Bounded variable logic, parameterized logarithmic space, and Savitch's theorem

Yijia Chen[1] and Moritz Müller[2]

[1] Department of Computer Science, Shanghai Jiaotong University, China
yijia.chen@cs.sjtu.edu.cn
[2] Kurt Gödel Research Center, University of Vienna, Austria
moritz.mueller@univie.ac.at

Abstract. We study the parameterized space complexity of model-checking first-order logic with a bounded number of variables. By restricting the number of the quantifier alternations we obtain problems complete for a natural hierarchy between parameterized logarithmic space and FPT. We call this hierarchy the *tree hierarchy*, provide a machine characterization, and link it to the recently introduced classes PATH and TREE. We show that the lowest class PATH collapses to parameterized logarithmic space only if Savitch's theorem can be improved. Finally, we settle the complexity with respect to the tree-hierarchy of finding short undirected paths and small undirected trees.

1 Introduction

The model-checking problem for first-order logic FO asks whether a given first-order sentence φ holds true in a given relational structure $\mathbf{A}$. The problem is PSPACE-complete in general and even its restriction to primitive positive sentences and two-element structures stays NP-hard. However, Vardi [24] showed in 1995 that the problem is solvable in polynomial time when restricted to a constant number of variables.

In database theory, a typical application of the model-checking problem, we are asked to evaluate a relatively short query φ against a large database $\mathbf{A}$. Thus, it has repeatedly been argued in the literature (e.g. [23]), that measuring in such situations the computational resources needed to solve the problem by functions depending only on the length of the input $(\varphi, \mathbf{A})$ is unsatisfactory. 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 of φ as parameter and asks for algorithms running in fpt time, that is, in time $f(|\varphi|) \cdot |\mathbf{A}|^{O(1)}$ for some computable function f. Sometimes, this relaxed tractability notion allows to translate (by an effective but often inefficient procedure) the formula into a form, for which the model-checking can be solved efficiently (see [14] for a survey). For example, algorithms exploiting Gaifman's locality theorem solve p-MC(FO) on structures of bounded local treewidth [12] in fpt time, and on bounded degree graphs even in parameterized logarithmic

space [10]. Parameterized logarithmic space, para-L, relaxes logarithmic space in much the same way as FPT relaxes polynomial time [4, 10].

Note that Vardi's result mentioned above implies that p-MC(FO^s) can be solved in fpt time, where FO^s denotes the class of first-order sentences using at most s variables. The starting point of this paper is the question whether one can solve p-MC(FO^s) in parameterized logarithmic space. We show that both a negative as well as a positive answer would imply certain breakthrough results in classical complexity theory. We now describe our results in some more details.

A first guess could be that when s increases, so does the space complexity of p-MC(FO^s). But it turns out that there is a parameterized logarithmic space reduction from p-MC(FO^s) to p-MC(FO^2) (implicit in Theorem 8 below). On the other hand, one can naturally stratify p-MC(FO^s) into subproblems p-MC(Σ_1^s), p-MC(Σ_2^s), ..., according to the number of quantifier alternations allowed in the input sentences. It leads to a hierarchy of classes TREE[t] consisting of the problems reducible to p-MC(Σ_t^s).

The lowest TREE[1] coincides with the class TREE introduced in [5]. The class TREE stems from the complexity classification of homomorphism problems under parameterized logarithmic space reductions, which refines Grohe's famous characterization of those homomorphism problems that are in FPT [13]. As shown in [5], they are either in para-L, or PATH-complete, or TREE-complete. The class PATH here had already been introduced by Elberfeld et al. [9].

All mentioned classes line up in the *tree hierarchy:*

$$\text{para-L} \subseteq \text{PATH} \subseteq \text{TREE}[1] \subseteq \text{TREE}[2] \subseteq \cdots \subseteq \text{TREE}[*] \subseteq \text{FPT}, \quad (1)$$

with p-MC(FO^s) being complete for TREE[$*$].

