Parameterised Counting Classes with Bounded Nondeterminism

Anselm Haak  
Leibniz Universität Hannover, Institut für Theoretische Informatik, Hannover, Germany
    haak@thi.uni-hannover.de

Arne Meier  
Leibniz Universität Hannover, Institut für Theoretische Informatik, Hannover, Germany
    meier@thi.uni-hannover.de

Om Prakash  
Department of Computer Science and Engineering, IIT Madras, Chennai, India
    op708543@gmail.com

Raghavendra Rao B. V.  
Department of Computer Science and Engineering, IIT Madras, Chennai, India
    bvrr@cse.iitm.ac.in

Abstract

Stockhusen and Tantau (IPEC 2013) introduced the operators $\text{para}_W$- and $\text{para}_\beta$- for parameterised space complexity classes by allowing bounded nondeterminism with read only and read-once access, respectively. Using these operators, they could characterise the complexity of many parameterisations of natural problems on graphs. In this article, we study the counting versions of the parameterised space bounded complexity classes introduced by Stockhusen and Tantau (IPEC 2013). We show that natural path counting problems in digraphs are complete for the newly introduced classes $\#\text{para}_W$-L and $\#\text{para}_\beta$-L. Finally, we introduce parameterised counting classes based on branching programs (BPs). We show that parameterised counting classes based on branching programs coincide with the corresponding parameterised space bounded counting classes.

1 Introduction

Parameterised complexity theory, introduced by Downey and Fellows [11], takes a multidimensional view on algorithmic complexity and has revolutionised the algorithmic world. In parameterised complexity theory, a problem comes with an associated parameter $k$ and the complexity analysis is now two dimensional, i.e., with respect to the input size as well as the parameter. The notion of fixed parameter tractability (FPT for short) is the proposed notion of efficient computation. A problem with parameter $k$ is FPT if there is a deterministic $f(k) \cdot n^{O(1)}$ time algorithm for computing it. The notion of intractability is captured by the classes XP and the $W$-hierarchy.

Since inception, the focus of parameterised complexity theory has been to develop efficient parameterised algorithms for NP-hard problems and to address structural aspects of the classes in the $W$-hierarchy and related complexity classes [14]. This lead to the development of machine based and logical characterisations of parameterised complexity classes (see the book of Downey and Fellows [14] for more details). While the structure of classes in
hierarchies such as the \( W \)- and \( A \)-hierarchy are well understood, a parameterised view of parallel and space bounded computation lacked attention.

Bannach, Stockhusen and Tantau \[3\] studied parameterised parallel algorithms for the first time. They used color coding techniques to obtain efficient parameterised parallel algorithms for several natural problems. Chen and Flum \[5\] obtained parameterised lower bounds for \( AC^0 \) by adapting circuit lower bound techniques to the parameterised setting.

Stockhusen and Tantau \[20\] studied parameterised space complexity classes. In fact, they have introduced parameterised analogues of deterministic and nondeterministic logarithmic space bounded classes. The machine based characterisations of \( WP \) and \( W[1] \), and the type of access to nondeterministic choices (multi-read or read-once) lead to four different variants of parameterised logspace, viz., \( \text{para}_W L \), \( \text{para}_\beta L \), \( \text{para}_W[^1] L \) and \( \text{para}_\beta_{\text{tail}} L \). Stockhusen and Tantau \[20\] obtained several natural complete problems such as parameterised variants of reachability for these classes.

Apart from the decision problems, counting problems have found a prominent place in complexity theory. Valiant \[22\] introduced the notion of counting complexity classes that capture natural counting problems such as counting the number of perfect matchings in a graph or counting the number of satisfying assignments in a CNF formula. Valiant’s theory of \#P-completeness lead to several structural insights into complexity classes around \( NP \) and interactive proof systems, and culminated in Toda’s theorem \[21\]. Introduced by Allender, Beals and Ogihara \[2\] the counting variants of space bounded complexity classes and their probabilistic counterparts have proven extremely fruitful. These classes characterise the complexity of natural problems such as testing if a determinant of an integer matrix is zero or not and computing the rank of a matrix. Moreover, the logspace analogue of the counting hierarchy is known to collapse \[17\].

Further down the complexity hierarchy, Caussinus et. al \[4\] introduced counting versions of \( NC^1 \) based on various characterisations of \( NC^1 \). Moreover, counting and gap variants of the class \( AC^0 \) were defined by Agrawal et al. \[1\]. The counting and probabilistic analogues of \( NC^1 \) exhibit properties similar to their logspace counterparts \[10\].

The parameterised theory of counting classes was pioneered by Flum and Grohe \[13\]. They showed that counting cycles of length \( k \) is complete for \#W[1]. Curticapean \[7\] further showed that counting matchings with \( k \) edges in a graph is also complete for \#W[1]. These results lead to several interesting completeness results and new techniques (see, e.g., the following papers of Curticapean \[8, 9\]).

Given the richness of counting variants of parallel complexity classes and the theory of parameterised counting problems, it is worthwhile exploring the parameterised versions of these classes. We define the counting variants of the parameterised space complexity classes introduced by Stockhusen and Tantau \[20\]. In particular, we define counting classes \#para_W L and \#para_\beta L and the variants based on tail-nondeterminism. We show that these classes are closed under addition and multiplication (Lemma \[12\]). Furthermore, we exhibit natural complete reachability problems for these classes (Theorem \[18\] and \[19\]).

