{A}, \varphi)$ of $\mathrm{MC}(\Sigma_1)$, our model-checker views φ as a tree and computes a node that splits the tree in a 1/3–2/3 fashion. More precisely, it computes a subformula $\varphi_0(\bar{y})$ of φ of size $\|\varphi_0(\bar{y})\|$ between 1/3 and 2/3 of $\|\varphi\|$. Our model-checker loops through all possible assignments $\bar{b}$ to the free variables $\bar{y}$ of $\varphi_0(\bar{y})$ and recurses to $\varphi_0(\bar{b})$. Returning from this call it recurses to the "rest" formula $\varphi_1^{\bar{b}}$. Intuitively this is the formula obtained by replacing $\varphi_0(\bar{b})$ by its truth value.

As long as the formulas in the recursion are large enough, they shrink in each recursive step by a constant fraction. When the recursion reaches a small formula it applies "brute force". Hence the recursion tree is of depth $O(\log\|\varphi\|)$. Since the tuples $\bar{b}$ from the loops can be stored in space $O(w(\varphi) \cdot \log|A|)$, this sketch should explain the main term $\log\|\varphi\| \cdot w(\varphi) \cdot \log|A|$ in the final space bound.

We first describe the "brute force" subroutine mentioned above, a folklore, straightforwardly defined model-checker:

Lemma 6.2. *There is an algorithm deciding $\mathrm{MC}(\mathrm{FO})$ that on an instance $(\varphi, \mathbf{A})$ of $\mathrm{MC}(\mathrm{FO})$ runs in space*

$$
O\big(\|\varphi\| \cdot \log|A| + \log|\varphi| + \log|\mathbf{A}|\big).
$$

Proof. We describe an algorithm $\mathbb{B}$ that decides the slightly more general (classical) problem

Instance:	a formula $\varphi = \varphi(\bar{x})$, a structure $\mathbf{A}$ and a tuple $\bar{a} \in A^{|\bar{x}|}$.
Problem:	$\mathbf{A} \models \varphi(\bar{a})$?

in space $O(\|\varphi\| \cdot \log|A| + \log n)$ on an instance $(\varphi, \mathbf{A}, \bar{a})$ of length n. The algorithm $\mathbb{B}$ implements a straightforward recursion on the logical syntax of φ.

If $\varphi(\bar{x}) = (\chi(\bar{x}) \wedge \psi(\bar{x}))$, it calls $\mathbb{B}(\chi(\bar{x}), \mathbf{A}, \bar{a})$. Upon returning from this call, $\mathbb{B}$ stores its answer, i.e. the bit b giving the truth value of $\mathbf{A} \models \chi(\bar{a})$. Then $\mathbb{B}$ calls $\mathbb{B}(\psi(\bar{x}), \mathbf{A}, \bar{a})$. Upon returning from this call with answer b', $\mathbb{B}$ answers the bit $b \cdot b'$.

If $\varphi(\bar{x}) = \exists x \chi(\bar{x}, x)$, then $\mathbb{B}$ loops over $b \in A$ and calls $\mathbb{B}(\psi(\bar{x}, x), \mathbf{A}, \bar{a}b)$; it answers with the maximal answer bit obtained.

The cases where φ is a negation, a disjunction or starts with $\forall$ are similarly explained.

If φ is atomic, it has the form $t_1=t_2$ or $R(t_1,\ldots,t_r)$ where R is an r-ary relation symbol and the t_i's are constants or variables. Assume the latter. Then $\mathbb{B}$ checks whether there exists $j \in [\|R^{\mathbf{A}}\|]$ such that for all $i \in [r]$ it holds that t_i^A equals the i-th component of the j-th tuple in $R^{\mathbf{A}}$. Here, t_i^A is $c^{\mathbf{A}}$ if t_i is a constant c; otherwise t_i is a free variable in φ and t_i^A is the corresponding component of $\bar{a}$.