The classes PATH and TREE deserve some special interest. They can be viewed as parameterized analogues of NL and LOGCFL (cf. [25]) respectively, and capture the complexity of some parameterized problems of central importance. For example,

p-DiPath
Instance: A directed graph $\mathbf{G}$ and $k \in \mathbb{N}$.
Parameter: k.
Problem: Is there a directed path of length k in $\mathbf{G}$?

is complete for PATH [9, 5], and here we show (Proposition 17) that

p-DiTree
Instance: A directed graph $\mathbf{G}$ and a directed tree $\mathbf{T}$.
Parameter: $\lvert\mathbf{T}\rvert$.
Problem: Is there an embedding of $\mathbf{T}$ into $\mathbf{G}$?

is complete for TREE. We always assume that paths are simple, i.e. without repeated vertices. And by a *directed tree* we mean a directed graph obtained from a tree by directing all edges away from the root.

A negative answer to our question whether p-MC(FO^s) $\in$ para-L is equivalent to para-L $\neq$ TREE[$*$] and, in particular, implies L $\neq$ P.[3] In contrast, a positive answer would imply para-L = PATH, a hypothesis we study in some detail here. Recall that Savitch's seminal result from 1969 can be equivalently stated as NL $\subseteq$ DSPACE($\log^2 n$). In Lipton's words [18] "one of the biggest embarrassments of complexity theory [...] is the fact that Savitch's theorem has not been improved [...]. Nor has anyone proved that it is tight." Hemaspaandra et al. [16, Corollary 2.8] showed that Savitch's theorem could be improved if there were problems of sublogarithmic density $o(\log n)$ and Turing hard for NL. We refer to [20] for more on this problem. Here we show:

Theorem 1. *If* para-L = PATH*, then* NL $\subseteq$ DSPACE$\big(o(\log^2 n)\big)$.

The hypothesis para-L $\neq$ PATH is hence implied by the hypothesis that Savitch's Theorem is optimal, and in turn implies L $\neq$ NL (see the discussion before Proposition 6).

Finally, we settle the complexity of two more problems with respect to the tree-hierarchy. First we show that the undirected version

p-PATH
Instance: An (undirected) graph $\mathbf{G}$ and $k \in \mathbb{N}$.
Parameter: k.
Problem: Is there a path of length k in $\mathbf{G}$?

of p-DIPATH is in para-L. To the best of our knowledge this has not been known before despite the considerable attention p-PATH has gained in parameterized complexity theory (e.g. [6, 1, 7]). It answers a question of [5]. Second, and in contrast to the just mentioned result, we prove that the undirected version

p-TREE
Instance: An (undirected) graph $\mathbf{G}$ and a tree $\mathbf{T}$.
Parameter: $\lvert\mathbf{T}\rvert$.
Problem: Is there an embedding of $\mathbf{T}$ into $\mathbf{G}$?

of p-DITREE stays TREE-complete.

2 Preliminaries

Structures and logic. A *vocabulary* τ is a finite set of relation, function and constant symbols. Relation and function symbols have an associated *arity*, a positive natural number. A *τ-term* is a variable, a constant or of the form $f(t_1, \cdots, t_r)$ where f is an r-ary function symbol and $t_1, \ldots, t_r$ are again τ-terms. A *τ-atom* has the form $t = t'$ or $R(t_1, \ldots, t_r)$ where R is an r-ary relation symbol and $t, t', t_1, \ldots, t_r$ are τ-terms. *τ-formulas* are built from atoms by means of $\wedge, \vee, \neg$ and existential and universal quantification $\exists x, \forall x$. The vocabulary τ is called

[3] In fact, general results from [10] imply that the hypotheses para-L $\neq$ FPT, TREE[$*$] $\neq$ FPT and L $\neq$ P are all equivalent.

relational if it contains only relation symbols. A *(finite) τ-structure* $\mathbf{A}$ consists in 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* $f^{\mathbf{A}} : A^r \to A$ and for each constant symbol $c \in \tau$ an *interpretation* $c^{\mathbf{A}} \in A$. We view *digraphs* as $\{E\}$-structures $\mathbf{G}$ for a binary relation symbol E such that $E^{\mathbf{G}}$ is irreflexive. A *graph* is a digraph $\mathbf{G}$ with symmetric $E^{\mathbf{G}}$. If $\mathbf{G}$ is a (di)graph, we refer to elements of G as *vertices* and to elements of $E^{\mathbf{G}}$ as *(directed) edges*.

Let τ be a relational vocabulary and $\mathbf{A}$, $\mathbf{B}$ two τ-structures. A *homomorphism from* $\mathbf{A}$ *to* $\mathbf{B}$ is a function $h : A \to B$ such that for every r-ary $R \in \tau$ we have $\big(h(a_1), \ldots, h(a_r)\big) \in R^{\mathbf{B}}$ whenever $(a_1, \ldots, a_r) \in R^{\mathbf{A}}$. We understand that there do not exist homomorphisms between structures interpreting different (relational) vocabularies. As has become usual in our setting, we understand that an *embedding* is an injective homomorphism.

