Fixed-parameter parallelizability

Jeffrey Finkelstein
Computer Science Department, Boston University
September 6, 2016

Abstract

Fixed-parameter parallelizability...

Contents

1 Introduction 2

2 Definitions 2

3 Fixed-parameter parallelizability 3

3.1 Definition of paraNC 3

3.2 Example problem in paraNC 4

3.3 Relationship between paraNC and NC 6

3.4 Approximable optimization problems 9

4 Fixed-parameter tractability 11

4.1 Definition of paraP 11

4.2 Example problem in paraP 11

4.3 Completeness in paraP 12

4.4 Parameterized complexity of efficient verification 14

5 paraNC is to NC as paraWNC is to NNC 16

5.1 Definition of paraWNC 16

5.2 Example problems in paraWNC 17

5.2.1 Small generating set problems 17

5.2.2 Weighted circuit satisfiability problems 18

5.2.3 Weighted formula satisfiability problems 20

5.3 Completeness in paraWNC 21
1 Introduction

We show the existence of a P-complete problem in paraNC (the circuit evaluation problem parameterized by depth) and a P-complete problem not in paraP, unless NC = P (the “small” circuit evaluation problem). We prove that paraNC = paraWNC exactly when NC = NNC[i(n) log n] subsection 5.4.

2 Definitions

Definition 2.1. A Boolean circuit, or simply a circuit, C, is an directed acyclic graph. The size of a circuit, denoted size(C), is the number of vertices in the underlying graph. The depth of a circuit, denoted depth(C), is the length of a longest path from the root to a sink.

Definition 2.2. A function f is circuit-computable if there is a nonuniform family of Boolean circuits \{C_n\}_{n \in \mathbb{N}} such that for each x we have f(x) = C_n(x), where n = |x|.

A language is circuit-decidable if it has a circuit-computable characteristic function. We may also require that the size and depth of each circuit C_n in the family be circuit-computable from just n, the length of the input. In this case, we say the language is circuit-decidable with uniform size and depth.

Nonuniformity is required in Lemma 3.10 among other theorems, in which the size of the input relative to the size of the parameter for an instance of the parameterized problem selects which of two circuits to use. If the circuits were uniform, we would need to include both alternatives as subcircuits of a single larger circuit, thereby making the circuits too deep to meet the required circuit depth bounds. TODO Early on, I tried to see what would happen if we place the parameterized running time in the uniformity part of the computation (i.e. paraL-uniform NC circuits instead of L-uniform paraNC circuits), but I could not show anything. However, I would like to take another look at this. It may be possible to adapt some theorems in later sections to use uniform circuits, but we did not pursue this.
Definition 2.3 (Decision problems and parameterized problems). A language is a set of binary strings. A parameterization is a computable function $\kappa$ from binary strings to natural numbers. A parameterized problem is a pair $(Q, \kappa)$, where $Q$ is a language and $\kappa$ is a parameterization.

Definition 2.4 (Slices of parameterized problems). For each positive integer $k$ and each parameterized problem $(Q, \kappa)$, the $k$th slice of $Q$, denoted $(Q, \kappa)_k$, is defined by

$$(Q, \kappa)_k = \{(x, k) \mid x \in Q \text{ and } \kappa(x) = k\}.$$ 

Definition 2.5 ($\text{NNC}^d$ and $\text{NC}^d$). Let $d$ be a natural number. A language $Q$ is in the class $\text{NNC}^d[\Theta(n)]$ if there is a nondeterministic circuit family $\{C_n\}$ such that for each string $x$ of length $n$,

- $x \in Q$ if and only if $C_n(x) = 1$,
- $\text{size}(C_n) \leq n^{O(1)}$,
- $\text{depth}(C_n) \leq O(\log^d n)$,
- $\text{nondet}(C_n) \leq b(n)$.

If $b$ is the zero function, then the language is in the class $\text{NC}^d$.

Here, the notion of “acceptance” for a circuit is a nondeterministic one: $C_n(x) = 1$ if and only if there is a binary string $w$ of length $b(n)$ such that $C_n(x, w) = 1$.

Throughout we will often assume without loss of generality that functions like circuit size and depth bounds, nondeterminism bounds, and polynomials, are increasing.

3 Fixed-parameter parallelizability

The $P$-complete problems are considered tractable but inherently sequential; adding more processors does not provide any significant reduction in the time required to find a solution, even though they can be solved in polynomial time. But can the lack of parallelizability for these problems depend on a parameterization of the problem, thereby isolating some of the “sequential-ness” of the problem in the value of the parameter?

3.1 Definition of $\text{paraNC}$

The $\text{para}$ “operator” defined in [16] applies generically to an arbitrary complexity class as follows. If $\mathcal{C}$ is a class of decision problems, then $\text{paraC}$ is the class of parameterized problems $(Q, \kappa)$ for which there is a decision problem $L \in \mathcal{C}$ and a computable function $f$ such that $x \in Q$ if and only if $(x, 1^{f(\kappa(x))}) \in L$. When $\mathcal{C} = \text{NC}$ in particular, we get the following equivalent definition.
Definition 3.1 (paraNC\(_d\)). Let \(d\) be a natural number. A parameterized problem \((Q, \kappa)\) is in the class paraNC\(_d\) if there is a circuit-computable function \(f\) and a nonuniform family \(\{C_{n,k}\}\) of bounded fan-in Boolean circuits such that for each string \(x\),

- \(x \in Q\) if and only if \(C_{n,k}(x) = 1\), where \(n = |x|\) and \(k = \kappa(x)\),
- \(\text{size}(C_{n,k}) \leq f(k)n^{O(1)}\),
- \(\text{depth}(C_{n,k}) \leq f(k) + O(\log^d n)\).

If the depth of the circuit is instead bounded by \(f(k)O(\log^d n)\), the class is denoted paraNC\(_d\)↑, a superclass of paraNC\(_d\). If the circuits are of unbounded fan-in, the classes are paraAC\(_d\) and paraAC\(_d\)↑, respectively. The classes paraAC\(_d\)↑ were first defined in [3].

A subtle point is that the value of the parameter \(\kappa(x)\) must be non-constant but also independent of the size of the instance \(x\) for the parameterized problem to be interesting. First, if \(\kappa(x)\) were bounded above by a constant for each \(x\), then the parameter would be irrelevant and the problem would simply be in the standard complexity class NC\(_d\). Thus depth \(O(\log^d n)\) and depth \(f(k)\log^d n\) are different. On the other hand, if \(\kappa(x)\) were bounded from below by a nondecreasing, unbounded function of \(|x|\), then the problem would be trivially in paraAC\(_0\)↑ by the technique of [17, Proposition 1.7]. Thus a formula like \(\log^2(kn)\), which may appear in the analysis of certain simulations of parameterized complexity classes (see Lemma 5.9 for example), becomes
\[
\log^2(kn) = (\log k + \log n)^2 = \log^2 k + \log k \log n + \log^2 n \leq \log^2 k + 2 \log^2 n,
\]
and thus \(\log^2(kn) = f(k) + O(\log^2 n)\) for some computable function \(f\).

3.2 Example problem in paraNC

The parameterized vertex cover problem is in paraAC\(_0\) [3], but the underlying decision problem is \(\text{NP}-\text{complete}\). We are interested in finding a problem in paraNC whose underlying decision problem is \(\text{P}-\text{complete}\). One way to do this is to choose \(Q\) to be a \(\text{P}-\text{complete}\) problem and \(\kappa\) to be the “degenerate” parameterization function \(\kappa(x) = |x|\).

Definition 3.2 (p-s-Circuit Evaluation).

- **Instance**: Boolean circuit \(C\) on \(n\) inputs with size \(m\) and depth \(d\), binary string \(x\) of length \(n\).
- **Parameter**: \(m\).
- **Question**: Does \(C(x) = 1\)?

Theorem 3.3. p-s-Circuit Evaluation is in paraNC and the underlying decision problem is \(\text{P}-\text{complete}\).

Proof. The circuit evaluation problem is \(\text{P}-\text{complete}\) by [22]. Since the parameterization is monotonically increasing with the size of the input, the problem is in paraNC by the technique of [17, Proposition 1.7].
To find a non-degenerate example, we can parameterize the circuit evaluation problem by depth instead of size.

**Definition 3.4** (*p*-d-Circuit Evaluation).

**Instance:** Boolean circuit $C$ on $n$ inputs with size $m$ and depth $d$, binary string $x$ of length $n$.

**Parameter:** $d$.

**Question:** Does $C(x) = 1$?

**Theorem 3.5.** *p*-d-Circuit Evaluation is in $\text{paraAC}^{0\uparrow}$ and the underlying decision problem is $\text{P}$-complete.

**Proof.** As stated in the proof of the previous theorem, the circuit evaluation problem is $\text{P}$-complete. Evaluating the circuit $C$ of size $m$ and depth $d$ on inputs $x$ can be performed by the depth-universal circuit $U$ of [13]. The size of $U$ is $O(m)$ and the depth is $d$, so there is a function $f$ such that the size is bounded by $f(d)m^{O(1)}$ and the depth by $f(d)$. Therefore the circuit evaluation problem parameterized by circuit depth is in $\text{paraAC}^{0\uparrow}$.

For a problem with a standard parameterization derived from an optimization problem (see Definition 3.15 below), consider the “depth of ones” problem. The circuit evaluation problem is a special case of the depth of ones problem if we choose $k$ to be the depth of the circuit $C$.

**Definition 3.6** (*p*-Ones Depth).

**Instance:** Boolean circuit $C$ on $n$ inputs with size $m$ and depth $d$, binary string $x$ of length $n$, positive integer $k$.

**Parameter:** $k$.

**Question:** When evaluating $C(x)$, do ones propagate to depth at least $k$?

As an optimization problem, the depth of ones problem is inapproximable up to any constant factor by any $\text{NC}$ circuit, unless $\text{NC} = \text{P}$ [21]. Contrast this with the complexity of the corresponding parameterized problem.

**Theorem 3.7.** *p*-Ones Depth is in $\text{paraAC}^{0\uparrow}$ and the underlying decision problem is $\text{P}$-complete.

**Proof.** Computing the depth of ones in a circuit is $\text{P}$-complete [21] (see also [19] Problem A.1.10)). The naïve algorithm for solving this problem is to take the subcircuit consisting of all gates starting from the inputs and extending through layer $k$, evaluating that (multi-output) circuit, then applying a single or gate to decide whether any of the gates at layer $k$ evaluated to one. For each gate at layer $k$, use an instance of the depth-universal circuit to evaluate the single-output circuit induced by that gate. This yields a circuit of depth $O(k)$ and size $f(k)m^{O(1)}$ for some $f$, where $m$ is the size of the circuit given as input. Therefore this problem is in $\text{paraAC}^{0\uparrow}$.  