To implement the recursion $\mathbb{B}$ stores a stack collecting the b's of the loops and the answer bits generated by the recursion as described. Scanning the whole stack allows us to determine in space $O(\log n)$ the formula $\psi(\bar{x},\bar{y})$ and tuple $\bar{b}$ such that the corresponding recursive call is $\mathbb{B}(\psi(\bar{x},\bar{y}),\mathbf{A},\bar{a}\bar{b})$. The depth of the recursion is at most $\|\varphi\|$, so the stack can be stored in space $\|\varphi\| \cdot (\log|A|+1)$. On an atomic formula $\mathbb{B}$ needs space $\log|\mathbf{A}| + \log|\varphi| \leq O(\log n)$ for the loops on j and i, and again $O(\log n)$ for the equality checks. Altogether we see that $\mathbb{B}$ can be implemented within the claimed space. $\qquad\square$

The following proposition is the heart of the argument. The advantage with respect to the "brute force" algorithm from the previous proposition is that the factor $\|\varphi\|$ in the space bound is replaced by $\log\|\varphi\|$. But since other factors are worsened this algorithm is not in general more space efficient.

Proposition 6.3. *There is an algorithm deciding* $\mathrm{MC}(\Sigma_1)$ *that on an instance* $(\varphi,\mathbf{A})$ *of* $\mathrm{MC}(\Sigma_1)$ *runs in space*

$$O\big(\log\|\varphi\| \cdot w(\varphi) \cdot \log|A| + \log\|\varphi\| \cdot \log|\varphi| + \log|\mathbf{A}|\big). \tag{6.1}$$

Proof. We describe an algorithm $\mathbb{A}$ deciding the problem

> *Instance:* a Σ_1-formula φ, a natural number $w \geq w(\varphi)$, a structure $\mathbf{A}$ and $\bar{a} \in A^w$.
> *Problem:* $\mathbf{A} \models \varphi(\bar{a})$?

on an instance $(\varphi,w,\mathbf{A},\bar{a})$ of length n in *allowed space*

$$O\big(\log\|\varphi\| \cdot w \cdot \log|A| + \log\|\varphi\| \cdot \log|\varphi| + \log n\big).$$

Notationally, $\mathbf{A} \models \varphi(\bar{a})$ means that the assignment that maps the i-th free variable in φ to the i-th component of $\bar{a}$ satisfies φ in $\mathbf{A}$. Here we suppose an order on the variables in φ, say according to appearance in φ. All we need is that the value assigned to a given variable in a given subformula of φ according to a given tuple $\bar{a} \in A^w$ can be determined in space $O(\log n)$.

For a sufficiently large constant $c \in \mathbb{N}$ to be determined in the course of the proof, $\mathbb{A}$ checks that

$$\|\varphi\| \geq c \cdot w + c. \tag{6.2}$$

If this is not the case, then $\mathbb{A}$ uses "brute force", that is, it runs the algorithm from Lemma 6.2.

Now suppose (6.2) holds. Choosing $c \geq 3$ this implies that the syntax tree of φ has at least 3 nodes. Then $\mathbb{A}$ computes in space $O(\log|\varphi|)$ a subformula φ_0 of φ such that

$$\|\varphi\|/3 \leq \|\varphi_0\| \leq 2\|\varphi\|/3. \tag{6.3}$$

Observe that φ_0 is a Σ_1-formula. Let $\bar{y} = y_1 \cdots y_{|\bar{y}|}$ list the free variables of φ_0 and note

$$|\bar{y}| \leq w(\varphi_0) \leq w(\varphi) \leq w. \tag{6.4}$$

Recall from the intuitive sketch of the proof that we intend to call $\mathbb{A}$ recursively on a "rest" formula $\varphi_1^{\bar{b}}$ where $\bar{b}$ is a $|\bar{y}|$-tuple from A. To define this formula, let $c_1,\ldots,c_{\max\{|\bar{y}|,1\}}$ be new constant symbols. For every free variable y_i of φ_0 check whether it has an occurrence

in φ_0 which is not a free occurrence in φ. If such an occurrence exists, all the free occurrences of y_i in φ_0 appear within a uniquely determined subformula $\exists y_i \chi$ of φ containing φ_0 as a subformula, where $\exists y_i$ binds these occurrences, that is, the free occurrences of y_i in φ_0 are also free in χ. Replace in φ the subformula $\exists y_i \chi$ by $\exists y_i (y_i = c_i \wedge \chi)$. Let φ_1 denote the resulting formula. Note that φ_1 does not depend on the order of how these replacements for the y_i are performed.