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. 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 $c : \{0,1\}^* \to \mathbb{N}$, the machine is said to *use* c *many nondeterministic (co-nondeterministic) bits* if for every $x \in \{0,1\}^*$ every run 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 logarithmic space computable *parameterization* $\kappa : \{0,1\}^* \to \mathbb{N}$, mapping any instance $x \in \{0,1\}^*$ to its *parameter* $\kappa(x) \in \mathbb{N}$. For a class $\mathcal{A}$ of structures we consider the parameterized *homomorphism problem*

p-Hom($\mathcal{A}$)
Instance: A structure $\mathbf{A} \in \mathcal{A}$ and a structure $\mathbf{B}$.
Parameter: $\lvert\mathbf{A}\rvert$.
Problem: Is there a homomorphism from $\mathbf{A}$ to $\mathbf{B}$?

Here, $|\mathbf{A}|$ denotes the size of a reasonable encoding of $\mathbf{A}$. Similarly, the parameterized *embedding problem* p-Emb($\mathcal{A}$) asks for an embedding instead of a homomorphism.

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 function $f : \mathbb{N} \to \mathbb{N}$. 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* $f(\kappa(x)) + O(\log |x|)$ for some computable $f : \mathbb{N} \to \mathbb{N}$. 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 $f : \mathbb{N} \to \mathbb{N}$, 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 ith bit b?

PATH and TREE. The class PATH (resp. TREE) contains those parameterized problems (Q, κ) such that Q is accepted by a nondeterministic Turing machine $\mathbb{A}$ which runs in parameterized logarithmic space, and for some computable function $f : \mathbb{N} \to \mathbb{N}$ uses $f(\kappa(x)) \cdot \log |x|$ many nondeterministic bits (and additionally $f(\kappa(x))$ many co-nondeterministic bits).

The class PATH has been discovered by Elberfeld et al. [9]. It captures the complexity of the fundamental problem:

p-REACHABILITY
Instance: A directed graph $\mathbf{G}$, two vertices $s, t \in G$, and $k \in \mathbb{N}$.
Parameter: k.
Problem: Is there a (directed) path of length k from s to t in $\mathbf{G}$?

Theorem 2 ([9, 5]). p-REACHABILITY *is* PATH*-complete (under pl-reductions).*

The class TREE has been introduced in [5] for the purpose of a classification of the complexities of homomorphism problems up to pl-reductions:

Theorem 3 ([15, 13, 5]). *Let $\mathcal{A}$ be a decidable class of relational structures of bounded arity. Then*

1. *if the cores of $\mathcal{A}$ have bounded tree-depth, then p-HOM($\mathcal{A}$) $\in$ para-L;*
2. *if the cores of $\mathcal{A}$ have unbounded tree-depth but bounded pathwidth, then p-HOM($\mathcal{A}$) is* PATH*-complete;*
3. *if the cores of $\mathcal{A}$ have unbounded pathwidth but bounded treewidth, then p-HOM($\mathcal{A}$) is* TREE*-complete;*
4. *if the cores of $\mathcal{A}$ have unbounded treewidth, then p-HOM($\mathcal{A}$) is not in* FPT *unless* W[1] = FPT.

Here, *bounded arity* means that there is a constant bounding the arities of symbols interpreted in structures from $\mathcal{A}$.[4] Understanding in a similar way the complexities of the embedding problems p-EMB($\mathcal{A}$) is wide open (see e.g. [11, page 355]). We know:

Theorem 4 ([5]). *For $\mathcal{A}$ as in Theorem 3 we have p-EMB($\mathcal{A}$) $\in$ para-L in case (1), p-EMB($\mathcal{A}$) $\in$ PATH in case (2), and p-EMB($\mathcal{A}$) $\in$ TREE in case (3).*

Note p-PATH and p-TREE are roughly the same as p-EMB($\mathcal{P}$) and p-EMB($\mathcal{T}$), where $\mathcal{P}$ and $\mathcal{T}$ denote the classes of paths and trees, respectively. The complexities of these important problems are left open by Theorems 3 and 4.

[4] We do not recall the notion of core nor the width notions here because we do not need them.