5
3.3 Relationship between $\text{paraNC}$ and $\text{NC}$

The lemmas in this section allow us to transform, in certain circumstances, a highly parallel algorithm for a decision problem into a fixed-parameter parallelizable one for a parameterized problem, or vice versa. They will be used in later sections to provide evidence against the collapse of larger complexity classes to $\text{paraNC}$.

Lemma 3.8. For each nondecreasing, unbounded, circuit-computable function $i$, there is a function $f_i$ such that $f_i(i(n)) \geq n$ for each $n \geq f(1)$. Furthermore, $f_i$ is nondecreasing, unbounded, and circuit-computable. (We call $f_i$ the “upper inverse” of $i$.)

Proof. Define $f_i$ by

$$f_i(k) = \max\{n_0 \in \mathbb{N} | \forall n \geq n_0 : i(n) \geq k\}.$$ 

Since $i$ is nondecreasing and unbounded, so is $f_i$.

To compute $f_i$, we use the fact that $i$ is nondecreasing is unbounded. We know that for each $k$ there is a natural number $n_k$ such that for all $n \geq n_k$, we have $i(n) \geq k$. Thus the algorithm for computing $f_i$ take $k$ as input and performs a binary search on $i(1), \ldots, i(n_k)$ to determine the largest $n$ such that $i(n) \geq k$. There will be at most $\log n_k$ comparison subcircuits, each requiring a computation of $i$ and a comparison with the integer $k$ (in binary, say), so the overall depth of the circuit computing $f_i$ is $O(\text{depth}(i) \log n \log \log k)$ and the size is $O(\text{size}(i) \log n \log \log k)$.

Definition 3.9. Suppose $d$ is a natural number, $(Q, \kappa)$ is a parameterized problem, and $Q'$ is a decision problem. There is a small parameter $\text{NC}^d$ many-one reduction from $(Q, \kappa)$ to $Q'$ if there is a nondecreasing, unbounded, circuit-computable function $i$ and an $\text{NC}^d$ family of circuits $\{R_n\}_{n \in \mathbb{N}}$ such that for each string $x$ of length $n$ with $\kappa(x) \leq i(n)$, we have $x \in Q$ if and only if $R_n(x) \in Q'$.

This lemma demonstrates that the closure of $\text{NC}$ under small parameter reductions is $\text{paraNC}$.

Lemma 3.10. Suppose $d$ is a natural number, $(Q, \kappa)$ is a parameterized problem, and $Q'$ is a decision problem. If $Q$ is circuit-decidable with uniform size and depth, $Q'$ is in $\text{NC}^d$, and there is a small parameter $\text{NC}^d$ many-one reduction from $(Q, \kappa)$ to $Q'$, then $(Q, \kappa)$ is in $\text{paraNC}^d$.

Proof. Let $i$ be the function that defines the upper bound on the parameter, below which there is an $\text{NC}^d$ many-one reduction from $Q$ to $Q'$. Let $\{R_n\}$ be the $\text{NC}^d$ circuit family computing the reduction. The nonuniform family of circuits $\{A_{n,k}\}$ that decides $(Q, \kappa)$ is defined by

$$A_{n,k} = \begin{cases} C_n^1 & \text{if } i(n) < k \\ C_n^2 \circ R_n & \text{otherwise} \end{cases}$$
where \{C^1_n\} is the family of circuits that decides \(Q\) with uniform size and depth, \{C^2_n\} is the family of \(\text{NC}^d\) circuits that decides \(Q'\), and \(n'\) is the number of output bits of \(R_n\). The correctness of \(A_{n,k}\) follows from the correctness of the subsequent circuits.

If \(i(n) \geq k\), then the size and depth of the circuit are polynomial and polylogarithmic in \(n\), respectively, because the size and depth of \(C^2_n\) and \(R_n\) are. For the case when \(i(n) < k\), consider the upper inverse \(f_i\) of \(i\) guaranteed by Lemma 3.8. By construction, \(n \leq f_i(i(n)) < f_i(k)\). Now

\[
\begin{align*}
\text{size}(A_{n,k}) &= \text{size}(C^1_n) = S(n) \leq S(f_i(k)), \\
\text{depth}(A_{n,k}) &= \text{depth}(C^1_n) = D(n) \leq D(f_i(k)),
\end{align*}
\]

where \(S\) and \(D\) are the (circuit-computable, nondecreasing) size and depth bounds for the circuit family \{\(C^1_n\}\). Thus in either case, there is a sufficiently large circuit-computable function \(f\) such the size of \(A_{n,k}\) is bounded above by \(f(k)n^{O(1)}\) and the depth \(f(k) + O(\log^d n)\).

The following corollary highlights the special case of the preceding lemma in which the reduction is the identity function.

**Corollary 3.11.** Suppose \((Q, \kappa)\) is a parameterized problem, \(d\) is a positive integer, and \(i\) is an unbounded, nondecreasing, circuit-computable function. Let \(i(n)\)-\(Q\) denote the problem of deciding, given \(x\) with \(\kappa(x) \leq i(|x|)\), whether \(x \in Q\). If \(i(n)\)-\(Q\) is in \(\text{NC}^d\), then \((Q, \kappa)\) is in \(\text{paraNC}^d\).

**Proof.** The identity function is a small parameter \(\text{NC}^d\) many-one reduction from \((Q, \kappa)\) to \(i(n)\)-\(Q\), thereby proving that \(Q\) is in \(\text{paraNC}^d\) by the previous lemma.

This lemma shows that a many-one reduction to a fixed-parameter parallelizable problem can sometimes induce a highly parallel algorithm, if the parameter functions are bounded for the reduced instance.

**Lemma 3.12.** Suppose \(d\) is a positive integer, \(Q\) is a decision problem, and \((Q', \kappa')\) is a parameterized problem. Suppose there is an \(\text{NC}^d\) many-one reduction from \(Q\) to \(Q'\), given by the circuit family \(\{R_n\}\), and \((Q', \kappa')\) is in \(\text{paraNC}^d\) by a circuit family \(\{C_{m,k}\}\) of size \(f(k)n^{O(1)}\) and depth \(f(k) + O(\log^d m)\) on inputs of length \(m\). If \(f(\kappa'(R_n(x))) \leq \min(n^{O(1)}, O(\log^d n))\), then \(Q\) is in \(\text{NC}^d\).

**Proof.** The circuit family that decides \(Q\) is \(\{A_n\}\), defined by \(A_n = C_{m,k} \circ R_n\), where \(m\) is the size of the output of \(R_n\) and \(k = \kappa'(R_n(x))\). Since \(\text{size}(R_n) = n^{O(1)}\), we have \(m = n^{O(1)}\) as well. For correctness,

\[
x \in Q \iff R_n(x) \in Q' \iff C_{m,k}(R_n(x)) = 1.
\]
For size and depth bounds,

\[
\begin{align*}
\text{size}(A_n) &= \text{size}(C_{m,k}) + \text{size}(R_n) \\
&= f(k)n^{O(1)} + n^{O(1)} \\
&= f(k)n^{O(1)} + n^{O(1)} \\
&= n^{O(1)}n^{O(1)} + n^{O(1)} \\
&= n^{O(1)},
\end{align*}
\]

and

\[
\begin{align*}
\text{depth}(A_n) &= \text{depth}(C_{m,k}) + \text{depth}(R_n) \\
&= f(k) + O(\log^d m) + O(\log^d m) \\
&= f(k) + O(\log^d m) \\
&= f(k) + O(\log^d n) \\
&= O(\log^d n) + O(\log^d n) \\
&= O(\log^d n).
\end{align*}
\]

The following corollary highlights the special case of the preceding lemma in which the decision problem of interest is a “bounded-parameter” version of the decision problem underlying the fixed-parameter parallelizable problem; compare this with Corollary 3.11. Below, a “nontrivial” parameterized problem is one in which \( \emptyset \not\subset Q \subsetneq \{0,1\}^* \).

**Corollary 3.13.** Suppose \((Q,\kappa)\) is a nontrivial parameterized problem, \(d\) is a positive integer, and \(i\) is an unbounded, nondecreasing, circuit-computable function. Let \(i(n)-Q\) denote the problem of deciding, given \(x\) with \(\kappa(x) \leq i(|x|)\), whether \(x \in Q\). If \((Q,\kappa)\) is in \(\text{paraNC}^d\) by a circuit family \(\{C_{n,k}\}\) of size \(f(k)n^{O(1)}\) and depth \(f(k) + O(\log^d n)\) on inputs of length \(n\) and \(f(i(n)) \leq \min(n^{O(1)}, O(\log^d n))\), then \(i(n)-Q\) is in \(\text{NC}^d\).

**Proof.** We will show a many-one reduction from the decision problem \(i(n)-Q\) to the decision problem \(Q\) underlying the parameterized problem \((Q,\kappa)\) that satisfies the conditions of the previous lemma. The reduction \(\{R_n\}\) is defined as follows.

\[
R_n(x) = \begin{cases} 
 x & \text{if } \kappa(x) \leq i(n), \\
 \bot & \text{otherwise},
\end{cases}
\]

where \(\bot\) is an arbitrary string not in \(Q\) (which must exist because the problem is nontrivial by hypothesis). As long as \(\kappa\) is computable by an \(\text{NC}^d\) circuit family, then so is \(R_n\). (The computation of \(i(n)\) is captured by the nonuniformity of the circuit family, so it does not affect the size or depth required by the circuit computing \(R_n\).)

The reduction \(R_n\) is a correct many-one reduction. If \(x \in i(n)-Q\), then \(\kappa(x) \leq i(|x|)\) and \(x \in Q\), thus \(R_n(x) \in Q\). If \(x \notin i(n)-Q\), then there are two
cases. In the first, $\kappa(x) > i(|x|)$, in which case $R_n(x) = \bot$, which is not in $Q$ by construction. In the second case, $x \notin Q$ so $R_n(x) \notin Q$.