Branching programs (BPs) are integral to the study of space-bounded and parallel complexity classes. Languages accepted by polynomial size log-space uniform branching programs characterise \( NL \). In fact, this result carries forward to the counting versions. Motivated by this, we consider parameterised counting languages based on deterministic (DBPs) and nondeterministic branching programs (BPs). We show that the class \#para_W L is equal to \#para_W DBP, i.e., the problem of counting the number of \( s-t \) paths in a deterministic branching program with \( k \)-bounded nondeterministic bits (Theorem \[13\]). We extend the notion of read-once access to nondeterministic bits to the case of branching programs using...
the notion of read-once certified BPs [16]. With our notion of read-once certified DBPs, we show that $\#_{\text{para}_s-L} = \#_{\text{para}_s-\text{DBP}}$. This characterisation also carries forward to the case of tail-nondeterministic variants of the newly introduced counting classes.

## 2 Preliminaries

We first define parameterised operators for space bounded complexity classes. The notations follow Stockhusen [19]. We consider three parameterised operators, viz., $\text{para}$-, $\text{para}_W$- and $\text{para}_s$-. The $\text{para}$- operator was introduced by Flum and Grohe [12] as a uniform way of obtaining parameterised versions of classical complexity classes.

► **Definition 1.** Let $\mathcal{C}$ be any complexity class. Then $\text{para-}\mathcal{C}$ is the class of all parameterised problems $P \subseteq \Sigma^* \times \mathbb{N}$ for some alphabet $\Sigma$ such that there is an alphabet $\Delta$, a computable function $\pi: \mathbb{N} \rightarrow \Delta^*$ and a language $L \in \mathcal{C}$ with $L \subseteq \Sigma^* \times \Delta^*$ such that for all $x \in \Sigma^*$, $k \in \mathbb{N}$: $(x, k) \in P \Leftrightarrow (x, \pi(k)) \in L$.

It may be noted that $\text{para-}\mathcal{P} = \text{FPT}$. The principal notion of intractability for parameterised problems is captured by the classes $\text{XP}$ and the $\text{W}$-hierarchy [13]. Though the $\text{W}$-hierarchy was defined based on the weighted satisfiability of formulae based on wefts, Flum and Grohe [12] obtained central classes in this context through bounded nondeterminism. Stockhusen and Tantau [20, 19] considered bounded nondeterminism in the case of space bounded and circuit based parallel complexity classes.

Since we study relatively weak classes in parameterised complexity, we need the concept of parameterised logspace reductions defined as follows.

► **Definition 2.** Let $P \subseteq \Sigma^* \times \mathbb{N}, Q \subseteq \Delta^* \times \mathbb{N}$ be parameterised problems. $P$ is para-logspace (many-one) reducible to $Q$, $P \leq_{\text{mlog}} Q$, if there is a function $r: \Sigma^* \times \mathbb{N} \rightarrow \Delta^* \times \mathbb{N}$ such that the following is true.

1. For all $(x, k) \in \Sigma^* \times \mathbb{N}$: $(x, k) \in P \Leftrightarrow r(x, k) \in Q$.
2. The function $r$ is computable by a para-logspace algorithm, that is, there is a computable function $f: \mathbb{N} \rightarrow \mathbb{N}$ such that $r$ is computable in space $O(f(k) + \log |x|)$.
3. There is a computable function $g: \mathbb{N} \rightarrow \mathbb{N}$ such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$ and $r(x, k) = (x', k')$, we have that $k' \leq g(k)$.

Using the concept of oracle machines (see, e.g., the textbook of Sipser [18]), we say, that a language $A$ is Turing-reducible in para-logspace to a language $B$, in symbols $A \leq_{T}^{\text{plog}} B$, if $A \in \text{para-}\mathcal{L}^P$.

### 2.1 Logspace-Bounded Complexity Classes and Branching Programs

Now, we give necessary definitions of the computational models and parameterised complexity classes relevant for the article.

**Turing Machines.** Let $\mathcal{C}$ be a complexity class based on Turing machines with space bound $s(n)$ and time bound $t(n)$. A $\mathcal{C}$-machine is a Turing machine that is $s$ space bounded and $t$ time bounded. A $\text{para-}\mathcal{C}$-machine is a Turing machine that is $s(|x| + f(k))$ space bounded and $t(|x| + f(k))$ time bounded for any input $(x, k)$, where $f$ is a computable function.

Nondeterministic Turing machines are a generalisation of Turing machines where multiple transitions from a given configurations are allowed. This can be formalised by allowing the transition to be a relation rather than a function. Sometimes, it is helpful to view nondeterministic Turing machines as deterministic Turing machines with an additional
Parameterised Counting Classes with Bounded Nondeterminism

nondeterministic choice tape. Let $M$ be a deterministic Turing machine with a choice tape. Let $x,y$ be strings over the underlying alphabet $\Sigma$. The language accepted by $M$ can be defined as:

$$L(M) = \{ x \in \Sigma^* \mid \exists y \in \Sigma^* : M \text{ accepts } x \text{ when the choice tape is initialised with } y \}.$$ 

In the above, the machine $M$ has two-way access to the choice tape.

Flum, Chen and Grohe \cite{Flum2011} obtained a characterisation of the $W$-hierarchy using the following notion of $k$-bounded nondeterministic Turing machines.

\begin{definition}[$k$-bounded Turing machines] A deterministic Turing machine $M$ with a choice tape, working on inputs of the form $(x,k)$ with $x \in \Sigma^*, k \in \mathbb{N}$ for some alphabet $\Sigma$, is said to be $k$-bounded if it reads at most $f(k) \cdot \log |x|$ bits from the choice tape, where $f$ is a computable function.
\end{definition}

\begin{proposition}[$\text{W}^p$] $\text{W}[P]$ is the set of all parameterised problems $Q$ that can be accepted by $k$-bounded FPT-machines with a choice tape.
\end{proposition}

Stockhausen and Tantau \cite{Stockhausen2010} specialised the above definition of $k$-bounded TMs to include space bounded computation.

\begin{definition}[$\text{para}_k\text{-C}$] For a complexity class $\mathcal{C}$ based on Turing machines, $\text{para}_{k}\text{-C}$ denotes the class of all parameterised problems $Q$ that can be accepted by a $k$-bounded $\text{para}\text{-C}$-machine with a choice tape.
\end{definition}