3 Model-checking bounded variable first-order logic

The tree hierarchy. Following [5] 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 they use *jumps* explained as follows. Recall our Turing machines have an input tape. During a computation on an input x of length $n := |x|$ 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 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.

Observe that of 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 $[n] := \{1, \dots, n\}$. 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 some of its successors, and that contains a universal configuration if it contains all of its successors.

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 in t blocks, the first consisting in existential configurations, the second in universal configurations, and so on. The machine $\mathbb{A}$ is *t-alternating* if for every input $x \in \{0,1\}^*$ every run of $\mathbb{A}$ on x is t-alternating.

Let $f : \{0,1\}^* \to \mathbb{N}$. The machine $\mathbb{A}$ *uses f jumps (bits)* if for every input $x \in \{0,1\}^*$ 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 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, we use regular expressions to denote languages over $\{j_\exists, j_\forall, c_\exists, c_\forall\}$.

Definition 5. 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, and uses $f \circ \kappa$ jumps and $f \circ \kappa$ bits. Furthermore, if $\mathbb{A}$ 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.

It is straightforward to verify PATH $\subseteq$ TREE = TREE[1] (cf. [5, Lemmas 4.5, 5.4]). Obviously, para-L $\subseteq$ PATH $\subseteq$ para-NL, and all classes are equal if L = NL (see [10]). Conversely, Elberfeld et al. [9] observed that L = NL if PATH = para-NL. In fact, using general results from [10] one can show:

Proposition 6. *1.* para-NL $\subseteq$ TREE[$*$] *if and only if* NL = L.

2. FPT $\subseteq$ TREE[$*$] *if and only if* P = L.

We shall need the following technical lemma.

Lemma 7 (Normalization). *Let $t \geq 1$. A parameterized problem (Q, κ) is in* 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 (with respect to κ) and such that for all $x \in \{0,1\}^*$ every run of $\mathbb{A}$ on x has nondeterminism type:*

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

Model-checking. 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 and existential (resp. universal) quantification. We use Σ_t^s and Π_t^s to denote $\mathrm{FO}^s \cap \Sigma_t$ and $\mathrm{FO}^s \cap \Pi_t$ respectively.

For a class of formulas Φ we consider the parameterized problem:

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

It is well known [24] that for all $s \in \mathbb{N}$ the problem p-MC(FO^s) is in FPT, indeed, the underlying classical problem is in P.

Theorem 8. *Let $t \geq 1$ and $s \geq 2$. Then p-MC(Σ_t^s) is* TREE[t]*-complete.*

Proof. The containment p-MC(Σ_t^s) $\in$ TREE[t] is straightforward. To show that p-MC(Σ_t^2) is hard for TREE[t], let $(Q, \kappa) \in$ 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 that 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, the number $f(\kappa(x))$ and hence 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 that 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 length $\leq m$ computation of $\mathbb{B}$ leading from c to c' that neither passes through a nondeterministic configuration nor through a configuration of space $> s$. We assume $\mathbb{B}$ 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 (length $O(s)$ binary codes of) nondeterministic space s configurations of $\mathbb{B}$ on x. The structure $\mathbf{A}$ 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 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 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 symbol F is interpreted by the set of those $c \in A$ such that

- $c \in C_\exists^{\mathbf{A}} \cup J_\exists^{\mathbf{A}}$ 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.

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 $\{d \in A \mid (c,d) \in E^{\mathbf{A}}\} \neq \emptyset$ we have:

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

It is easy to check that $\mathbf{A}$ is implicitly pl-computable from x. 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 $> s$ 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 $|w|$. If $|w| = 1$, define $\varphi_w(x) := Fx$. For $|w| \geq 1$ define:

$$\begin{aligned}
\varphi_{c_\forall w}(x) &:= C_\forall x \wedge \big(\varphi_w(s_0(x)) \wedge \varphi_w(s_1(x))\big),\\
\varphi_{c_\exists w}(x) &:= C_\exists x \wedge \big(\varphi_w(s_0(x)) \vee \varphi_w(s_1(x))\big),\\
\varphi_{j_\exists w}(x) &:= J_\exists x \wedge \exists y\big(E(x,y) \wedge \exists x(x = y \wedge \varphi_w(x))\big),\\
\varphi_{j_\forall w}(x) &:= J_\forall x \wedge \forall y\big(\neg E(x,y) \vee \forall x(\neg x = y \vee \varphi_w(x))\big).
\end{aligned}$$

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; then (recall the definition of an accepting configuration from page 6)

$$c \text{ is accepting} \iff \mathbf{A} \models \varphi_w(c). \tag{3}$$

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

By the Normalization Lemma 7 we can assume that all runs of $\mathbb{B}$ on x have nondeterminism type w of the form (2). For such a w we observe that $\varphi_w(x)$ is in Σ_t^2 and 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 $\varphi'_w(x)$. Note that $\varphi'_w(x) \in \Sigma_t^2$.