Finally, we consider the value of $f(\kappa(R_n(x)))$. If $\kappa(x) \leq i(n)$, then by construction

$$f(\kappa(R_n(x))) \leq f(\kappa(x)) \leq f(i(n)) \leq \min(n^{O(1)}, O(\log^d n)).$$

On the other hand, if $\kappa(x) > i(n)$, then $\kappa(R_n(x)) = \kappa(\bot) = O(1)$, which is bounded above by both $n^{O(1)}$ and $O(\log^d n)$ for all but finitely many $n$. Thus we have shown that $f(\kappa(R_n(x)))$ satisfies the upper bound required by Lemma 3.12 and the conclusion, $i(n)\cdot Q$ is in $\text{NC}^d$, follows.

### 3.4 Approximable optimization problems

Yet another way to do this is to find an optimization problem whose budget problem is $\text{P}$-complete while admitting a highly parallel approximation scheme.

**Definition 3.14.** An optimization problem $O$ is a four-tuple $(I, S, m, t)$, where $I$ is the set of instances, $S$ is the set of pairs $(x, w)$ where $w$ is a solution for $x$, the function $m$ computes the measure (or objective value) for such a pair, and $t$ is either min or max.

**Definition 3.15.** The standard parameterization of a minimization problem $O$, denoted $p-O$, is $(Q, \kappa)$, where $Q = \{(x, k) \mid m^*(x) \leq k\}$ and $\kappa(x, k) = k$. The inequality is reversed for a maximization problem.

**Definition 3.16.** Suppose $(I, S, m, t)$ is an optimization problem and $(x, y) \in S$. The performance ratio of the solution $y$ (with respect to $x$), denoted $R(x, y)$, is defined by

$$R(x, y) = \max_{m(x, y) \leq m^*(x)} \left( \frac{m(x, y)}{m^*(x)} \cdot \frac{m^*(x)}{m(x, y)} \right)$$

The performance ratio $R(x, y)$ is a number in the interval $[1, \infty)$. The closer $R(x, y)$ is to 1, the better the solution $y$ is for $x$, and the closer $R(x, y)$ to $\infty$, the worse the solution.

**Definition 3.17.** An approximation scheme for an optimization problem is a function $A$ such that for all $x$ and all positive integers $k$ we have $(x, A(x, k)) \in S$ and $R(x, A(x, k)) \leq 1 + \frac{1}{k}$.

An approximation scheme induces a family of functions, $\{A_k\}_{k \in \mathbb{N}}$, that form progressively better approximations for the optimization problem.

**Definition 3.18.** Suppose $O$ is an optimization problem with $O = (I, S, m, t)$ with $I$ and $S$ in $\text{NC}$ and $m$ in $\text{FNC}$. An optimization problem $O$ is in $\text{NCAS}$ if there is an approximation scheme $A$ for $O$ such that for each $k$, we have $A_k \in \text{NC}$, where $A_k(x) = A(x, k)$ for each $x$. The problem is in $\text{FNCAS}$ if there is an approximation scheme $A$ for $O$ such that $A \in \text{FNC}$ (i.e. on both inputs).

This definition is adapted from [17, Definition 1.31]
**Definition 3.19.** An optimization problem $O$ is in $\text{ENCAS}$ if there is a circuit family $\{A_{n,k}\}$ and circuit-computable functions $f$ and $g$ such that

- $\{A_{n,k}\}$ is an approximation scheme for $O$,
- $\text{size}(A_{n,k}) \leq f(k)n^{O(1)}$,
- $\text{depth}(A_{n,k}) \leq g(k)\log^{O(1)}n$.

**Proposition 3.20.** $\text{FNCAS} \subseteq \text{ENCAS} \subseteq \text{NCAS}$. 

**TODO** Positive linear programming is in $\text{NCAS}$, is it also in $\text{ENCAS}$? This theorem is an adaptation of [17, Theorem 1.32].

**Theorem 3.21.** Let $O$ be an optimization problem. If $O$ is in $\text{ENCAS}$, then $p\text{-}O$ is in $\text{paraNC}$. 

**Proof.** Assume without loss of generality that $O$ is a minimization problem; the proof is similar if it is a maximization problem. Let $\{m_n\}$ be the NC circuit family that computes the measure function. Let $\{A_{n,k}\}$ be the circuit family such that

- $R(x, A_{n,k}(x,k)) \leq 1 + \frac{1}{k}$ for each $x$ and $k$,
- $\text{size}(A_{n,k}) \leq f(k)n^{O(1)}$,
- $\text{depth}(A_{n,k}) \leq f(k) + O(\log^{O(1)}n)$,

for some circuit-computable function $f$. Define the circuit family $\{C_{n,k}\}$ as

$$C_{n,k}(x,k) = 1 \iff m(x, A_{n,k+1}(x,k+1)) \leq k,$$

so $C_{n,k}$ outputs 1 if and only if the approximate solution corresponding to parameter $k+1$ measures less than $k+1$. (The function $m$ is really a circuit as well, chosen from a family of circuits depending on the number of bits in its inputs.)

The size of $C_{n,k}$ is $O(\text{size}(m) + \text{size}(A_{n,k+1}))$ and the depth is $O(\text{depth}(m) + \text{depth}(A_{n,k+1}))$. For some sufficiently large circuit-computable function $f'$, the size and depth bounds are $f'(k+1)n^{O(1)}$ and $f'(k+1)+O(\log^{O(1)}n)$, respectively.

It remains to show correctness of $C_{n,k}$.

Let $x$ be a string, let $k$ be a natural number, and let $y = A_{n,k+1}(x,k+1)$. If $C_{n,k} = 1$, then $m(x,y) \leq k$, so $m^*(k) \leq k$ and therefore $(x,k) \in p\text{-}O$. For the converse, if $C_{n,k} = 0$, then $m(x,y) \geq k+1$, so

$$m^*(x) \geq \frac{m(x,y)}{1 + \frac{1}{k+1}} \geq \frac{k+1}{1 + \frac{1}{k+1}} = \frac{(k+1)^2}{k+2} > k.$$ 

Thus $(x,k) \notin p\text{-}O$. Therefore, we conclude that $p\text{-}O$ is in $\text{paraNC}$. 

\[\square\]
The converse does not hold: the minimum vertex cover problem is a counterexample because it is in \( \text{paraNC} \) [3, Theorem 4.5]. **TODO Show an example of an optimization problem whose budget problem is P-complete and whose standard parameterization is in paraNC but for which no ENCAS exists.**

Our goal now reduces to finding an optimization problem in ENCAS whose budget problem is P-complete.

**Definition 3.22 (Maximum Flow).**

**Instance:** directed graph \( G \), a natural number capacity \( c_e \) for each edge \( e \), source node \( s \), and target node \( t \).

**Solution:** flow \( F \), defined as a real number \( F_e \) for each edge \( e \) such that \( F_e \leq c_e \) and at each vertex the total in-flow is at least the total out-flow.

**Measure:** total in-flow at \( t \).

**Type:** maximization.

**Theorem 3.23.** If \( \text{NC} = \text{RNC} \), then the budget problem for Maximum Flow is P-complete and the standard parameterization is in paraNC.

**Proof.** The budget problem for Maximum Flow is P-complete [19, Problem A.4.4]. The Maximum Flow problem is in randomized FNCAS [14, Theorem 4.5.2]. If \( \text{NC} = \text{RNC} \), then randomized FNCAS equals deterministic FNCAS. Thus, the problem is in ENCAS, by Proposition 3.20. Finally, the standard parameterization is in paraNC by Theorem 3.21. \( \square \)

**TODO Can the randomization part of the RNC algorithm for MaxFlow be absorbed in the fixed-parameter part of the algorithm?**

## 4 Fixed-parameter tractability

### 4.1 Definition of paraP

The definition of paraP is analogous to that of paraNC. (For historical reasons, this class is usually known as FPT.)

**Definition 4.1.** A parameterized problem \((Q, \kappa)\) is in paraP if there is a deterministic Turing machine \( M \), a polynomial \( p \), and a computable function \( f \) such that \( M \) decides \( Q \) within \( f(\kappa(x))p(n) \) steps.

### 4.2 Example problem in paraP

Consider the parameterized high degree subgraph problem.

**Definition 4.2 (p-High Degree Subgraph, aka p-HDS).**

**Instance:** undirected graph \( G \), positive integer \( d \).

**Parameter:** \( d \).

**Question:** Does \( G \) have a vertex-induced subgraph of minimum degree at least \( d \)?
The underlying decision problem is in \( P \), so the parameterized problem is fixed-parameter tractable.

**Theorem 4.3** ([2]). \( p\text{-HDS} \) is in \( \text{para}P \).

The first two slices of the problem are in \( \text{NC} \), whereas all other slices are \( P \)-complete.

**Theorem 4.4** ([2]).

1. Both \( pHDS_1 \) and \( p\text{-HDS}_2 \) are in \( \text{NC} \).
2. For each positive integer \( d \) greater than two, \( p\text{-HDS}_d \) is \( P \)-complete.

TODO The proof that the high degree subgraph problem is \( P \)-complete does not translate to proving that the parameterized high degree subgraph problem is \( \text{para}P \)-complete; for that reduction to work, the parameterization would need to be on the size of the graph, not the degree \( d \).

TODO A reduction to place this problem in \( \text{para}EP \) does not seem possible, since it seems hard to encode the degree of a subgraph in that way?

### 4.3 Completeness in \( \text{para}P \)

We define \( P \)-completeness so that problems that are \( P \)-complete are unlikely to see a significant decrease in time complexity when parallelism is allowed, under the assumption that \( \text{NC} \neq P \). Let us define \( \text{para}P \)-completeness similarly, so that \( \text{para}P \)-complete problems are unlikely to see a significant decrease in “parameterized” time complexity when “parameterized” parallelism is allowed, under the assumption that \( \text{para}NC \neq \text{para}P \). We already know that each \( P \)-complete problem induces a \( \text{para}P \)-complete problem with a trivial parameterization [16, Proposition 14], however we are interested in natural problems with non-trivial parameterizations. This subsection proves the existence of nontrivial parameterized problems that are not in \( \text{para}P \) (under the assumption \( \text{para}NC \neq \text{para}P \)) and whose underlying decision problems are \( P \)-complete, complementing section 3, which proves the existence of parameterized problems in \( \text{para}NC \) whose underlying decision problems are \( P \)-complete. Furthermore, this fills a hole left by [15].

The assumption \( \text{para}NC \neq \text{para}P \) is reasonable because it is equivalent to the inequality \( \text{NC} \neq P \).