Moreover, using (6.4),

$$\|\varphi_1\| \leq \|\varphi\| + 2|\bar{y}| \leq \|\varphi\| + 2w. \tag{6.5}$$

The algorithm $\mathbb{A}$ then loops through $\bar{b} = (b_1, \ldots, b_{|\bar{y}|}) \in A^{|\bar{y}|}$ and does two recursive calls:

(R0) Recursively call $\mathbb{A}(\varphi_0, w, \mathbf{A}, \bar{b})$ to check whether $\mathbf{A} \models \varphi_0(\bar{b})$. If $\mathbf{A} \models \varphi_0(\bar{b})$, then replace the subformula φ_0 in φ_1 by $c_1 = c_1$; otherwise by $\neg c_1 = c_1$.

Let $\varphi_1^{\bar{b}}$ be the resulting formula. Further, let $\mathbf{A}^{\bar{b}}$ be the expansion of $\mathbf{A}$ that interprets the constants $c_1, \ldots, c_{|\bar{y}|}$ by $b_1, \ldots, b_{|\bar{y}|}$ respectively.

(R1) Recursively call $\mathbb{A}\big(\varphi_1^{\bar{b}}, w, \mathbf{A}^{\bar{b}}, \bar{a}\big)$ and output its answer.

Note that in (R1) we have $w(\varphi_1^{\bar{b}}) \leq w(\varphi) \leq w$, so the algorithm recurses to an instance of our problem. It is routine to verify that

$$\mathbf{A} \models \varphi(\bar{a}) \quad \Longleftrightarrow \quad \text{there exists } \bar{b} \in A^{|\bar{y}|} \text{ such that } \mathbf{A}^{\bar{b}} \models \varphi_1^{\bar{b}}(\bar{a}).$$

Thus, $\mathbb{A}$ correctly decides whether $\mathbf{A} \models \varphi(\bar{a})$. It remains to show that $\mathbb{A}$ can be implemented in the allowed space.

We first estimate the depth of the recursion. In (R0) the algorithm recurses to formula φ_0 and $\|\varphi_0\| \leq 2\|\varphi\|/3$ by (6.3). In (R1) the algorithm recurses on $\varphi_1^{\bar{b}}$, a formula obtained from φ_1 by replacing the subformula φ_0 by an atomic formula or the negation of an atomic formula. Then

$$\|\varphi_1^{\bar{b}}\| \leq \|\varphi\| + 2w - \|\varphi_0\| + 2 \leq 2\|\varphi\|/3 + 2w + 2 = 3\|\varphi\|/4 + (2w + 2 - \|\varphi\|/12)$$

where the inequalities hold by (6.5) and (6.3), respectively. Provided c in (6.2) is large enough, this implies $\|\varphi_1^{\bar{b}}\| \leq 3\|\varphi\|/4$. It follows that the recursion depth is $O(\log\|\varphi\|)$.

In each recursive call the structure the algorithm recurses to is determined by a tuple $\bar{b}$ from the loop. This tuple has length at most w (cf. (6.4)). The formula $\varphi_1^{\bar{b}}$ in (R1) is determined by the truth value of $\mathbf{A} \models \varphi_0(\bar{b})$. The algorithm recurses either

(P0) to the formula φ_0, or

(P1) to the formula $\varphi_1^{\bar{b}}$ as defined if $\mathbf{A} \models \varphi_0(\bar{b})$, or

(P2) to the formula $\varphi_1^{\bar{b}}$ as defined if $\mathbf{A} \not\models \varphi_0(\bar{b})$.

To implement the recursion, $\mathbb{A}$ maintains a sequence of tuples $\bar{b}_1, \ldots, \bar{b}_d$ and a sequence of "possibilities" $(p_1, \ldots, p_d) \in \{0, 1, 2\}^d$. The length d is bounded by $O(\log\|\varphi\|)$, the depth of the recursion. In a recursive call as described above the sequence of tuples is expanded by the tuple $\bar{b}$ from the loop, and the possibility sequence by $0, 1, 2$ depending on which recursion from (P0), (P1), (P2) is taken. Both sequences can be stored in allowed space, namely

$$O(d \cdot w \cdot \log|A|) \leq O(\log\|\varphi\| \cdot w \cdot \log|A|).$$

We still have to explain how, given $(\bar{b}_1, \ldots, \bar{b}_d)$ and $(p_1, \ldots, p_d)$, the algorithm determines the corresponding formula ψ and structure $\mathbf{B}$ it has to recurse to. The structure $\mathbf{B}$ is an expansion of the input structure $\mathbf{A}$ by a sequence of constants interpreted by the sequence