To define $\mathbf{A}'$ we expand $\mathbf{A}$ setting $E_b^{\mathbf{A}'} := \{(c,d) \mid (s_b^{\mathbf{A}}(c), d) \in E^{\mathbf{A}}\}$, $J_{\exists b}^{\mathbf{A}'} := \{c \mid s_b^{\mathbf{A}}(c) \in J_\exists^{\mathbf{A}}\}$, and $J_{\forall b}^{\mathbf{A}'} := \{c \mid s_b^{\mathbf{A}}(c) \in J_\forall^{\mathbf{A}}\}$. Then we have for all $c \in A$:

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

As the assumption of (3) 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

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

Then $x \mapsto \left(\exists x(Sx \wedge \varphi'_w(x)), \mathbf{A}'\right)$ is a reduction as desired. □

Now, the following are derived by standard means.

Corollary 9. *Let $t' > t \geq 1$. If* Tree[t] *is closed under complementation, then* Tree[t'] = Tree[t].

Corollary 10. *Let $s \geq 2$. Then p-*MC(FO^s) *is* Tree[$*$]*-complete. In particular,* TREE[$*$] $\subseteq$ FPT.

Remark 11. It is not known whether PATH or TREE are closed under complementation (cf. [5]). Their classical counterparts NL and LOGCFL are (cf. [2]), but both proofs break under the severe restrictions on nondeterminism in the parameterized setting.

4 PATH and classical complexity theory

Savitch's Theorem [22] is a milestone result linking nondeterministic space to deterministic space. It states that the problem

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

is in DSPACE($\log^2 n$). It is a long-standing open problem whether this can be improved. We prove the following stronger version of Theorem 1:

Theorem 12. *Assume whether* $(\mathbf{G}, s, t, k) \in p$-Reachability *can be decided in deterministic space $f(k) + O(\log |G|)$ for a function $f : \mathbb{N} \to \mathbb{N}$ (which is not necessarily computable). Then* Reachability $\in$ DSPACE$\big(o(\log^2 n)\big)$.

Proof. (Sketch) Let $\mathbb{A}$ be an algorithm deciding whether $(\mathbf{G}, s, t, k) \in p$-Reachability in space $f(k) + O(\log |G|)$. We can assume that $f(k) \geq k$ for every $k \in \mathbb{N}$. Then let $\iota : \mathbb{N} \to \mathbb{N}$ be nondecreasing and unbounded such that

$$f(\iota(n)) \leq \log n, \text{ and hence } \iota(n) \leq \log n \tag{4}$$

for all $n \in \mathbb{N}$. Note that we might not know how to compute $\iota(n)$. Now let $\mathbf{G} = (G, E^{\mathbf{G}})$ 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* $\ell := \ell(k)$ such that

$$k^\ell \geq n - 1, \text{ and hence } \ell = O(\log n / \log k). \tag{5}$$

We define a sequence of directed graphs $(\mathbf{G}_i^k)_{i\le\ell}$ with self-loops. Each $\mathbf{G}_i^k$ has vertices $G_i^k := G$ and a directed edge $(u,v) \in E^{\mathbf{G}_i^k}$ 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 $E^{\mathbf{G}}$; and by (5) there is a path from s to t in $\mathbf{G}$ if and only if there is an edge from s to t in $\mathbf{G}_\ell^k$. 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 $\mathbf{G}_i^k$ if and only if there is a path from u to v in $\mathbf{G}_{i-1}^k$ of length at most k. The following recursive algorithm $\mathbb{C}$ decides, given a directed graph $\mathbf{G}$, $k, i \in \mathbb{N}$, and $u, v \in G$, whether $(u,v) \in E^{\mathbf{G}_i^k}$.