**Proposition 4.5.** \( \text{para}NC = \text{para}P \) if and only if \( \text{NC} = P \).

**Proof.** The proof is trivial if we use the definitions of the complexity class \( \text{para}C \) as the class of all parameterized problems \( (Q, \kappa) \) for which there is a language \( L \) in the complexity class \( C \) such that \( x \in Q \) if and only if \( (x, 1f^{\kappa(x)}) \in L \). See [16, Proposition 8], for example. \( \square \)
Definition 4.6 (paraP-completeness). A parameterized problem \((Q, \kappa)\) is paraP-hard if for each parameterized problem \((R, \lambda)\), there is a paraNC many-one reduction from \((R, \lambda)\) to \((Q, \kappa)\). If furthermore \((Q, \kappa)\) is in paraP, then it is paraP-complete.

Proposition 4.7. If a paraP-complete problem is in paraNC, then paraNC = paraP.

Proof. Follows from the downward closure of paraNC under paraNC many-one reductions.

The following problem is adapted from Short Deterministic Turing Machine Computation in [10].

Definition 4.8 (p-Bounded Halting Problem, aka p-BHP).

Instance: deterministic Turing machine \(M\), binary string \(x\) of length \(n\), positive integer \(t\) in unary, positive integer \(c\).

Parameter: \(t/n^c\)

Question: Does \(M\) accept \(x\) within \(t\) steps?

Theorem 4.9. p-BHP is paraP-complete.

Proof. The underlying decision problem is in P (by a standard simulation on the deterministic universal Turing machine), so the parameterized problem is in FPT. To show paraP-hardness, we use a generic reduction. Let \((Q, \kappa)\) be an arbitrary parameterized problem in paraP and let \(M\) be the deterministic Turing machine that decides \(Q\) in time \(f_M(k)n^c\) for some (circuit-)computable function \(f_M\) and some positive integer \(c\). The reduction is \(x \mapsto (M, x, 1^{f_M(k)n^c}, c)\). This is computable by a (nonuniform) circuit family of constant depth and size \(f(k)n^O(1)\), where \(f\) is a circuit-computable function. The parameter of the reduced instance is \(f_M(k)n^c/n^c\), or simply \(f_M(k)\), which satisfies the parameter bound required by the definition of paraNC many-one reduction. Therefore we conclude that p-BHP is paraP-complete.

The following problem is a modification of the problem BS-BD-CVP from [11].

Definition 4.10 (p-Small Circuit Evaluation, aka p-SCE).

Instance: Boolean circuit \(C\) on \(n\) inputs, binary string \(x\) of length \(n\), positive integer \(k\), positive integer \(\alpha\), multi-output Boolean circuit \(f\) with size and depth of \(C\) at most \(f(k)n^\alpha\).

Parameter: \(k\)

Question: Does \(C(x) = 1?\)

This theorem is related to [11, Corollary 2], where the authors prove that the BS-BD-CVP problem is complete for the class PNC (a class that exists between paraNC and paraP) under paraNC many-one reductions. While their reduction is a generic reduction, ours is a reduction from the parameterized bounded halting problem.
Theorem 4.11. \( p\text{-SCE} \) is \( \text{paraP-complete} \).

Proof. Membership in \( \text{paraP} \) is straightforward to prove: use the natural algorithm for evaluating a circuit which can be performed in linear time with respect to the size of the circuit. We must also compute \( f(k)n^\alpha \) and compare it with the size and depth of the circuit \( C \). Both of these are polynomial-time algorithms with respect to the size of the input, and hence the problem is in \( \text{paraP} \).

Now we prove \( \text{paraP-hardness} \). The reduction from \( p\text{-BHP} \) is

\[
(M, x, 1^t, c) \mapsto (C_M, x, k, \alpha, f),
\]

where

- \( C_M \) is the standard circuit of size \( O(t^2) \) and depth \( O(t) \) simulating \( t \) steps of the action of \( M \) on inputs of length \( n \),
- \( x \) is copied directly from the input,
- \( k = t/n^\epsilon \),
- \( \alpha = 2c \),
- \( f \) is the function \( x \mapsto x^2 \),

This reduction is computable in the appropriate size and depth bounds, and its correctness follows from the correctness of the standard deterministic Turing machine-to-circuit reduction. To check that the reduced instance is well-formed, let us verify that the circuit \( C_M \) meets the size and depth requirements. The size of \( C_M \) is \( O(t^2) \), which is \( O(((t/n^\epsilon)n^\epsilon)^2) \), or simply \( f(k)n^\alpha \). Similarly the depth of \( C_M \) is \( O(t) \), which is smaller than \( O(t^2) \), and thus bounded above by \( f(k)n^\alpha \) as well. (There are some constants in the size and depth bounds that we have ignored, but those can be incorporated into the definition of \( f \).) Finally, the parameter in the original instance, \( t/n^\epsilon \), is exactly the parameter of the reduced instance, so this reduction meets the necessary parameter bound. Therefore we have shown a correct \( \text{paraNC} \) many-one reduction from a \( \text{paraP-complete} \) problem.

As expected, if any of these \( \text{paraP-complete} \) problems are fixed-parameter parallelizable, then \( \text{paraNC} = \text{paraP} \).

It seems that most \( \text{P-complete} \) problems will end up being \( \text{paraP-complete} \) under this notion of completeness. This doesn’t really help us distinguish between different \( \text{P-complete} \) problems based on how fixed-parameter parallelizable they are. In the next subsection, we try a different approach.

### 4.4 Parameterized complexity of efficient verification

Consider the parameterized weighted circuit satisfiability and circuit evaluation problems. A circuit is \( k\text{-satisfiable} \) if there is a satisfying assignment of Hamming weight exactly \( k \).
Definition 4.12 (p-Circuit $k$-Satisfiability, aka $p$-$k$-CSat).
Instance: Boolean circuit $C$, natural number $k$.
Parameter: $k$.
Question: Is $C$ $k$-satisfiable?

The corresponding parameterized weighted circuit evaluation problem would then be as follows. Let $\|x\|_1$ denote the Hamming weight (that is, the number of ones) in $x$.

Definition 4.13 (p-Circuit $k$-Evaluation, aka $p$-$k$-CE).
Instance: Boolean circuit $C$, binary string $x$, natural number $k$.
Parameter: $k$.
Question: Does $\|x\|_1 = k$ and $C(x) = 1$?

This is a little silly, since the Hamming weight of $x$ can be computed easily (in $\text{NC}^1$ but not in $\text{AC}^0$), but it is technically the “verification” problem corresponding to the satisfiability problem above. Instead, we use a problem that is a little less silly while still equivalent under $\text{paraNC}^1$ many-one reductions.

Instance: Boolean circuit $C$, binary string $x$.
Parameter: $\|x\|_1$.
Question: Does $C(x) = 1$?

In the setting of decision problems, we know that $\text{NP}$ can be characterized as the closure of the circuit satisfiability problem under polynomial-time many-one reductions,

$$\text{NP} = [\text{Circuit Satisfiability}]^{\leq_P}_{m},$$

and $\text{P}$ as the closure of the circuit evaluation problem under $\text{NC}^1$ many-one reductions,

$$\text{P} = [\text{Circuit Evaluation}]^{\leq_{\text{NC}^1}}_{m}.$$  

In the setting of parameterized problems, the class $\text{paraWP}$ can be characterized as the closure of the parameterized weighted circuit satisfiability problem under fixed-parameter tractable many-one reductions,

$$\text{paraWP} = [p$-$k$-$\text{CSat}]^{\leq_{\text{paraP}}}_{m}.$$  

Following the above pattern, we define a new class as the closure of the parameterized weighted circuit evaluation problem under fixed-parameter parallelizable many-one reductions,

$$\text{paraEP} = [p$-$WCE]^{\leq_{\text{paraNC}}}_{m}.$$  

(“E” for evaluation). TODO This is a name conflict with an existing class called $\text{EP}$.

Since the underlying decision problem, the problem of evaluating a circuit on a given input, is in $\text{P}$, the parameterized problem $p$-WCE is trivially in $\text{paraP}$. Since $\text{paraNC}$ reductions compose, $\text{paraNC}$ is a subset of $\text{paraP}$, and $\text{paraP}$ is closed under $\text{paraP}$ reductions, we conclude that $\text{paraEP}$ is a subset of $\text{paraP}$.
Theorem 4.15. \( \text{paraEP} \subseteq \text{paraP} \).

Is \( \text{paraEP} = \text{paraP} \)? The standard simulation of a deterministic Turing machine by a circuit, as in [22], for example, fails to provide a \( \text{paraNC} \) many-one reduction to the parameterized weighted circuit evaluation problem, since the natural reduction would be of the form \( x \mapsto (C, x, \|x\|_1) \), but the parameter value \( \|x\|_1 \) is not necessarily bounded by a function of \( \kappa(x) \). **TODO Show something bad happens if \( \text{paraEP} = \text{paraP} \).**

**TODO Show a problem that is known to be in \( \text{paraP} \) (for example one of the deterministic Turing machine computation problems maybe) is actually in \( \text{paraEP} \).**

The same issue prevents us from showing that \( \text{paraNC} \subseteq \text{paraEP} \).

5 paraNC is to NC as paraWNC is to NNC

The previous section shows one way of proving that a problem is likely not parallelizable, even in a parameterized sense: proving it complete for \( \text{paraP} \). A highly parallel algorithm for such a problem would imply that every problem that could be solved by a fixed-parameter tractable algorithm could be solved by a fixed-parameter parallelizable algorithm, a notion that violates our intuition about the nature of time. There is another way of proving a problem unlikely to be fixed-parameter parallelizable, one that relies on our intuition about the nature of nondeterminism. Our intuition is that nondeterministic computation cannot be simulated deterministically by any algorithm that is significantly more efficient than simply enumerating each possible branch of the nondeterministic computation. We can use this intuition to provide evidence that an entire family of weighted circuit satisfiability problems is unlikely to be parallelizable in a parameterized sense; this section provides formalizes this idea.

Specifically, **Theorem 5.15** proves that \( \text{NC}^d = \text{NNC}^d[\omega(\log n)] \) is equivalent to the corresponding collapse of the classes of parameterized problems. On the way to proving this equivalence, we also prove that the parameterized versions of the natural complete problems in \( \text{NNC}^d[\omega(\log n)] \) are complete for the corresponding parameterized complexity classes (**Theorem 5.10**). Before getting to those theorems, we of course define the necessary parameterized complexity classes and provide some examples of member problems. These complete problems and complexity class collapses validate our intuition that the parameterized complexity classes behave like the classes of problems decidable by algorithms augmented with limited nondeterminism. These results complement similar equivalences from [17, Theorem 3.29] and [12, Theorem 15], as discussed in subsection 5.4 below. We stop short of providing a general theorem that supercedes all of these theorems.