For example, $\text{para}_{\text{NL}}\text{-L}$ denotes the parameterised version of NL with $k$-bounded nondeterminism. Naturally, one can also restrict this model by only giving one-way access to the nondeterministic tape. This leads to the definition of the $\text{para}_{\beta}$- operator on complexity classes.

\begin{definition}[$\text{para}_{\beta}\text{-C}$] For a complexity class $\mathcal{C}$ based on Turing machines, $\text{para}_{\beta}\text{-C}$ denotes the class of all parameterised problems $Q$ that can be accepted by a $k$-bounded $\text{para}\text{-C}$-machine with a choice tape with one-way read access to the choice tape.
\end{definition}

The machine characterisation of $\text{W}[1]$ requires the notion of tail-nondeterminism \cite{Flum2011}. A $k$-bounded machine is said to be tail-nondeterministic if the nondeterministic bits are read in the last $g(k) \cdot \log n$ steps of the computation, for some computable $g$. The tail-nondeterministic version of $\text{para}_{\text{NL}}\text{-C}$ and $\text{para}_{\beta}\text{-C}$ are denoted, respectively, by $\text{para}_{\text{NL}}\text{-tail}\text{-C}$ and $\text{para}_{\beta}\text{-tail-C}$.

\textbf{Branching programs} \quad Recall that a branching program $P$ is a layered directed acyclic graph with a source node $s$ and a sink node $t$. The vertices of the branching program are labelled by input variables in $\{x_1, \ldots, x_n\}$ and edges are labelled by $0/1$. An input $a_1 \cdots a_n \in \{0,1\}^n$ is accepted by $P$ if there is a directed $s$ to $t$ path (short: $s$-$t$-path) $\rho$ that is consistent with $a$, that is, for each edge $(u,v)$ in $\rho$, $\text{label}(u,v) = a_i$ where $\text{label}(u) = x_i$. The branching program $P$ is said to be deterministic if every vertex except the sink has exactly two out-going edges, one labelled by $0$ and the other by $1$. The size of the program $P$ is the number of vertices in it, the length is the length of a longest path starting from $s$. If $P$ has length $\ell$, then we assume that the vertices of $P$ are partitioned into layers $L_0 \cup L_1 \cup \ldots \cup L_\ell$, where $L_0$ contains the source and $L_\ell$ contains the sink. By layer $i$, we mean the set of vertices in $L_i$.

\begin{remark} Throughout this article, we assume branching programs are logspace uniform. In the case of parameterised classes, $\text{para-L}$ uniformity is assumed. For more details about notions of uniformity, the reader is referred to the textbook of Vollmer \cite{Vollmer2009}.
\end{remark}
Let $\text{BP}$ denote the set of all languages accepted by polynomial size, logspace uniform families of branching programs. Let $\text{DBP}$ denote the set of all languages accepted by polynomial size, logspace uniform families of deterministic branching programs.

If $\mathcal{C}$ is a complexity class based on families of branching programs, then a family of $\mathcal{C}$-BPs is a family of branching programs that respects the resource bounds of $\mathcal{C}$. As we will consider branching programs that cope with parameterised problems, we need to incorporate the parameter accordingly. For that reason, in this context families of branching programs are of the form $P := (P_{n,m})_{n,m \geq 0}$. The language accepted by $P$ is the set of all inputs $(x,k) \in \Sigma^* \times \mathbb{N}$ such that $P_{|x|,|k|}$ accepts $(x,k)$. Let $\mathcal{C}$ be a complexity class based on families of branching programs of size $s(n)$. A family of $\text{para-}\mathcal{C}$-BPs is a family of branching programs $(P_{n,m})_{n,m \geq 0}$ of size $s(n + f(m))$.

Note that, while the operator $\text{para-}$ is defined for arbitrary complexity classes, the operators $\text{para}_{\text{W}-}$ and $\text{para}_{\text{D}-}$ are only defined with respect to Turing machine based classes. These operators can be generalised to complexity classes based on branching programs by extending the notion of $k$-bounded nondeterminism to this context, though. A family of $\text{branching programs with nondeterministic input}$ has nondeterministic choices as additional input to its branching programs. Let $P := (P_{n,m})_{n,m \geq 0}$ be such a family and $\ell(n,m)$ be the number of nondeterministic input bits in $P_{n,m}$. We say that $P$ accepts an input $(x,k) \in \{0,1\}^* \times \mathbb{N}$, if there is a $y \in \{0,1\}^{\ell(|x|,|k|)}$ such that $P_{|x|,|k|}$ accepts $(x,k,y)$. Also, denote by $\#\text{acc}P_{|x|,|k|}(x,k)$ the number of $y \in \{0,1\}^{\ell(|x|,|k|)}$ such that $P$ accepts $(x,k,y)$. Furthermore, $P$ is said to be $k$-bounded if there exists a computable function $f$ such that for all $n,m \geq 0$, the number $\ell(n,m) \leq f(m) \cdot \log n$ and $P_{n,m}$ has size $f(m) \cdot n^{O(1)}$.

**Definition 8.** Let $\mathcal{C}$ be any branching program based complexity class. Then, $\text{para}_{\text{W}-}\mathcal{C}$ is the class of all parameterised languages computable by $k$-bounded families of $\text{para-}\mathcal{C}$-BPs.

We are interested in the case when $\mathcal{C}$ is either $\text{BP}$ or $\text{DBP}$.

We will now introduce a notion of read-once access to nondeterministic bits for the above classes. Let $P(x,y)$ be a branching program with two inputs $x = x_1 \cdots x_n$ and $y = y_1 \cdots y_m$. Here, $y$ is the nondeterministic input. The program $P$ is said to be deterministic, if for every node $u$ in $P$, $u$ has exactly two outgoing edges, one labelled by 0 and the other by 1. We say that $P$ is read-once certified if there are layers $i_0 < i_1 < i_2 < \cdots < i_m$ in the underlying graph of $P$ such that the variable $y_j$ occurs as a label only in layer numbers $\eta$ such that $i_{\eta-1} \leq \eta \leq i_{\eta}$.