$\bar{b}_{i_1} \cdots \bar{b}_{i_\ell}$ where $i_1 < \cdots < i_\ell$ list those indices $j \in [d]$ such that $p_j \neq 0$. This structure can be computed from the two sequences and the input in logarithmic space. Note that $\mathbf{B}$ is an expansion of $\mathbf{A}$ by interpreting at most $w \cdot d$ many constants, so

$$\log |\mathbf{B}| \leq \log(|\mathbf{A}| + 2 \cdot w \cdot d) \leq \log |\mathbf{A}| + \log w + \log \log \|\varphi\| + O(1). \qquad (6.6)$$

To determine the formula ψ, consider the function R that maps a formula χ and a number $i \leq 2$ to the formula according to (Pi). Since we defined the recursion possibilities (Pi) only on formulas satisfying (6.2), let us agree that $R(\chi, i) := \chi$ on formulas violating (6.2). The desired formula ψ is obtained by iterating this function along $(p_1, \ldots, p_d)$: compute $\psi_1 := R(\varphi, p_1)$, $\psi_2 := R(\psi_1, p_2)$, $\ldots$ and output $\psi := \psi_d$. Note that all these formulas have length $O(|\varphi|)$, and each iteration step is computable in space $O(\log |\varphi|)$. Hence, the whole iteration can be implemented in space

$$O(d \cdot \log |\varphi|) \leq O(\log \|\varphi\| \cdot \log |\varphi|).$$

If $(\psi, w, \mathbf{B}, \bar{b})$ is such that $\|\psi\|$ violates (6.2), i.e. $\|\psi\| < c \cdot w + c$, then $\mathbb{A}$ invokes the "brute force" algorithm from Lemma 6.2. This requires space

$$O\big(w \cdot \log |A| + \log |\psi| + \log |\mathbf{B}|\big).$$

By $|\psi| \leq O(|\varphi|)$ and (6.6), this is allowed space. $\qquad \square$

Remark 6.4. Recall Example 2.1 defines a pl-reduction from p-STCON$_\leq$ to p-MC(Σ_1^2) that maps an instance $(\mathbf{G}, s, t, k)$ of p-STCON$_\leq$ to an instance $(\mathbf{A}, \varphi)$ of p-MC($\bar{\Sigma}_1^2$) with $|\varphi| \leq O(k)$. Combining with the algorithm from Proposition 6.3 we thus decide p-STCON$_\leq$ in space

$$O(\log k \cdot \log |G|). \qquad (6.7)$$

This is a slightly more detailed statement of Savitch's Theorem 5.3. As pointed out by an anonymous reviewer, a kind of converse holds, namely, Proposition 6.3 can be derived from Savitch's Theorem by means of a reduction:

Sketch of a second proof of Proposition 6.3. Given an instance $(\mathbf{A}, \varphi)$ of p-MC(Σ_1) construct the following directed graph $\mathbf{G}$. Its vertices G are triples (ψ, α, b) where $b \in \{0, 1\}$ is a bit and ψ, α are as in the proof of Theorem 4.1: ψ is a subformula of φ and α is an assignment of its free variables. Again we assume that negations appear only in front of atoms. Note

$$|G| \leq 2 \cdot \|\varphi\| \cdot |A|^{w(\varphi)}. \qquad (6.8)$$

The goal is to define the edges of $\mathbf{G}$ in such a way that there is a path from $s := (\varphi, \emptyset, 0)$ to $t := (\varphi, \emptyset, 1)$ in $\mathbf{G}$ if and only if $\mathbf{A} \models \varphi$. Moreover, if this is the case, then there is such a path of length at most $k := \|\varphi\|$. This allows to decide whether $\mathbf{A} \models \varphi$ by running Savitch's algorithm on $(\mathbf{G}, s, t, k)$. By (6.7) and (6.8) this needs space (6.1) and thus proves Proposition 6.3 (it will be clear that this space suffices to construct $\mathbf{G}$).

It remains to define the edges of $\mathbf{G}$. If ψ is an atom or a negated atom, add an edge from $(\psi, \alpha, 0)$ to $(\psi, \alpha, 1)$ if and only if α satisfies ψ in $\mathbf{A}$.

If ψ is $(\psi_0 \vee \psi_1)$, add edges from $(\psi, \alpha, 0)$ to $(\psi_0, \alpha_0, 0)$ and $(\psi_1, \alpha_1, 0)$, and add edges from $(\psi_0, \alpha_0, 1)$ and $(\psi_1, \alpha_1, 1)$ to $(\psi, \alpha, 1)$; here α_0 and α_1 are the restrictions of α to the free variables of ψ_0 and ψ_1, respectively.