5.1 Definition of paraWNC

The paraW “operator” defined in [15, Definition 3.1] applies generically to an arbitrary complexity class as follows. If \( C \) is a class of decision problems, then
**ParaWC** is the class of parameterized problems \((Q, \kappa)\) for which there is a \(C\) machine \(M\) and computable functions \(f\) and \(h\) such that \(x \in Q\) if and only if there is a string \(w\) of length \(h(\kappa(x)) \log |x|\) such that \(M\) accepts on input \((x, 1^{f(\kappa(x))})\) and nondeterministic input \(w\); access to the nondeterministic input is two-way.

In a loose sense, for most classes \(C\), deterministic \(C\) is to nondeterministic \(C\) as \(\text{para}C\) is to \(\text{paraWC}\).

When \(C = NC\) in particular, we get the following equivalent definition.

**Definition 5.1 (paraWNC\(^d\)).** Let \(d\) be a natural number. A parameterized problem \((Q, \kappa)\) is in the class \(\text{paraWNC}\(^d\)\) if there are circuit-computable functions \(f\) and \(h\), and a nondeterministic circuit family \(\{C_{n,k}\}\) such that for each string \(x\),

- \(x \in Q\) if and only if \(C_{n,k}(x) = 1\), where \(n = |x|\) and \(k = \kappa(x)\),
- \(\text{size}(C_{n,k}) \leq f(k)n^{O(1)}\),
- \(\text{depth}(C_{n,k}) \leq f(k) + \log^d n\),
- \(\text{nondet}(C_{n,k}) \leq h(k) \log n\).

**5.2 Example problems in paraWNC**

What kind of problems are in the class \(\text{paraWNC}\)? Some problems are given in [15], we provide a few more in this section.

**5.2.1 Small generating set problems**

The parameterized semigroup rank problem is in \(\text{paraWNL}\) [15, Theorem 3.12], which is contained in \(\text{paraWNC}\(^2\). Let us consider the parameterized group rank problem, a restriction of the parameterized semigroup rank problem. We can show that this problem is in \(\text{paraWL}\), which is also in \(\text{paraWNC}\(^2\).

**Definition 5.2 (p-Group Rank).**

- **Instance:** finite group \(G\) given as a product table, positive integer \(k\).
- **Parameter:** \(k\).
- **Question:** Does \(G\) have a generating set of cardinality \(k\)?

This problem is highly parallelizable, given that a witness of size \(O(k \log n)\) is provided.

**Theorem 5.3.** \(p\)-Group Rank is in \(\text{paraWL}\).

**Proof.** The Turing machine receives \(G\) and \(k\) as input and a subset \(S\) of \(k\) group elements as a witness. It loops over each element \(g\) in \(G\) and decides whether \(g \in \langle S \rangle\). The machine accepts exactly when all of the subgroup membership tests pass. Looping over \(n\) elements uses \(O(\log n)\) space. Deciding whether \(g\) is in \(\langle S \rangle\) is the subgroup membership problem, which is in \(\text{SL}\), which in turn equals \(L\). Thus the overall space usage is \(O(\log n)\). The size of the witness is \(k \log n\), and two-way access is required, since we execute the subgroup membership procedure \(n\) times. We conclude that this parameterized problem is in \(\text{paraWL}\). \(\square\)
5.2.2 Weighted circuit satisfiability problems

Let $\|x\|_1$ denote the Hamming weight (that is, the number of ones) of a binary string. This problem is the restriction of $p$-CSAT to bounded depth circuits.

**Definition 5.4 ($p$-$k$-NC$^d$CSAT).**

**Instance:** Boolean circuit $C$ on $m$ inputs with depth $\log^d m$, natural number $k$.

**Parameter:** $k$.

**Question:** Is there a binary string $w$ such that $\|w\|_1 = k$ and $C(x) = 1$?

To prove membership in $\text{paraWNC}^d$, we will use the following decoder function. Since we will also later use its inverse, the encoder function, we define it here as well.

**Definition 5.5.** The encoder function, $E_n : \{0, 1\}^m \rightarrow \{0, 1\}^{\log m}$, is defined by

$$E_n(x) = \begin{cases} i & \text{if } x \text{ has exactly one 1 at index } i \\ 0^{\log n} & \text{otherwise.} \end{cases}$$

for each binary string $x$ of length $m$. The decoder function, $D_n$, is defined as the inverse of the encoder function.

**Lemma 5.6.** For each natural number $m$, both the encoder and decoder functions are computable by a circuit of size $O(m)$ and depth $O(\log m)$.

**Proof.** This can be found in, for example, Lemmas 2.5.3 and 2.5.4 of [26]. □

This theorem is an adaptation of [9, Lemma 3.3].

**Lemma 5.7.** For each positive integer $d$, we have $p$-$k$-NC$^d$CSAT is in $\text{paraWNC}^d$.

**Proof.** The algorithm will be composed of several subcircuits.

- Let $U$ be the depth-universal circuit \[13\] for depth $\log^d m$.
- Let $D_{m,k}$ be the function defined by $D_{m,k}(w) = D_m(w_1) \lor \cdots \lor D_m(w_k)$, where $\lor$ denotes bitwise OR for strings of length $m$, the binary string $w_i$ is the $i$th block of $w$ of size $\log m$, and $D_m$ is the decoder function (Definition 5.5).
- Let $\Delta$ be the function that takes $k$ blocks of $\log m$ bits each as input and evaluates to true exactly when each pair of blocks of size $\log m$ are distinct. For example, if $m = 4$, then $\Delta(0101) = 0$ but $\Delta(1011) = 1$.

Define the nonuniform, nondeterministic $\text{NC}^d$ circuit family $\{A_{m,k}\}$ that decides $p$-$k$-NC$^d$CSAT by

$$A_{n,k}((C, k), w) = U(C, D_{m,k}(w)) \land \Delta(w).$$
In other words, $A_{n,k}$ interprets its witness string $w$ of length $k \log m$ as an encoding of a string of length $m$ containing exactly $k$ ones (as enforced by $\Delta$), then evaluates the circuit $C$ on that string.

This algorithm correctly decides the underlying decision problem. For each circuit $C$ of size $n$ and each integer $k$,

$$(C, k) \in p-k-\mathsf{NC}^d\text{-CSAT} \iff \exists x \in \{0,1\}^m : \|x\|_1 = k \text{ and } C(x) = 1$$

$$\iff \exists w \in \{0,1\}^{k \log m} : (C(D_{m,k}(w)) \land \Delta(w)) = 1$$

$$\iff \exists w \in \{0,1\}^{k \log m} : (U(C, (D_{m,k}(w)))) \land \Delta(w)) = 1$$

$$\iff \exists w \in \{0,1\}^{k \log m} : A_{n,k}((C, k), w) = 1.$$

If we assume without loss of generality that $m \leq n$ (by padding with useless gates, for example), then the number of nondeterministic bits used is less than $k \log m$, which is of the form $h(k) \log n$.

For the size and depth bounds, we need to determine the size and depth of the circuits for $U$, $D_{m,k}$, and $\Delta$.

- The depth-universal circuit $U$ has size $m^{O(1)}$ and depth $O(\log^d n)$, which is $O(\log^d m)$ because we can assume without loss of generality that the size of $C$ is at least the number of its inputs.

- Decoding a single block of size $\log m$ requires size $O(m)$ and depth $O(\log m)$ by Lemma 5.6 Decoding all $k$ blocks of size $\log m$ requires $k$ copies of that subcircuit, along with an OR tree for each of the $m$ output bits. Thus the size of $D_{m,k}$ is $O(km + m \log k)$ and the depth $O(\log m + \log k)$.

- Comparing two binary strings of length $\log m$ for inequality requires $O(\log m)$ size and $O(\log \log m)$ depth. Comparing all $\binom{k}{2}$ pairs of blocks and requiring they are all distinct thus requires a circuit of size $O((\binom{k}{2}) \log m)$ and depth $O(\log m + \log k)$.

Hence the overall size of $A_{n,k}$ is of the form $f(k)m^{O(1)}$ and the depth $f(k)+\log^d m$. Since we can assume without loss of generality that $m \leq n$, we get size and depth bounds with the appropriate dependence on $k$ and $n$.

Since we have shown a $\text{paraWNC}^d$ circuit family deciding $p-k-\text{NC}^d\text{-CSAT}$, we conclude that the problem is in $\text{paraWNC}^d$. \hfill $\square$

If we replace $k$ with any function $i(n)$ bounded above by a polynomial in $n$, we get the (already known) membership of the underlying decision problem in $\text{NNC}^d[i(n) \log n]$. The weight $k$ can be at most the number of inputs to the circuit, which is in turn at most the size of the circuit, so a polynomial upper bound on $k$ suffices to cover all meaningful values of $k$.

**Corollary 5.8** ([9]). Suppose $d$ is a positive integer and $i$ is a circuit-computable function such that $i(n) \leq n^{O(1)}$. Then the decision problem $i(n)-\text{NC}^d\text{-CSAT}$ is in nonuniform $\text{NNC}^d[i(n) \log n]$. 

19
Proof. The only change we need to make the circuit from the proof of the previous lemma is to add a single AND with the result of deciding the inequality \( k \leq i(n) \). The computation of \( i(n) \) can be assumed part of the nonuniformity of the circuit, at which point the comparison requires only logarithmic depth. (If we wish the deciding circuit to be uniform, we can require that \( i(n) \) be computable in uniform depth \( O(\log^d n) \).)

5.2.3 Weighted formula satisfiability problems

This technique also provides an alternate proof that parameterized weighted Boolean formula satisfiability problem, denoted \( p\text{-FSat} \), is in \( \text{paraWNC}^1 \), implicit in [15, Theorem 3.6]. The proof directly implements the decoder function of Definition 5.5 as a Boolean formula.

Lemma 5.9 ([15, Theorem 3.6]). \( p\text{-FSat} \) is in \( \text{paraWNC}^1 \).