**Definition 9.** Let $\mathcal{C}$ be any branching program based complexity class. Then, $\text{para}_{\text{W}-}\mathcal{C}$ is the class of all parameterised languages computable by $k$-bounded families of $\text{para-}\mathcal{C}$-BPs that are read-once certified.

The above definition can also be specialised to the case of tail-nondeterministic computation yielding the operators $\text{para}_{\text{W}\{1\}-}$ and $\text{para}_{\text{D-tail}}$.

### 3 Parameterised Counting Classes

In this section, we define the counting counterparts based on the parameterised classes defined using bounded nondeterminism. As in the case of Boolean complexity classes, we assume that all branching programs considered throughout this article are $\text{para-}\mathcal{L}$ uniform. A parameterised function is a function $f : \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$, where the value of the function on input $x \in \{0,1\}^*, k \in \mathbb{N}$ is given by $f(x,k)$. Here, $k$ is the parameter.
Parameterised Counting Classes with Bounded Nondeterminism

Definition 10. Let \( C \) be a complexity class. A parameterised function \( f \) is said to be in \( \#paraw-C \) if there is a \( k \)-bounded \( paraw \)-machine \( M \) such that for all inputs \((x,k)\), the number of accepting paths of \( M \) on input \((x,k)\) is equal to \( f(x,k) \).

The classes \( \#paraw, \#paraw[1], \#paraw_{1} \), and \( \#paraw_{1} \)-tail \( C \) are defined in an analogous way. In this article, our main focus will be on the case where \( C \) is either \( L \) or DBP. For preciseness, we give a formal definition of these classes, starting with \( L \).

Definition 11. Let \( f \) be a parameterised function. Then, \( f \in \#paraw-L \) if there exists a computable function \( g : \mathbb{N} \to \mathbb{N} \), and a \( k \)-bounded, \( O(\log |x| + g(k)) \) space bounded NTM \( M \) such that for all \((x,k)\): \( f(x,k) = \#\text{acc}M(x,k) \). We say that \( f \) is also in

- \( \#paraw_{1} \)-L if \( M \) has read-once access to its nondeterministic bits,
- \( \#paraw[1]-L \) if \( M \) is tail-nondeterministic, and
- \( \#paraw_{1}\text{-tail}-L \) if \( M \) has read-once access to its nondet. bits and is tail-nondeterministic.

For any counting complexity class, it is important to have closure properties under arithmetic operations such as addition and multiplication. We observe, that the classes defined above are closed under these operations.

Theorem 12. For any \( C \in \{ \#paraw-L, \#paraw[1]-L, \#paraw_{1}-L, \#paraw_{1}\text{-tail}-L \} \), the class \( C \) is closed under addition and multiplication.

Proof. We first consider the case where \( C \in \{ \#paraw-L, \#paraw_{1}-L \} \). The argument is similar for both of the classes. We give details for the case when \( C = \#paraw-L \). Let \( f_{1} \) and \( f_{2} \) be in \( \#paraw-L \) via the NTMs \( M_{1} \) and \( M_{2} \). Let \( h \) be a computable function such that \( M_{1} \) and \( M_{2} \) are \( c \cdot \log |x| + h(k) \) space bounded. Now, consider an NTM \( M \) that nondeterministically chooses a single bit and simulates \( M_{1} \) on \((x,k)\) if the bit is 0, else simulates \( M_{2} \). Clearly, \( M \) is \( O(\log |x| + h(k)) \) space bounded. The number of nondeterministic bits used by \( M \) on any computation path is one more than the maximum of the number of nondeterministic bits used by \( M_{1} \) and \( M_{2} \) on any computation path and, as a result, \( M \) is \( k \)-bounded. Now, from the construction of \( M \), we have \( \#\text{acc}M(x,k) = f_{1}(x,k) + f_{2}(x,k) \).

We conclude that the class \( \#paraw-L \) is closed under addition. Exactly the same argument works for the case when \( M_{1} \) and \( M_{2} \) have read-once access to the nondeterministic bits, have tail-nondeterminism, or both.

For the case of multiplication, consider the NTM \( M' \) that simulates \( M_{1} \) and \( M_{2} \) as follows: \( M' \), on input \((x,k)\), first simulates \( M_{1} \) on input \((x,k)\). If \( M_{1} \) accepts, then \( M' \) simulates \( M_{2} \) on \((x,k)\), and accepts if and only if \( M_{2} \) does so. Since \( M_{1} \) and \( M_{2} \) are \( k \)-bounded, \( M' \) is also \( k \)-bounded. The space complexity of \( M' \) is at most the maximum of that of \( M_{1} \) and \( M_{2} \). Now, from construction of \( M' \), we have that \( \#\text{acc}M'(x,k) = f_{1}(x,k) \cdot f_{2}(x,k) \). Accordingly, the class \( \#paraw-L \) is closed under multiplication. Additionally, if \( M_{1} \) and \( M_{2} \) only have read-once access to the nondeterministic bits, so has the new machine \( M' \). We conclude that \( \#paraw_{1}-L \) is closed under multiplication.