If ψ is $(\psi_0 \wedge \psi_1)$, add edges from $(\psi, \alpha, 0)$ to $(\psi_0, \alpha_0, 0)$, from $(\psi_0, \alpha_0, 1)$ to $(\psi_1, \alpha_1, 0)$, and from $(\psi_1, \alpha_1, 0)$ to $(\psi, \alpha, 1)$; here α_0, α_1 are defined as in the previous case.

If ψ is $\exists y \chi$, then add, for every $a \in A$, edges from $(\psi, \alpha, 0)$ to $(\chi, \beta_a, 0)$ and from $(\chi, \beta_a, 1)$ to $(\psi, \alpha, 1)$; here, β_a equals α if y is not free in χ, and otherwise extends α by mapping y to a. $\qquad \square$

We now extend the space bound from the previous proposition to $\mathrm{MC}(\Sigma_t)$ for each $t \geq 1$.

Proof of Theorem 6.1. Similarly as in the previous proposition, we give an algorithm $\mathbb{A}$ deciding

Instance:	a Σ_t-formula φ, a natural number $w \geq w(\varphi)$, a structure $\mathbf{A}$ and $\bar{a} \in A^w$.
Problem:	$\mathbf{A} \models \varphi(\bar{a})$?

on an instance $(\varphi, w, \mathbf{A}, \bar{a})$ of length n in *allowed space*

$$O\big(\log \|\varphi\| \cdot w \cdot \log |A| + \log \|\varphi\| \cdot \log |\varphi| + \log n\big).$$

For $t = 1$ this is what has been shown in the proof of Proposition 6.3. So we assume $t \geq 2$ and proceed inductively.

Let $\psi_1, \ldots, \psi_r \in \Pi_{t-1}$ be such that φ results from these formulas by existential quantification and positive Boolean combinations. Of course, such formulas ψ_j are computable in logarithmic space from φ. For $j \in [r]$ let $s_j \leq w(\varphi) \leq w$ be the number of variables occurring freely in ψ_j and let $\bar{x}_j$ be a length s_j tuple listing these variables.

Let $\mathbf{A}^*$ be the structure with universe A that interprets for each $j \in [r]$ an s_j-ary relation symbol R_j by

$$R_j^{\mathbf{A}^*} := \{\bar{b} \in A^{s_j} \mid \mathbf{A} \models \psi_j(\bar{b})\}.$$

By the induction hypothesis we can compute $\mathbf{A}^*$ in allowed space. Moreover

$$|\mathbf{A}^*| \leq \|\varphi\| + |A| + \|\varphi\| \cdot |A|^w. \tag{6.9}$$

Define the formula φ^* by replacing for every $j \in [r]$ the formula $\psi_j(\bar{x}_j)$ in φ by $R_j(\bar{x}_j)$. Clearly, we have $\varphi^* \in \Sigma_1$,

$$\|\varphi^*\| \leq \|\varphi\| \quad \text{and} \quad |\varphi^*| \leq O(|\varphi|). \tag{6.10}$$

More importantly,

$$\mathbf{A} \models \varphi(\bar{a}) \quad \Longleftrightarrow \quad \mathbf{A}^* \models \varphi^*(\bar{a}).$$

Since φ^* is a Σ_1-formula with $w(\varphi^*) \leq w(\varphi) \leq w$ we have that $(\varphi^*, w, \mathbf{A}^*, \bar{a})$ is an instance of the problem treated in the proof of Proposition 6.3. We can thus decide whether $\mathbf{A}^* \models \varphi^*(\bar{a})$ in space

$$O\big(\log \|\varphi^*\| \cdot w \cdot \log |A| + \log \|\varphi^*\| \cdot \log |\varphi^*| + \log |\mathbf{A}^*|\big)$$

By (6.10) and (6.9), this is allowed space. $\qquad \square$

7. Summary and future directions

We have studied the parameterized space complexity of model-checking bounded variable first-order logic, i.e. p-$\mathrm{MC}(\mathrm{FO}^s)$ for fixed $s \geq 2$. We stratified the problem into subproblems according quantifier alternation rank and showed (Theorem 4.1) that the respective subproblems p-$\mathrm{MC}(\Sigma_t^s)$ are complete for the levels of the tree-hierarchy $\mathrm{TREE}[t]$, the alternation hierarchy above the class TREE from [CM15]. We further showed that allowing function symbols does not increase the space complexity of these problems (Theorem 4.6). This gives a quite fine-grained picture of the space complexities up to pl-reductions.