Proof. We adapt the “\( k \log n \) trick” from circuit inputs to formula variables. This requires a reimplementation of the decoder function of Definition 5.5 as a function on Boolean variables. Similar to the proof of Lemma 5.7, the algorithm involves composing a decoder and an algorithm for evaluating a Boolean formula, after nondeterministically choosing a witness. Again, we use some subcircuits:

- Let \( U \) be the \( \text{NC}^1 \) algorithm for evaluating a Boolean formula [6, 7].
- Let \( M_i \) be the function on \( \log m \) that outputs the \( i \)th minterm of its input variables (for example, \( M_4(w_1, w_2, w_3) = w_1 \land \neg w_2 \land \neg w_3 \)). This function acts like the decoder in Lemma 5.7.
- Let \( R_{m,k}(\phi) \) be the function that replaces each instance of a variable \( x_i \) in a Boolean formula \( \phi \) on \( m \) variables with \( \bigvee_{j=1}^k M_i(\vec{v}_j) \), where \( \vec{v}_j \) denotes the \( j \)th block of size \( \log m \) in a tuple of \( k \log m \) new variables \( v_1, \ldots, v_{k \log m} \).
- Let \( \Delta \) be the function that takes \( k \) blocks of \( \log m \) bits each as input and evaluates to true exactly when each pair of blocks of size \( \log m \) are distinct.

Define the nonuniform, nondeterministic \( \text{NC}^1 \) circuit family \( \{A_{n,k}\} \) that decides \( p\text{-FSat} \) by

\[
A_{n,k}(\phi, k, w) = U(R_{m,k}(\phi), w) \land \Delta(w).
\]

In other words, \( A_{n,k} \) interprets its witness string \( w \) of length \( k \log m \) as the encoding of an assignment to the variables of \( \phi \) in which exactly \( k \) variables are set to true, then evaluates the formula \( \phi \) with respect to the decoded assignment.

This algorithm correctly decides the weighted Boolean formula satisfiability problem. Similar to the proof of Lemma 5.7, \( \phi \) has a satisfying assignment of weight exactly \( k \) if and only if \( \phi' \) has a satisfying assignment (of arbitrary weight). If we assume without loss of generality that \( m \leq n \) (by padding with tautological conjuncts, for example), then the number of nondeterministic bits used is less than \( k \log m \), which is of the form \( h(k) \log n \).
The algorithm also has the appropriate size and depth bounds. Since each minterm is of size exactly $\log m$ and each variable $x_i$ is represented the disjunction of $k$ such minterms, the new formula $\phi'$ is of size $|\phi|k\log m$, which is just $nk\log m$.

- Each variable $x_i$ can be replaced in parallel, and within that replacement, each disjunct $M_i(\vec{w}_j)$ can be replaced in parallel as well. The circuit that writes $M_i(\vec{w}_j)$ is a circuit of size $O(\log m)$ and depth $O(\log m)$ (for using the index $i$ as a selector in a multiplexer), so $R_{m,k}$ can be implemented by a circuit of size $O(nk\log m)$ and depth $O(\log m)$.

- The circuit $U$ receives a formula of size $O(nk\log m)$ and an assignment of length $O(k\log m)$. Since $U$ is a circuit of size polynomial in its the size of its input, its size is $O((nk\log m)^c)$ for some constant $c$. Similarly, its depth is $O(\log^d(nk\log m))$.

- As in Lemma 5.7, the circuit for $\Delta$ is of size $O(k^2\log m)$ and depth $O(\log m + \log k)$.

The overall size and depth of the circuit $A_{n,k}$ are therefore of the form $f(k)n^{O(1)}$ and $f(k) + O(\log^d n)$, respectively.

At this point, we have shown a correct $\text{paraWNC}^1$ algorithm for $p$-FSat. \Box

The same proof works for the problems of deciding whether a circuit or a formula has a satisfying assignment of weight at most $k$, as well (one could even remove the $\Delta$ subcircuit entirely, but that is not necessary).

5.3 Completeness in $\text{paraWNC}$

We saw that the parameterized weighted Boolean formula satisfiability problem, $p$-FSAT, is in $\text{paraWNC}^1$ in Lemma 5.9. In fact, it is complete for $\text{paraWNC}^1$ under $\text{paraFO}$ many-one reductions [15, Theorem 3.6]. TODO Can we use the “template on top of a verification language” to prove completeness in complexity classes of decision problems with limited nondeterminism as well? The parameterized weighted Boolean circuit satisfiability problem, denoted $p$-CSAT, is the same problem with Boolean circuits instead of Boolean formulas; this problem is complete for $\text{paraWP}$ under $\text{paraFO}$ many-one reductions by a similar proof. It makes sense to expect, then, that the problems $p$-$k$-$\text{NC}^d$CSAT may be complete for the classes $\text{paraWNC}^d$, for each positive integer $d$.

In order to prove completeness for these problems, we use the strategy from [9, Theorem 3.6], which relies on the “$k\log n$ trick” (see [17] Corollary 3.13), or the origin [1]. There the authors prove that the decision problem underlying $p$-$k$-$\text{NC}^d$CSAT is complete for $\text{NNC}^d[k\log n]$. As we will see in the next section (and repeated throughout this paper), parameterized complexity and limited nondeterminism in decision complexity are closely related.

Theorem 5.10. For each positive integer $d$, we have $p$-$k$-$\text{NC}^d$CSAT is complete for $\text{paraWNC}^d$ under $\text{paraNC}^1$ many-one reductions.
Proof. Membership in $\text{paraWNC}^d$ was proven in Lemma 5.7.

Suppose $(Q, \kappa)$ is in $\text{paraWNC}^d$, so there is a nonuniform $\text{NC}^d$ circuit family $\{C_{n,k}\}$ and circuit-computable functions $f$ and $h$ such that

- $x \in Q$ if and only if $C_{n,k}(x) = 1$,
- $\text{size}(C_{n,k}) \leq f(k)n^{O(1)}$,
- $\text{depth}(C_{n,k}) \leq f(k) + O(\log^d n)$,
- $\text{nondet}(C_{n,k}) \leq h(k) \log n$.

On input $x$ of length $n$, let $C_x$ denote the circuit $C_{n,k}$ with $x$ hardcoded as its first $n$ inputs. Thus $C_x$ is a circuit with $h(k) \log n$ inputs such that $x \in Q$ if and only if $C_x$ is satisfiable. Let $E_{n,k}$ denote the function defined by $E_{n,k}(w) = E_n(w_1) \circ \cdots \circ E_n(w_{h(k)})$, where $\circ$ denotes string concatenation, $E_n$ is the encoder function of Definition 5.5 and $w_i$ is the $i$th block of size $n$ in the string $w$, for each string $w$ of length $h(k)n$. The reduction is then $x \mapsto (C', h(k))$, where $C' = C_x \circ E_{n,k}$.

The circuit $C'$ is of the correct form to be an input to $p$-$k$-$\text{NC}^d$-$\text{CSAT}$. The size of $C'$ is

$$\text{size}(C') = \text{size}(E_{n,k}) + \text{size}(C_x) \leq h(k)\text{size}(E_n) + f(k)n^{O(1)} \leq h(k)O(n) + f(k)n^{O(1)}$$

and the depth

$$\text{depth}(C') = \text{depth}(E_{n,k}) + \text{depth}(C_x) \leq \text{depth}(E_n) + (f(k) + \log^d n) \leq O(\log n) + f(k) + \log^d n \leq f(k) + O(\log^d n).$$

The number of inputs to $C'$ is $h(k)n$, so we need the size to be polynomial in $h(k)n$ and the depth to be $\log^d(h(k)n)$. The size bound is satisfied if we choose $h(k)$ so that $f(k) \leq h(k)$ and the depth bound satisfied if we choose $h(k)$ so that $f(k) \leq \log^d h(k)$, ignoring some constants that can be incorporated into the function $h$. We can choose $h$ this way without loss of generality, because choosing a larger $h$ does not affect membership of $(Q, \kappa)$ in $\text{paraWNC}^d$. (This does, however, cause the nondeterminism upper bound of the problem $(Q, \kappa)$ to be extremely loose and the dependence on $k$ in the size and depth bounds of $C'$ to be extremely high, but it is technically sufficient.)

The reduction is a correct many-one reduction between the underlying decision problems. Suppose $x \in Q$, so $C_x$ is satisfiable. Since $E_n$ is surjective, so is $E_{n,k}$, hence there is a string $w$ (of length $h(k)n$) such that $C'(w) = 1$. Furthermore,
this string satisfies $\|w\|_1 = h(k)$ since all preimages of $E_{n,k}$ satisfy this equality. Therefore, $C'$ has a satisfying assignment of Hamming weight exactly $h(k)$. For the converse, suppose $C'$ has a satisfying assignment $w$ of weight exactly $h(k)$. Then there is a satisfying assignment of length $h(k) \log n$ for $C_x$, namely $E_{n,k}(w)$. Therefore, $x$ is in $Q$.

The reduction is $\text{paraNC}^1$-computable. The size of the circuit computing the reduction is simply the size of the output, which is $\text{size}(C') + \text{size}(h_k)$, where $h_k$ denotes the circuit computing $h$ on inputs of size $k$. Both addends are of the form $f'(k)n^{O(1)}$ for some circuit-computable function $f'$. The depth of the circuit computing the reduction is dominated by the depth of the $h_k$, which is bounded above by $f'(k) + \log n$ for some function $f'$. Thus the size and depth requirements for the reduction are met. Finally, if $R$ denotes the reduction and $\kappa'$ denotes the parameterization for $p$-$k$-$\text{NC}^d$-$\text{CSat}$,

$$\kappa'(R(x)) \leq \kappa'((C', h(\kappa(x)))) = h(\kappa(x)).$$

Since $h$ is circuit-computable by hypothesis, the reduction meets the parameterization bound.

Since we have shown a correct $\text{paraNC}^1$ many-one reduction from an arbitrary parameterized problem in $\text{paraWNC}^d$ to $p$-$k$-$\text{NC}^d$-$\text{CSat}$, we conclude that $p$-$k$-$\text{NC}^d$-$\text{CSat}$ is complete for the class. $\square$

As in Corollary 5.8 if we replace $k$ with any function $i(n)$ bounded above by a polynomial in $n$, we get the (already known) completeness of the decision problem $i(n)$-$\text{NC}^d$-$\text{CSat}$ in the complexity class $\text{NNC}^d[i(n)\log n]$. This is a slight improvement, since [9] Theorem 3.6 proves that the problem of deciding whether a circuit has a satisfying assignment of weight at most $i(n)$ is complete for $\text{NNC}^d[i(n)\log n]$ for all $d \geq 2$ under logarithmic space many-one reductions.

**Corollary 5.11** ([9, Theorem 3.6]). Suppose $d$ is a positive integer and $i$ is a circuit-computable function such that $i(n) \leq n^{O(1)}$. Then $i(n)$-$\text{NC}^d$-$\text{CSat}$ is complete for $\text{NNC}^d[i(n)\log n]$ under nonuniform $\text{NC}^1$ many-one reductions.