It may be noted that the construction above does not work in the case of tail-nondeterministic machines. In order to prove closure of the classes \( \#paraw[1]-L \) and \( \#paraw_{1}\text{-tail}-L \) under multiplication, the construction of a \( k \)-bounded NTM \( M \) is as follows. On input \((x,k)\), \( M \) first simulates \( M_{1} \) on input \((x,k)\) until it encounters a nondeterministic transition in \( M_{1} \). As soon as it sees a nondeterministic transition in \( M_{1} \), \( M \) stores the encoding of this configuration in its work-tape without simulating the step of \( M_{1} \). Then \( M \) starts simulating machine \( M_{2} \) from its initial configuration on input \((x,k)\) until a nondeterministic transition of \( M_{2} \) is seen.
Now, the machine $M$ simulates $M_1$ starting from its stored configuration. If $M_1$ accepts, then $M$ runs $M_2$ from its stored configuration accepting if and only if $M_2$ accepts. It is not hard to see that the resulting machine is tail-nondeterministic and satisfies the required properties. Additionally, if $M_1$ and $M_2$ require only read-once access to the nondeterministic choices, so does $M$. This concludes the proof.

Using the notion of bounded nondeterminism for the case of branching programs, we can define the counting versions of the branching program based complexity classes. Here, counting is based on the number of accepting paths (that is, $s$-$t$-paths) with respect to all possible assignments to the nondeterministic bits.

\begin{definition}
Let $f$ be a parameterised function. Then, $f \in \#\text{para}_W\text{-DBP}$ if there exists a $k$-bounded family $P = \{P_{n,m}\}_{n,m \geq 0}$ of DBPs such that for all $(x,k)$: $f(x,k) = \#\text{acc}P_{|x|,|k|}(x,k)$. We say that $f$ is also in

- $\#\text{para}_\gamma\text{-DBP}$ if $P$ is read-once certified,
- $\#\text{para}_W[\gamma]\text{-DBP}$ if $P$ is tail-nondeterministic, and
- $\#\text{para}_\gamma\text{-tail}\text{-DBP}$ if $P$ is read-once certified and tail-nondeterministic.

\end{definition}

It is known that $\mathsf{NL}$ coincides with class of all languages accepted by logspace uniform families of branching programs of polynomial size. In fact, the construction is folklore and the desired family of branching programs can be obtained from the configuration graph of a nondeterministic logspace bounded machine. Moreover, this relation is parsimonious and preserves the number of accepting paths, and consequently, applies to the corresponding counting classes as well. We show a similar relationship for the parameterised counting classes.

In general, the configuration graph of a nondeterministic space bounded machine is not layered and layering an arbitrary directed acyclic graph might require reachability which in turn does not allow logspace uniformity. However, we can achieve the layering of the configuration graph by adding a simple step counter as explained next.

Let $M$ be a $k$-bounded, $s$ space bounded TM. Without loss of generality, $M$ has a unique accepting configuration. Let $\hat{M}$ be the following modification of $M$ accepting the same language as $M$: In addition to the tapes of $M$, $\hat{M}$ has an extra tape that keeps a step counter for $M$ represented in binary. The machine $\hat{M}$ simulates $M$ step by step and after each step of $M$, it increments the counter by one. Note that at most $O(s)$ bits are required to store the counter. Note that, for a given input $x$, a configuration of $\hat{M}$ on $x$ can be represented as $(\gamma,i)$ where $\gamma$ is a configuration of $M$ on the input $x$, and $i$ is the content of the additional tape of $\hat{M}$ that contains the binary counter. To simplify things, we assume that $i$ is a number in $\{0,\ldots, 2^{O(s)}\}$, ignoring the head position information on the tape used for storing the counter. The normalised configuration graph of $\hat{M}$ is a graph on the set of all possible configurations of $\hat{M}$ on any input of length $n$, with the edge relations as follows. There is an edge from configuration $(\gamma,i)$ to $(\gamma',i')$ if and only if $\gamma'$ is reachable from $\gamma$ in a single step of $M$ and $i' = i + 1$. That is, the steps required to update the counter are treated as a single edge in the normalised configuration graph.

A generic configuration graph $G_{M,n,m}$ of $M$ on inputs $(x,k)$ with $x = x_1 \cdots x_n$ and $m = |k|$ is the normalised configuration graph of $\hat{M}$, where the vertices are the set of all possible configurations of $\hat{M}$ on any input length $n,m$ as above. Furthermore, we assume that the vertices of $G_{M,n,m}$ are labelled from variables in $\{x_1,\ldots,x_n,k_1,\ldots,k_m\}$ and edges are labelled from $\{0,1\}$ in the same way as for branching programs. It is not hard to see that a generic configuration graph for input lengths $n,m \geq 0$ is a layered directed acyclic graph (dag). Moreover, it can be seen that for any $(x,k) \in \{0,1\}^* \times \mathbb{N}$, $M$ accepts $(x,k)$ if and
only if there is a directed path from the initial configuration to the accepting configuration in $G_{M,|x|,|k|}$ such that the edge labels along the path evaluate to 1 under the input $(x, k)$. Finally, we may note that there is a \texttt{para-L}-machine $N$ that, given $(C, i)$ and $(C', i')$, decides if $G_M$ has an edge from $(C, i)$ to $(C', i')$. Accordingly, the construction of $G_{M,n,m}$ is \texttt{para-L} uniform.

We prove the following equivalence among the counting classes defined above.

> **Theorem 14.** For any $a \in \{W, W[1], \beta, \beta$-tail$\}$, we have that $\#\texttt{para}_a$-DBP $= \#\texttt{para}_a$-L.

**Proof.** We will first show that the BP based classes are contained in the corresponding logspace based classes (“$\subseteq$”). We give the detailed argument to the case of $\#\texttt{para}_W$-DBP and outline the changes required (if any) for the rest.

Let $f \in \#\texttt{para}_W$-DBP be a parameterised function via the family $P := (P_{n,m})_{n,m \geq 0}$ of branching programs of size $g(m) \cdot n^c$ for $c \in \mathbb{N}$. We assume that $P$ is \texttt{para-L}-uniform. Consider a $k$-bounded Turing machine $M$ which evaluates the BP, such that every $s$-t-path in $P_{|x|,|k|}$ corresponds to a unique accepting path of $M$. The machine $M$ is described in Algorithm 1 and implicitly uses the uniformity machine to query $P_{n,m}$.