However, it is open whether the tree-hierarchy is strict even under some plausible complexity assumption. We showed it does not collapse to para-L if Savitch's Theorem is optimal, in fact, then already PATH $\neq$ para-L (Theorem 1.2). It follows that p-$\mathrm{MC}(\Sigma_1^2)$

cannot be solved in parameterized logarithmic space if Savitch's Theorem is optimal. Under this assumption we proved a stronger result (Theorem 1.4) stating, intuitively, that the naïve model-checking algorithm is space-optimal.

Finally, Theorem 1.4 gives a highly space-efficient model-checking algorithm for p-MC(Σ_t^s). We presented two constructions, a direct one and another, pointed out to us by an anonymous reviewer, via a reduction of p-MC(Σ_1^2) to p-STCON$_\leq$ and Savitch's algorithm.

We view the results about the tree hierarchy as a contribution to the fine-structure theory of FPT (cf. [Mül14]). In fact, Theorems 2.7 and 1.1 indicate that the tree hierarchy contains many natural parameterized problems. However, as pointed out by an anonymous reviewer, Theorem 1.1 (2) implies that problems in TREE[∗] have shallow circuits, and hence, intuitively, TREE[∗] is a small subclass of FPT. Parameterized circuit complexity is another emerging theory about the fine-structure of FPT [EST15, BST15, BT18, CF18, CMY18, CF19].

We repeat the questions whether PATH or TREE are closed under complementation. We noted that a positive answer for TREE would imply a collapse of the tree hierarchy (Corollary 4.2). We do not know whether this also follows from PATH being closed under complementation.

As a further structural question we do not know how para-NL relates to the tree hierarchy. Proposition 3.4 (1) gives only a partial answer.

We showed that the straightforward model-checking algorithm is space-optimal under the hypothesis that Savitch's Theorem is optimal (Theorem 1.3). We conjecture that similar optimality results can be derived for other natural algorithms as well. Future work will show to what extent the hypothesis that Savitch's Theorem is optimal can play a role in space complexity similar to the one played by the ETH in time complexity.

Acknowledgements

References

[AB09]　　S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[BCD⁺89]　A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo, and M. Tompa. Two applications of inductive counting for complementation problems. *SIAM Journal on Computing*, 18(3):559–578, 1989.

[BST15]　　M. Bannach, C. Stockhusen, and T. Tantau. Fast parallel fixed-parameter algorithms via color coding. In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, pages 224–235, 2015.

[BT18]　　M. Bannach and T. Tantau. Computing hitting set kernels by acˆ0-circuits. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, pages 9:1–9:14, 2018.

[CCDF95]　L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. On the structure of parameterized problems in NP. *Information and Computation*, 123(1):38–49, 1995.

[CCDF97]　L. Cai, J. Chen, R. G. Downey, and M.l R. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, 84(1):119–138, 1997.

[CF14]　　Y. Chen and J. Flum. On optimal inverters. *The Bulletin of Symbolic Logic*, 20(01):1–23, 2014.

[CF18]   Y. Chen and J. Flum. Tree-depth, quantifier elimination, and quantifier rank. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 225–234, 2018.

[CF19]   Y. Chen and J. Flum. Some lower bounds in parameterized AC^0. *Information and Computation*, 267:116–134, 2019.

[CFG05]   Y. Chen, J. Flum, and M. Grohe. Machine-based methods in parameterized complexity theory. *Theoretical Computer Science*, 339(2-3):167–199, 2005.

[Che14]   H. Chen. On the complexity of existential positive queries. *ACM Transactions on Computational Logic*, 15(1):9:1–9:20, 2014.

[CM77]   A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 77–90, 1977.

[CM14]   Y. Chen and M. Müller. Bounded variable logic, parameterized logarithmic space, and Savitch's theorem. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, pages 183–195, 2014.

[CM15]   H. Chen and M. Müller. The fine classification of conjunctive queries and parameterized logarithmic space. *ACM Transactions on Computation Theory*, 7(2):7:1–7:27, 2015.

[CM17]   H. Chen and M. Müller. One hierarchy spawns another: Graph deconstructions and the complexity classification of conjunctive queries. *ACM Transactions on Computational Logic*, 18(4):29:1–29:37, 2017.