**Proof.** Membership was proven in Corollary 5.8. The proof of completeness is identical to that of the previous theorem, replacing $k$ with $i(n)$. (If we wish to have uniform completeness, we can require that $i(n)$ be computable in uniform depth $O(\log n)$.) $\square$

This technique also gives us an alternate proof of the completeness $p$-$\text{FSat}$ [15, Theorem 3.6]. However, we can prove only completeness under the $\text{paraNC}^1$ reduction instead of the tighter $\text{paraFO}$ reduction.

**Theorem 5.12.** $p$-$\text{FSat}$ is complete for $\text{paraWNC}^1$ under $\text{paraNC}^1$ many-one reductions.

---

1Technically, we need to guarantee the satisfying input to $C_x$ has no all-zero blocks to make this statement. For each parameterized problem $(Q, \kappa)$ there is another equivalent problem that satisfies such a requirement on the witnesses for $Q$ with no change in complexity.
Proof. Membership in $\text{paraWNC}^1$ was proven in Lemma 5.9.

To prove hardness, it suffices to show a $\text{paraNC}^1$ reduction from $p$-$k$-$\text{NC}^1$-$\text{CSAT}$ to $p$-$\text{FSAT}$, since the circuit problem is complete for $\text{paraWNC}^1$ by the previous theorem. The reduction is $(C, k) \mapsto (\phi, k)$, where $\phi$ is the standard reduction for logarithmic depth circuits to polynomial size Boolean formulas. This reduction recursively replaces each subcircuit with its corresponding formula, using the input gates of the circuit as the variables for the formula. Since the depth of the (fan-in 2) circuit is $O(\log n)$, where $n$ is the number of inputs to $C$, the size of the formula is $2^{O(\log n)}$, which is polynomial in $n$. Furthermore, since the variables appearing in the formula are exactly the input gates of the circuit, the circuit has a satisfying assignment of weight $k$ if and only if the formula has a satisfying assignment of weight $k$. Finally, the reduction is computable by circuit of size $O(|\phi|)$ and depth $O(\text{depth}(C))$, so it is computable in $\text{NC}^1$. TODO someone verify this. Since furthermore the parameter is unchanged by the reduction, the $\text{NC}^1$-computable function trivially induces a $\text{paraNC}^1$ reduction. 

Now we have a full interpolation for the inclusion chain

$$\text{paraWNC}^1 \subseteq \text{paraWNC}^2 \subseteq \cdots \subseteq \text{paraWP},$$

via the chain of parameterized reductions between corresponding complete problems

$$p$-$\text{FSAT} \leq p$-$k$-$\text{NC}^d$-$\text{CSAT} \leq \cdots \leq p$-$\text{CSAT}.$$ 

Whether the $p$-$k$-$\text{NC}^d$-$\text{CSAT}$ problems are complete under $\text{paraFO}$ many-one reductions remains open.

Finally, we extend [15, Corollary 3.7] using this new family of complete problems. That theorem states that $\text{NC}^1 = \text{P}$ implies $W[\text{SAT}] = W[\text{P}] (= \text{paraWP})$, so we wish to generalize it to allow for an interpolation between $\text{NC}^1$ and $\text{P}$. This connects a collapse in parameterized complexity classes to one in classical complexity classes.

**Corollary 5.13.** For each positive integer $d$, if $\text{NC}^d = \text{P}$, then $W[\text{NC}^d$-$\text{SAT}] = W[\text{P}] (= \text{paraWP}).$

**Proof.** By definition, $W[\text{NC}^d$-$\text{SAT}]$ is the closure of $p$-$k$-$\text{NC}^d$-$\text{CSAT}$ under $\text{paraP}$ many-one reductions and $W[\text{P}]$ is the closure of $p$-$\text{CSAT}$ under $\text{paraP}$ many-one reductions. If $\text{NC}^d = \text{P}$, then there is a $\text{paraP}$ many-one reduction from $p$-$\text{CSAT}$ to $p$-$k$-$\text{NC}^d$-$\text{CSAT}$ (because $p$-$\text{CSAT}$ is now in $\text{NC}^d$). Thus every parameterized problem that reduces to $p$-$\text{CSAT}$ also reduces to $p$-$k$-$\text{NC}^d$-$\text{CSAT}$. We conclude that $W[\text{NC}^d$-$\text{SAT}] = W[\text{P}].$ 

### 5.4 Does paraNC equal paraWNC?

The main theorem in this section, Theorem 5.15, provides evidence that the parameterized complexity classes $\text{paraNC}$ and $\text{paraWNC}$ are distinct. Compare it with the following similar theorems (here $\beta$ indicates one-way access to the witness as opposed to $W$, which indicates two-way access).
• \( \text{paraP} = \text{paraWP} \) if and only if \( P = \text{NP}[\omega(\log n)] \) [17, Theorem 3.29].

• \( \text{paraL} = \text{paraβL} \) if and only if \( L = \text{NL}[\omega(\log n)] \) [12, Theorem 15].

**TODO Can we generalize all of these theorems?**

This lemma is an adaptation of [17, Lemma 3.24]. It is the “opposite” of Lemma 3.8.

**Lemma 5.14.** Suppose \( d \) is a positive integer. For each increasing, circuit-computable function \( f \), there is a function \( i_{f,d} \) such that

\[
f(i_{f,d}(n)) \leq \min(n, \log^d n)
\]

for each \( n \geq f(1) \). Furthermore, \( i_{f,d} \) is nondecreasing, unbounded, and circuit-computable. (We call \( i_{f,d} \) the “lower inverse” of \( f \).)

**Proof.** Define \( i_{f,d} \) by

\[
i_{f,d}(n) = \max\{j \in \mathbb{N} \mid f(j) \leq \min(n, \log^d n)\}.
\]

For the boundary case where \( n \) is smaller than \( f(1) \), define \( i_{f,d}(n) = 1 \). It is straightforward to prove that this function is nondecreasing and unbounded. (The integer \( d \) must be greater than zero to guarantee \( i_{f,d} \) is unbounded.)

To compute \( i_{f,d} \), we use the fact that \( f \) is increasing to perform a binary search on the values \( f(1), \ldots, f(\min(n, \log^d n)) \) to determine the largest \( j \) such that \( f(j) \leq \min(n, \log^d n) \). In the worst case, there will be at most \( \log n \) comparison subcircuits, each requiring a computation of \( f \), so the overall depth of the circuit computing \( i_{f,d} \) is \( O(\text{depth}(f) \log n) \) and the size is \( O(\text{size}(f) \log n) \). \( \square \)

This theorem is an adaptation of [17, Theorem 3.29].

**Theorem 5.15.** Suppose \( d \) is a positive integer. \( \text{paraNC}^d = \text{paraWNC}^d \) if and only if there is a circuit-computable, nondecreasing, unbounded function \( i \) such that \( \text{NC}^d = \text{NNC}^{[i(n) \log n]} \).

**Proof.** First we prove the reverse implication. Assume \( \text{NC}^d = \text{NNC}^{[i(n) \log n]} \), for some \( i \). Since \( p-k \text{-} \text{NC}^d \text{-CSAT} \) is complete for \( \text{paraWNC}^d \) by Theorem 5.10, it suffices to show that this problem is in \( \text{paraNC}^d \), which necessitates the claimed collapse. By Corollary 5.11, the problem \( i(n) \text{-} \text{NC}^d \text{-CSAT} \) is in \( \text{NNC}^{[i(n) \log n]} \), which means it is also in \( \text{NC}^d \) by assumption. If \( (Q, \kappa) \) denotes \( p-k \text{-} \text{NC}^d \text{-CSAT} \) and \( i(n)-Q \) denotes \( i(n) \text{-} \text{NC}^d \text{-CSAT} \), then the former is in \( \text{paraNC}^d \) by Corollary 3.11.

Now we prove the forward implication. Assume \( \text{paraNC}^d = \text{paraWNC}^d \). By Theorem 5.10, the parameterized problem \( p-k \text{-} \text{NC}^d \text{-CSAT} \) is in \( \text{paraWNC}^d \), which means it is also in \( \text{paraNC}^d \) by assumption. Suppose the circuit family witnessing its membership in \( \text{paraNC}^d \) has size \( f(k)n^{O(1)} \) and depth \( f(k) + \log^d n \) for some circuit-computable function \( f \). Assume without loss of generality that \( f \) is increasing. Choose \( i \) to be the “lower inverse” function \( i_{f,d} \) guaranteed by Lemma 5.14. Since \( i(n) \text{-} \text{NC}^d \text{-CSAT} \) is complete for \( \text{NNC}^{[i(n) \log n]} \) by Corollary 5.11, it suffices to show that this problem is in \( \text{NC}^d \), which necessitates the claimed collapse. If \( (Q, \kappa) \) denotes \( p-k \text{-} \text{NC}^d \text{-CSAT} \) and \( i(n)-Q \) denotes \( i(n) \text{-} \text{NC}^d \text{-CSAT} \), then the latter is in \( \text{NC}^d \) by Corollary 3.13. \( \square \)
As a corollary, we can relate the completeness of parameterized or decision problems to collapses in decision or parameterized classes, respectively.

**Corollary 5.16.** Suppose $d$ is a positive integer. The following are equivalent.

1. $p$-$k$-$\text{NC}^d\text{CSAT}$ is in $\text{paraNC}^d$.
2. There is a circuit-computable, nondecreasing, unbounded function $i$ such that $i(n)$-$\text{NC}^d\text{-CSAT}$ is in $\text{NC}^d$.
3. $\text{paraNC}^d = \text{paraWNC}^d$.
4. There is a circuit-computable, nondecreasing, unbounded function $i$ such that $\text{NC}^d = \text{NNC}^d[i(n) \log n]$.

### 5.5 **Is paraWNC in paraP?**

The previous section demonstrates that it is unlikely that $\text{paraWNC}$ is contained in $\text{paraNC}$. By relaxing the collapses in Theorem 5.15, we show that it is also unlikely that $\text{paraWNC}$ is contained in $\text{paraP}$. However, the collapses in this theorem are weaker.

**Theorem 5.17.** Suppose $d$ is a positive integer. $\text{paraWNC}^d \subseteq \text{paraP}$ if and only if there is a circuit-computable, nondecreasing, unbounded function $i$ such that $\text{NNC}^d[i(n) \log n] \subseteq \text{P}$.