1. **if** $i = 0$ **then** output whether $\big(u = v$ or $(u,v) \in E^{\mathbf{G}}\big)$ and return
2. simulate $\mathbb{A}$ on $\big(\mathbf{G}_{i-1}^k, u, v, k\big)$
3. **if** $\mathbb{A}$ queries "$(u',v') \in E^{\mathbf{G}_{i-1}^k}$?" **then** call $\mathbb{C}(\mathbf{G}, k, i-1, u', v')$.

For every $k \ge 2$ let $\mathbb{C}^k$ be the algorithm which, given a directed graph $\mathbf{G}$ and $s, t \in G$, first computes $\ell = \ell(k)$ as in (5) 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 $\mathbf{G}$. 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

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

These are space $f(k) + O(\log n)$ configurations, so by (5) $\mathbb{C}^k$ runs in space

$$O\Big(\log k + \log n + \ell \cdot \big(f(k) + \log n\big)\Big) = O\Big(\log k + \frac{f(k)\cdot\log n + \log^2 n}{\log k}\Big).$$

By (4) this is $o(\log^2 n)$ for $k := \iota(n)$. We would thus be done if we could compute $\iota(n)$, say, in space $O(\log n)$. In particular, this can be ensured under the hypothesis para-L = PATH of Theorem 1 which allows to choose f space-constructible. The general case needs some additional efforts. It can be handled using the strategy underlying Levin's optimal inverters [17, 8], namely to simulate all $\mathbb{C}^2, \mathbb{C}^3, \dots$ in a diagonal fashion. □

The trivial brute-force algorithm (cf. [5, Lemma 3.11]) decides p-MC(Σ_1^2) $\big($indeed, the whole p-MC(FO)$\big)$ in space $O\big(|\varphi|^2 \cdot \log|\mathbf{A}|\big)$. Assuming the optimality of Savitch's Theorem, this is space-optimal in the following sense:

Corollary 13. *If* REACHABILITY $\notin$ DSPACE($o(\log^2 n)$)*, then whether* $(\varphi, \mathbf{A}) \in p$-MC($\Sigma_1^2$) *cannot be decided in deterministic space* $o\big(f(|\varphi|)\cdot\log|\mathbf{A}|\big)$ *for any* f.

We close this section by characterizing the collapse of PATH to para-L similarly as analogous characterizations of W[P] = FPT [3], or BPFPT = FPT [19].

Definition 14. 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.

Theorem 15. para-L = PATH *if and only if there exists a space-constructible function* $c(n) = \omega(\log(n))$ *such that* NL[c] = L.

5 Embedding Undirected Paths and Trees

As mentioned in the Introduction it is known that (see [5, Theorem 4.7]):

Proposition 16. *p-DiPath is PATH-complete.*

A straightforward but somewhat tedious argument shows:

Proposition 17. *p-DiTree is TREE-complete.*

The following two results determine the complexities of the undirected versions of these two problems. Somewhat surprisingly, the complexity of the former drops to para-L while the latter stays TREE-complete:

Theorem 18. *p-Path $\in$ para-L.*

Theorem 19. *p-Tree is TREE-complete.*

Theorem 18 answers a question posed in [5, Section 7]. Its proof is based on the well-known color-coding technique. Specifically, we shall use the following lemma from [11, page 349]:

Lemma 20. *For every sufficiently large $n \in \mathbb{N}$, it holds that for all $k \leq n$ and for every k-element subset X of $[n]$, there exists a prime $p < k^2 \cdot \log n$ and $q < p$ such that the function $h_{p,q} : [n] \to \{0, \ldots, k^2 - 1\}$ given by $h_{p,q}(m) := (q \cdot m \bmod p) \bmod k^2$ is injective on X.*

Proof of Theorem 18. Let $\mathbf{G} = \big([n], E^{\mathbf{G}}\big)$ be a graph and $0 < k < n$. Assume n is large enough for Lemma 20 to apply. Using its notation we set

$$F := \Big\{ g \circ h_{p,q} \;\Big|\; g : \{0, \ldots, (k+1)^2 - 1\} \to [k+1] \text{ and } q < p < (k+1)^2 \log n \Big\}.$$

For $f \in F$ let $\mathbf{G}(f)$ be the graph obtained from $\mathbf{G}$ by deleting all edges $(u, v) \in E^{\mathbf{G}}$ with $|f(u) - f(v)| \neq 1$. By Lemma 20 one readily verifies that $\mathbf{G}$ contains a path of length k if and only if there are $f \in F$ and $u, v \in [n]$ such that $f(u) = 1$, $f(v) = k + 1$, and there is a path from u to v in $\mathbf{G}(f)$.