[CMY18]   Y. Chen, M. Müller, and K. Yokoyama. A parameterized halting problem, the linear time hierarchy, and the MRDP theorem. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 235–244, 2018.

[DKV02]   V. Dalmau, P. G. Kolaitis, and M. Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, pages 310–326, 2002.

[EF99]   H.-D. Ebbinghaus and J. Flum. *Finite model theory, second edition*. Perspectives in Mathematical Logic. Springer, 1999.

[EST15]   M. Elberfeld, C. Stockhusen, and T. Tantau. On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica*, 71(3):661–701, 2015.

[FG01]   J. Flum and M. Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM Journal on Computing*, 31(1):113–145, 2001.

[FG03]   J. Flum and M. Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.

[FG06]   J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[FGW06]   J. Flum, M. Grohe, and M. Weyer. Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits. *Journal of Computer and System Sciences*, 72(1):34–71, 2006.

[GIKW19] J. Gao, R. Impagliazzo, A. Kolokolova, and R. Williams. Completeness for first-order properties on sparse structures with algorithmic applications. *ACM Transactions on Algorithms*, 15(2):23:1–23:35, 2019.

[GK11]   M. Grohe and S. Kreutzer. Methods for algorithmic meta theorems. In *Model Theoretic Methods in Finite Combinatorics*, volume 558 of *Contemporary Mathematics*. AMS, 2011.

[GKS17]   M. Grohe, S. Kreutzer, and S. Siebertz. Deciding first-order properties of nowhere dense graphs. *Journal of the ACM*, 64(3):17:1–17:32, 2017.

[Gro07]   M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1):1:1–1:24, 2007.

[Gro08]   M. Grohe. Logic, graphs, and algorithms. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and Automata: History and Perspectives in Honor of Wolfgang Thomas*, volume 2 of *Texts in Logic and Games*, pages 357–422. Amsterdam University Press, 2008.

[Gro14]   M. Grohe. Algorithmic meta theorems for sparse graph classes. In *Computer Science - Theory and Applications - 9th International Computer Science Symposium in Russia, CSR 2014, Moscow, Russia, June 7-11, 2014. Proceedings*, pages 16–22, 2014.

[GSS01]   M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 657–666, 2001.

[HOT94]   L. A. Hemaspaandra, M. Ogihara, and S. Toda. Space-efficient recognition of sparse self-reducible languages. *Computational Complexity*, 4:262–296, 1994.

[Imm82]   N. Immerman. Upper and lower bounds for first order expressibility. *Journal of Computer and System Sciences*, 25(1):76–98, 1982.

[Imm88]   N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, 1988.

[KV00]   P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61(2):302–332, 2000.

[Lev73]   L. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[Lip10]   J. R. Lipton. *The P = NP Question and Gödel's Lost Letter*. Springer, 2010.

[MM13]   J. A. Montoya and M. Müller. Parameterized random complexity. *Theory of Computing Systems*, 52(2):221–270, 2013.

[Mül14]   M. Müller. Parameterized logarithmic space and fine structure of FPT. *FPT News: The Parameterized Complexity Newsletter*, 10(2), September 2014.

[Pot17]   A. Potechin. Bounds on monotone switching networks for directed connectivity. *Journal of the ACM*, 64(4):29:1–29:48, 2017.

[Ruz80]   W. L. Ruzzo. Tree-size bounded alternation. *Journal of Computer and System Sciences*, 21(2):218–235, 1980.

[Sav70]   W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[SSS10]   N. Schweikardt, T. Schwentick, and L. Segoufin. Database theory: Query languages. In *Algorithms and theory of computation*, pages 19.1–19.34. Chapman & Hall/CRC, 2010.

[Sze87]   R. Szelepcsényi. The moethod of focing for nondeterministic automata. *Bulletin of the EATCS*, 33:96–99, 1987.

[Var95]   M. Y. Vardi. On the complexity of bounded-variable queries. In *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 22-25, 1995, San Jose, California, USA*, pages 266–276, 1995.

[Ven91]   H. Venkateswaran. Properties that characterize LOGCFL. *Journal of Computer and System Sciences*, 43(2):380–404, 1991.

[Wig92]   Avi Wigderson. The complexity of graph connectivity. In *Mathematical Foundations of Computer Science 1992, 17th International Symposium, MFCS'92, Prague, Czechoslovakia, August 24-28, 1992, Proceedings*, pages 112–132, 1992.