It is evident from the construction that the set of all accepting paths $M$ are in bijective correspondence with the set of all $s$-t-paths in $P_{n,m}$, where $s$ is the starting and $t$ the accepting node of $P_{n,m}$. As a result, $\#\text{acc} M(x, k) = \#\text{acc} P_{|x|,|k|}(x, k)$, for all $(x, k) \in \{0, 1\}^* \times \mathbb{N}$. Note that the machine $M$ is $k$-bounded, and since there is a \texttt{para-L} uniformity machine for $P_{n,m}$, $M$ requires at most $O(log(g(k) \cdot |x|^c) + log(|x|)$ bits of space. Moreover, if $P_{n,m}$ is read-once certified for all $n, m$, then $M$ requires only a read-once access to nondeterministic bits. Finally, if $P_{n,m}$ is tail-nondeterministic, so is $M$. This shows “$\subseteq$” for all four equalities.

Now, we prove “$\supseteq$”. Our argument crucially uses the fact that the generic configuration graph $G_{M,n,m}$ of a $k$-bounded machine is layered and can be constructed using a \texttt{para-L} uniformity machine. As in the case of “$\subseteq$”, we argue for the case of $\#\texttt{para}_W$-L and mention the modifications required (if any) for the remaining classes. Let $f \in \#\texttt{para}_W$-L via the $k$-bounded machine $M$ using $O(log(n) + g(k))$ space on all inputs $(x, k)$. Without loss of generality, we assume that the machine $M$ reads either from the input tape or from the choice tape in any configuration. Let $P_{n,m} := G_{M,n,m}$ be the generic configuration graph of $M$ for input length $n$ and $m = |k|$. Then, $M$ accepts $(x, k)$ if and only if there is an assignment to $y := y_1, \ldots, y_l \in \{0, 1\}$ such that $P_{n,m}$ has a directed path from the initial configuration to the accepting configuration. In fact, there is 1-to-1-correspondence between accepting computation paths of $M$ and choices for $y$. As a result, $\#\text{acc} P_{|x|,|k|}(x, k) = \#\text{acc} M(x, k)$ as
required. Since \(G_{M,n,m}\) can be constructed using a \(\text{para-L}\) uniformity machine. This shows that \(\text{para}_{\text{W}-L} \subseteq \text{para}_{\text{W}-\text{DBP}}\).

For the case of \(\text{para}_{\beta}\text{-L}\), we need to show that the resulting DBP \(P_{n,m}\) is read-once certified. However, it may happen that the configuration \((C, i)\) reads variable \(y_j\), whereas \((C', i)\) reads variable \(y_{j'}\) with \(j \neq j'\). This makes \(P_{n,m}\) far from being read-once certified. However, a crucial observation is that in any start to terminal path in \(P_{n,m}\), the \(y\)-variables are read in the order \(y_1, \ldots, y_t\), and if \(y_j\) is read at any point, then none of the \(y_i\)'s for \(i < j\) will be read along this path after this point. Accordingly, with suitable staggering we can make \(P_n\) read-once certified. We sketch the process below.

For \(1 \leq i \leq m\), let \(i_\alpha\) be the largest number such that there is a configuration \(C\) such that \((C, i_\alpha)\) reads the nondeterministic bit \(y_i\). Now the idea is to stagger the computations so that the last read of \(y_i\) occurs, i.e., until layer \(i_\alpha\). That is for \(y_i\), we wait till layer number \(i_\alpha\) before proceeding to read \(y_2\), and for \(y_2\) we wait till layer \(2_\alpha\) before proceeding to read \(y_3\) and so on. This can be achieved by adding necessary dummy nodes that have a single outgoing edge labelled by 1. For the whole process we only need the value of \(i_\alpha\) which can be computed in \(\text{para-L}\) given access to the uniformity machine. Consequently, the overall staggering process can be done in \(\text{para-L}\) so that the resulting branching program is uniform. It may be noted that we do not alter the number of accepting paths during the above process.

From the above, we conclude that \(\text{para}_{\beta}\text{-L} \subseteq \text{para}_{\beta}\text{-DBP}\). Finally, in the case of tail-nondeterminism, we may note that if \(M\) is tail-nondeterministic then so is \(G_{M,n,m}\) for every input length \(n\). Since the above staggering process does not alter tail-nondeterminism, we conclude that \(\text{para}_{W[1]-L} \subseteq \text{para}_{W[1]-\text{DBP}}\) and \(\text{para}_{\beta\text{-tail-L}} \subseteq \text{para}_{\beta\text{-tail-DBP}}\).

**Remark 15.** In the above proof, for the graph \(P_{n,m} = G_{M,n,m}\), the properties such as \(k\)-bounded nondeterminism, read-once certified nondeterministic bits and tail-nondeterminism are preserved only in the component that contain the initial and accepting configurations. The remaining components may not satisfy these properties and these components are not relevant to the program.

In fact, in the proof of Theorem 14 we did not use the property that \(P_{n,m}\) is deterministic. Accordingly, with similar arguments, we get the following result (proof details are omitted).

**Theorem 16.** For any \(o \in \{W, W[1], \beta, \beta\text{-tail}\}\), we have that \(#\text{para}_{o}\text{-BP} = #\text{para}_{o}\text{-NL}\).

Now, as in the case of logspace bounded classes, closure under addition and multiplication follows witnessed by Theorem 12.

**Corollary 17.** For any class \(C \in \{#\text{para}_{W}\text{-DBP}, #\text{para}_{W[1]}\text{-DBP}, #\text{para}_{\beta}\text{-DBP}, #\text{para}_{\beta\text{-tail}}\text{-DBP}\}\), \(C\) is closed under sum and product.