To decide whether $(\mathbf{G}, k) \in p$-Path we cycle through all tuples (g, p, q, u, v) with $g : \{0, \ldots, (k+1)^2 - 1\} \to [k+1], q < p < (k+1)^2 \log n$, and $u, v \in [n]$, and test whether $g(h_{p,q}(u)) = 1$, $g(h_{p,q}(v)) = k + 1$, and there is a path from u to v in $\mathbf{G}(g \circ h_{p,q})$. For every such test we simulate Reingold's logarithmic space algorithm [21] for Reachability on (undirected) graphs. The simulation relies on the fact that $\mathbf{G}(g \circ h_{p,q})$ is implicitly pl-computable from (g, p, q) and $\mathbf{G}$. $\square$

Acknowledgements

We are indepted to Hubie Chen who proposed the classes TREE[t] for study. We thank him and Jörg Flum for their comments on earlier drafts of this paper. Further we want to thank the referees for their comments. The research of the first author is partially supported by National Nature Science Foundation of China via Projects 61373029 and 61033002. The second author has been supported by the FWF (Austrian Science Fund) Project P 24654 N25.

References

1. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. of Comp. and Syst. Sciences*, 75(8):423–434, 2009.
2. A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo, and M. Tompa. Two applications of inductive counting for complementation problems. *SIAM J. on Computing*, 18(3):559–578, 1989.
3. L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. On the structure of parameterized problems in NP. *Information and Computation*, 123:38–49, 1995.
4. L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, 84(1):119–138, 1997.
5. H. Chen and M. Müller. The fine classification of conjunctive queries and parameterized logarithmic space complexity. In *Proc. of PODS'13*, 309–320, full version arXiv:1306.5424 [cs.CC], 2013
6. J. Chen, J. Kneis, S. Lu, D. Mölle, S. Richter, P. Rossmanith, S.-H. Sze, and F. Zhang. Randomized divide-and-conquer: improved path, matching, and packing algorithms. *SIAM J. on Computing*, 38(6):2526–2547, 2009.
7. Y. Chen and J. Flum. On parameterized path and chordless path problems. In *Proc. of CCC'07*, 250–263, 2007.
8. Y. Chen and J. Flum. On optimal inverters. *Bull. Symb. Logic*, 2014. To appear.
9. M. Elberfeld, C. Stockhusen, and T. Tantau. On the space complexity of parameterized problems. In *Proc. of IPEC'12*, LNCS 7535, 206–217, 2012.
10. J. Flum and M. Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.
11. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
12. M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable structures. *J. of the ACM*, 48(6):1184–1206, 2001.
13. M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. of the ACM*, 54(1):1:1–1:24, 2007.
14. M. Grohe and S. Kreutzer. Methods for algorithmic meta theorems. In *Model Theoretic Meth. in Finite Combinatorics*, Cont. Math. 558, 181–206. AMS, 2011.
15. M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *Proc. of STOC'01*, 2001.
16. L. A. Hemaspaandra, M. Ogihara, S. Toda. Space-efficient recognition of sparse self-reducible languages. *Computational Complexity* 4:262–296, 1994.
17. L. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.
18. J. R. Lipton. *The P = NP Question and Gödel's Lost Letter*. Springer, 2010.
19. J.-A. Montoya and M. Müller. Parameterized random complexity. *Theory of Computing Systems*, 52(2):221–270, 2013.
20. A. Potechin. Bounds on monotone switching networks for directed connectivity. In *Proc. of FOCS'10*, 553–562, 2010.
21. O. Reingold. Undirected connectivity in log-space. *J. of the ACM*, 55(4), 2008.
22. W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. of Comp. and Syst. Sciences*, 4(2):177–192, 1970.
23. N. Schweikardt, T. Schwentick, and L. Segoufin. Database theory: Query languages. *Algorithms and theory of computation*, 19.1–19.34. Chapman & Hall/CRC, 2010.
24. M. Y. Vardi. On the complexity of bounded-variable queries. In *Proc. of PODS'95*, 266–276. ACM Press, 1995.
25. H. Venkateswaran. Properties that characterize LOGCFL. *J. of Comp. and Syst. Sciences*, 43(2):380–404, 1991.