## 4 Complete Problems

In this section, we consider natural problems which are complete for the introduced classes. The focus will be on reachability questions in which we count the number of paths.

| Problem: | \(p\text{-#REACH}_b\) |
| --- | --- |
| Input: | DAG \(G = (V, E)\) s.t.f.a. \(v \in V: \deg v \leq b, s, t \in V\) and \(a \in \mathbb{N}\). |
| Parameter: | \(k\). |
| Output: | number of \(s\)-\(t\)-paths of length \(a\) if \(a \leq k \cdot \log |V|\), 0 otherwise. |

**Theorem 18.** \(p\text{-#REACH}_b\) is \(\text{para}_{\beta}\text{-L}\)-complete with respect to \(\leq_{m}^{\text{pol}}\text{-reductions}\).
Adjacency within $G$ for membership, we use Algorithm 2 but nondeterministically guess each pair of edges. Furthermore, we extend the graph such that all paths through only deterministic configurations are substituted by a single edge. That there is a unique accepting configuration.

### Theorem 19.

$p$-$\#\text{PATH}_b$ is $\#\text{para}_γ$-complete with respect to $\leq_{\text{P}log}$-reductions.

**Proof.** For membership, we use Algorithm 2 but nondeterministically guess each pair $s, t \in V$.

For hardness, consider some problem $L \in p$-$\#\text{para}_γ$-L accepted by a machine $M$ and let $f$ be a computable function. Without loss of generality, we assume that on all inputs $(x, k)$ every computation paths of $M$ on input $x$ use exactly $f(k) \cdot \log(|x|)$ nondeterministic bits and that there is a unique accepting configuration.

Similarly as in the proof of Theorem 18, let $G(x) = (V(x), E(x))$ be the configuration graph such that all paths through only deterministic configurations are substituted by a single edge. Furthermore, we extend $G(x)$ such that we add a path of fresh vertices $v_1, \ldots, v_{\ell \log |x|}$
with \((v_i, v_{i+1}) \in E(x)\) for \(1 \leq i < \log |x|\). The reason for this construction lies in possible “bad sequences” of configurations as depicted in Figure 1. Finally, we extend \(G(x)\) by \(m\) isolated vertices such that \(m + |V(x)| \geq |x|\). Now, any path in \(G(x)\) going from \(v_1\) to the initial configuration \(s(x)\) and then to the accepting configuration \(t\) is of length \(\ell := (f(k) + 1) \cdot \log |x|\).

Notice, that the number of such \(v_1\)-\(t\)-paths is equivalent to the number of accepting paths of \(M\) on input \(x\). To precisely calculate \(#\text{acc}_M(x)\), we will use two oracle calls to different \(p\)-\#PATH instances yielding a \(\leq p\log\)-reduction as required.

At first, we compute the result \(n_1\) of the oracle call \(p\)-\#PATH\(_b\)(\(G(x), \ell, \log |x|\)). Then, we modify \(G(x)\) yielding a graph \(G'(x)\) by deleting the edge \((v_1, v_2)\). This ensures that in \(G'(x)\) among paths of length \((f(k) + 1) \cdot \log |x|\) exactly the “good” accepting paths are missing compared to \(G(x)\). Then we store the value of the oracle call \(p\)-\#PATH\(_b\)(\(G'(x), \ell, \log |x|\)) in the variable \(n_2\). Finally, calculate the difference \(n_1 - n_2\) which is equivalent to \(#\text{acc}_M(x)\). ▶

Next we consider a problem that combines a reachability problem with model-checking for propositional logic, that is, we only count paths that are models of a given propositional formula. This idea stems from Haak et al. [15].

\section*{Figure 1} Construction of \(G(x)\) in the proof of Theorem 19. The black chains of unnamed vertices depict possibly occurring “bad” configuration sequences in the configuration graph.
Proof. Membership: We can use Algorithm 2 to find paths as before. We then need to check whether the chosen path also satisfies the formula $\varphi$. For this, we can use a standard logspace-algorithm for propositional model-checking and whenever we need the value of a variable, we re-compute the whole path constructed using Algorithm 2 before reusing the nondeterministic bits and check whether the respective edge is used in that path. This obviously yields a $\#\text{para}_W$-$L$-algorithm.

Hardness: We use the same construction as in the proof of Theorem 18. The difference is that nondeterministic bits can be reused. What this means that we need to ensure that we only count paths that on which different queries to the same nondeterministic bit assume the same value for that bit. Let $G(x, k) = (V(x, k), E(x, k))$, $s(x, k)$, $t$ and $f$ be as in the proof of Theorem 18. Let $\ell: E(x, k) \rightarrow \mathbb{N}$ be a labeling function stating which nondeterministic bit is read on each edge of $G(x, k)$ and let $\text{val}: E(x, k) \rightarrow \{0, 1\}$ be a function which value of the nondeterministic bit corresponds to each edge. We can now define

$$\varphi(x, k) = \bigwedge_{e_1, e_2 \in E(x, k), \ell(e_1) = \ell(e_2), \text{val}(e_1) \neq \text{val}(e_2)} \neg e_1 \lor \neg e_2,$$

expressing that the values assumed for the nondeterministic bits are consistent throughout a path. $\varphi(x, k)$ can be computed in parameterised logspace. This allows us to define the desired reduction as $(x, k) \mapsto ((G(x, k), s(x, k), t, \varphi(x, k), f(k)) \cdot \log(|x|)), f(k))$.

Similarly, one can define a problem $\text{p-}\#\text{CycleCover-CNF}$ which is counting cycle covers that satisfy a given CNF-formula and thereby one easily obtains the following corollary.

$\triangleright$ Corollary 21. $\text{p-}\#\text{CycleCover-CNF}$ is $\#\text{para}_W$-$L$-complete with respect to $\leq^{\text{plog}}_T$.

5 Conclusion

In this paper, we introduced parameterised counting classes defined via bounded nondeterminism and focused on (nondeterministic) logarithmic space as well as (deterministic) branching programs. We showed closure properties of these classes with respect to sum and product (Theorem 12 and Corollary 17). Furthermore, we showed that deterministic branching programs coincide with logarithmic space under these notions (Theorem 14) and observed the same for $\text{BP}$ and $\text{NL}$ (Theorem 16). Moreover, we presented natural problems complete for $\text{para}_\beta$-$L$ (Theorem 18 and 19) as well as $\text{para}_W$-$L$ (Theorem 20 and Corollary 21) in the form of reachability questions. Notice, that the studied problems can be considered restricted to dags yielding the same completeness results. The following are interesting questions for further research:

- Establish a broader spectrum of complete problems for the classes $\text{para}_\beta$-$L$ and $\text{para}_W$-$L$, e.g., in the realm of satisfiability questions.
- Find natural complete problems for classes based on nondeterministic branching problems.
- Further study closure properties of the classes introduced in this work.
- It could be interesting to study the connection between $\text{para}_\beta$-$L \cap \text{para}_W[1]$-$L$ and the class $\text{para}_\beta$-$\text{tail}$-$L$—might these two classes coincide?
- It could be interesting to identify further characterisations of the introduced classes, e.g., in the vein of descriptive complexity, which could highlight their robustness.
References

1. Manindra Agrawal, Eric Allender, and Samir Datta. On $\text{TC}^0$, $\text{AC}^0$, and arithmetic circuits. *J. Comput. Syst. Sci.*, 60(2):395–421, 2000. URL: https://doi.org/10.1006/jcss.1999.1675

2. Eric Allender, Robert Beals, and Mitsunori Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8(2):99–126, 1999. URL: https://doi.org/10.1007/s000370050023

3. Max Bannach, Christoph Stockhusen, and Till Tantau. Fast parallel fixed-parameter algorithms via color coding. *arXiv preprint arXiv:1509.06984*, 2015.

4. Hervé Caussinus, Pierre McKenzie, Denis Thérien, and Heribert Vollmer. Nondeterministic $\text{NC}^1$ computation. *J. Comput. Syst. Sci.*, 57(2):200–212, 1998. URL: https://doi.org/10.1006/jcss.1998.1588

5. Yijia Chen and Jörg Flum. Some lower bounds in parameterized $\text{AC}^0$. *arXiv preprint arXiv:1606.08014*, 2016.

6. Yijia Chen, Jörg Flum, and Martin Grohe. Bounded nondeterminism and alternation in parameterized complexity theory. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003)*, 7–10 July 2003, Aarhus, Denmark, pages 13–29. IEEE Computer Society, 2003. URL: https://doi.org/10.1109/CCC.2003.1214407

7. Radu Curticapean. Counting matchings of size k is $\text{w}[1]$-hard. In Fedor V. Fomin, Rūsiņ Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013*, Riga, Latvia, July 8-12, 2013, *Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 352–363. Springer, 2013. URL: https://doi.org/10.1007/978-3-642-39206-1_30

8. Radu Curticapean. *The simple, little and slow things count: on parameterized counting complexity*. PhD thesis, Saarland University, 2015. URL: http://scidok.sulb.uni-saarland.de/volltexte/2015/6217/

9. Radu Curticapean. Block interpolation: A framework for tight exponential-time counting complexity. *Inf. Comput.*, 261(Part):265–280, 2018. URL: https://doi.org/10.1016/j.ic.2018.02.008

10. Samir Datta, Meena Mahajan, B. V. Raghavendra Rao, Michael Thomas, and Heribert Vollmer. Counting classes and the fine structure between $\text{NC}^1$ and $L$. *Theor. Comput. Sci.*, 413(43):3678–3695, 2012. URL: https://doi.org/10.1016/j.tcs.2011.05.050

11. Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

12. Jörg Flum and Martin Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.

13. Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004. URL: https://doi.org/10.1137/S0097539703427203

14. Jörg Flum and Martin Grohe. *Parameterized complexity theory*. Springer Science & Business Media, 2006.

15. Anselm Haak, Juha Kontinen, Fabian Müller, Heribert Vollmer, and Fan Yang. Counting of teams in first-order team logics. *CoRR*, abs/1902.00246, 2019. URL: http://arxiv.org/abs/1902.00246

16. Meena Mahajan and B. V. Raghavendra Rao. Small-space analogues of valiant’s classes. In *Fundamentals of Computation Theory, 17th International Symposium, FCT 2009*, Wroclaw, Poland, September 2-4, 2009. *Proceedings*, pages 250–261, 2009. URL: https://doi.org/10.1007/978-3-642-03409-1_23
Parameterised Counting Classes with Bounded Nondeterminism

17. Mitsunori Ogihara. The PL hierarchy collapses. *SIAM J. Comput.*, 27(5):1430–1437, 1998. URL: [https://doi.org/10.1137/S0097539795295924](https://doi.org/10.1137/S0097539795295924).

18. Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

19. Christoph Stockhusen. *On the space and circuit complexity of parameterized problems*. PhD thesis, University of Lübeck, Germany, 2017. URL: [http://www.zhb.uni-luebeck.de/epubs/ediss1780.pdf](http://www.zhb.uni-luebeck.de/epubs/ediss1780.pdf).

20. Christoph Stockhusen and Till Tantau. Completeness results for parameterized space classes. In *International Symposium on Parameterized and Exact Computation*, pages 335–347. Springer, 2013.

21. Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991. URL: [https://doi.org/10.1137/0220053](https://doi.org/10.1137/0220053).

22. Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. URL: [https://doi.org/10.1016/0304-3975(79)90044-6](https://doi.org/10.1016/0304-3975(79)90044-6).

23. Heribert Vollmer. *Introduction to circuit complexity: a uniform approach*. Springer Science & Business Media, 2003.