**Proof.** The proof is identical to that of Theorem 5.15, replacing $\text{paraNC}^d$ and $\text{NC}^d$ with $\text{paraP}$ and $\text{P}$, respectively. It uses versions of Corollary 3.11 and Corollary 3.13 with similar changes.

An alternate approach is to demonstrate related collapses in parameterized complexity classes. We extend [15, Corollary 3.8], which states that $\text{paraWNC}^1 \subseteq \text{paraP}$ if and only if $\text{paraP} = W[\text{SAT}]$. As before, we wish to generalize this equivalence to allow for an interpolation between $\text{NC}^1$ and $\text{P}$.

**Theorem 5.18.** Suppose $d$ is a positive integer. $\text{paraWNC}^d \subseteq \text{paraP}$ if and only if $\text{paraP} = W[\text{NC}^d\text{SAT}]$.

**Proof.** If $\text{paraWNC}^d \subseteq \text{paraP}$, then $p$-$k$-$\text{NC}^d\text{CSAT}$ is in $\text{paraP}$, so the closure of $p$-$k$-$\text{NC}^d\text{CSAT}$ under $\text{paraP}$ many-one reductions is contained in the closure of $\text{paraP}$ under the same reductions, which is just $\text{paraP}$. Thus $\text{paraP} = W[\text{NC}^d\text{SAT}]$.

If $\text{paraP} = W[\text{NC}^d\text{SAT}]$, then every parameterized problem that reduces to $p$-$k$-$\text{NC}^d\text{CSAT}$ under $\text{paraP}$ many-one reductions is in $\text{paraP}$. Since $p$-$k$-$\text{NC}^d\text{CSAT}$ is complete for $\text{paraWNC}^d$ under $\text{paraNC}^d$ many-one reductions, every problem in $\text{paraWNC}^d$ reduces to $p$-$k$-$\text{NC}^d\text{CSAT}$ under $\text{paraNC}^d$ many-one reductions, and hence under $\text{paraP}$ many-one reductions as well. Thus $\text{paraWNC}^d \subseteq \text{paraP}$.

26
5.6 Does paraWNC equal paraWP?

**Conjecture 5.19.** Suppose $d$ is a positive integer. \( \text{paraWNC}^d = \text{paraWP} \) if and only if there is a nondecreasing, unbounded, circuit-computable function $i$ such that $\text{NC}^d[i(n) \log n] = \text{NP}[i(n) \log n]$.

**Justification.** It seems that

\[
p-k\text{-CSAT} \leq^m_{\text{paraNC}^d} p-k\text{-NC}^d\text{CSAT}
\]

should imply

\[
i(n)\text{-CSAT} \leq^m_{\text{NC}^d} i(n)\text{-NC}^d\text{-CSAT}.
\]

However, the problem is the nondeterminism bounds: on one side, the bound is $f(k) \log n$ and on the other $i(n) \log n$.

**6 The paraWNC hierarchy**

For the necessary background in logic, see [17, Chapter 4].

**6.1 Definition of paraWNC\([t]\)**

Let $\Phi$ be a class of formulas and let $\phi$ be an element of $\Phi$ with one free relation variable $X$ of arity $s$. Define the parameterized weighted definability problem as follows.

**Definition 6.1** ($p$-WD$_{\phi}$ [17, Section 5.1]).

- **Instance:** structure $A$, natural number $k$.
- **Parameter:** $k$
- **Question:** Is there an $S \subseteq A^s$ such that $|S| = k$ and $A \models \phi(S)$?

Let $p$-WD-$\Phi$ be the class of all problems $p$-WD$_{\phi}$ where $\phi$ is in $\Phi$.

**TODO** Using closure under paraNC$^1$ reductions just gives the paraWP$[t]$ classes, since paraWP$[t]$ is probably just the closure under first-order reductions. We may have to use a weft-based definition here.

**6.2 Example problem in paraWNC$[t]$**

**TODO** Show a natural problem in some $\text{paraWNC}[t]$, perhaps a weft-restricted $\text{NC}^d$ circuit satisfiability problem.

In subsection 5.2 we proved that $p$-GROUP RANK is in paraWNC$^2$. This problem may not be in paraWNC$^2[t]$ for some finite $t$. Unfortunately, the best first-order formulas that describe this problem are not of the form $\Sigma_k$ for some finite natural number $k$. In general, the number of variables grows with the order of the group.
There is a first-order formula that has $O(1)$ variables, but $O(\log n)$ quantifier alternations, using the strategy from the PSPACE-completeness of TQBF [25, Lemma 2.3]. This places the problem in $\text{FO}[\log n]$, which is $\text{AC}^1$, but doesn’t provide placement in any finite level of the paraWNC hierarchy.

The authors of [4] were not able to show membership in $\text{FO}[\log \log n]$, which would have slightly improved the above membership, but still wouldn’t place the problem in a finite level of the paraWNC hierarchy.

There is a first-order formula that has two alternations (beginning with $\forall$), but $O(\log n)$ variables [25, Lemma 3.5]. The fact that the number of variables grows with the size of the input means this does not place the problem in any descriptive complexity class, nor any parameterized complexity class.

There is a $\text{FO}[\text{DTC}]$ formula for the problem, since group membership is in L, which equals $\text{FO}[\text{DTC}]$. Although this subsumes the membership in $\text{FO}[\log n]$ given in the first item above, this does not help us place the problem in a parameterized complexity class.

This basically means that the problem is in a class like paraWNC$^2[\log n]$, and any improvement would likely follow an improvement of the descriptive complexity of the subgroup membership problem.

6.3 Completeness in paraWNC$[t]$  

TODO If we show a complete problem here, make sure to give a corollary in the section below that it is not in paraNC unless something bad happens.

6.4 Does paraNC equal paraWNC$[t]$?

**Theorem 6.2.** Suppose $d$ is a positive integer, $t$ is a positive integer greater than one, and $i$ is a circuit-computable nondecreasing function. $\Pi_t \text{CSAT}$ is complete for $\text{GC}[i(n) \log n, \Pi_t]$ under $\text{NC}^d$ many-one reductions.

**Proof.** TODO Fill me in... □

**Theorem 6.3.** Suppose $d$ is a positive integer and $t$ is a positive integer greater than one. $p-\Pi_t \text{FSAT}$ is complete for paraWNC$^d[t]$ under paraNC$^d$ many-one reductions.

**Proof.** TODO Fill me in... □

Since we will construct reductions between these problems, we need an efficient and highly parallel algorithm for transforming a circuit into an equivalent formula. The input variables must be identical in order to guarantee that the parameter values are identical.
Lemma 6.4. Suppose $t$ is a positive integer. There is an $\text{NC}^2$ many-one reduction from $\Pi_t\text{CSat}$ to $\Pi_t\text{FSat}$. Furthermore, the reduction preserves witnesses in the following (strong) sense. For each circuit $C$, if $\phi$ is the image of $C$ under the reduction, then $\text{Var}(C) = \text{Var}(\phi)$ and for each input vector $x$, we have $C(x) = 1$ if and only if $\phi(x) = 1$.

Proof. The reduction operates as follows on input $(C, k)$, where $C$ is a Boolean circuit of size $m$ and depth $t$ and $k$ is a positive integer. Construct a trie (also known as a prefix tree) from the $O(m^t)$ possible paths from output gate to input gate, then output $(\phi, k)$, where $\phi$ is the Boolean formula represented by the constructed trie. The output gate is the root of the trie, the internal gates are the internal nodes of the tree, and the input gates are the leaf nodes of the trie (one leaf node for each input gate). By construction, the formula has the same set of variables as the circuit, and any path from output gate to input gate in the circuit has the same labels as the corresponding path in the trie, so for any input $x$, we have $C(x) = \phi(x)$. This also implies that $C$ has a satisfying assignment of weight $k$ if and only if $\phi$ has a satisfying assignment of weight $k$.

Constructing a trie from $O(m^t)$ binary strings can be done by a circuit with $O(\log m^t)$ time and $O(m^2)$ size [23] (TODO someone needs to verify this). Since each “character” in our “strings” is really an element of $\{1, \ldots, m\}$, there is an extra $O(\log m)$ depth penalty for reducing an alphabet of size $m$ to the binary alphabet. Thus the overall depth is $O(t \log^2 m)$ and the size $O(m^{2t} \log m)$. Since $t$ is a constant with respect to the size of the input $C$, this is an $\text{NC}^2$ circuit.

The following theorem is an adaptation of [8, Theorem 4.3]. It is a translation of Theorem 5.15 to the finite levels of the $\text{WP}$ hierarchy:

- the collapse $\text{paraNC}^d = \text{paraWNC}^d[\tau]$ is weaker than $\text{paraNC}^d = \text{paraWNC}^d$,
- the inclusion $\text{GC}[i(n) \log n, \Pi_t\text{LOGTIME}] \subseteq \text{NC}^d$ is weaker than the collapse $\text{NC}^d = \text{NNC}^d[i(n) \log n]$.

$\Pi_t\text{LOGTIME}$ is a subclass of $\text{LH}$, the logarithmic time hierarchy, which equals $\text{AC}^0$ [20, Corollary 5.32]. Since $\text{AC}^0$ is a strict subset of $\text{NC}^1$ [18], $\Pi_t\text{LOGTIME}$ is a strict subclass of $\text{NC}^1$. Although, $\Pi_t\text{LOGTIME}$ is strictly weaker than $\text{NC}^1$, the addition of $\omega(\log n)$ nondeterministic bits seems to give it power beyond that of $\text{NC}$, which seems able to simulate only $O(\log n)$ bits (by enumerating each string of that length in parallel). This theorem suggests that a collapse of the parameterized complexity classes yields a deterministic simulation of $\omega(\log n)$ bits, which would violate our intuition of nondeterminism. See similar comments after [8, Theorem 4.3] and compare with the conclusion of [Theorem 5.17].

Theorem 6.5. Suppose $d$ and $t$ are positive integers with $t$ greater than one. $\text{paraNC}^d = \text{paraWNC}^d[\tau]$ if and only if there is a circuit-computable, unbounded, nondecreasing function $i$ such that $\text{GC}[i(n) \log n, \Pi_t\text{LOGTIME}] \subseteq \text{NC}^d$. 

29
Proof. First, we prove the reverse implication. Assume there is a function \( i \) such that \( GC[i(n) \log n, \Pi_t \text{LOGTIME}] \subseteq NC^d \). Since \( \Pi_t \text{CSAT} \) is in the class \( GC[i(n) \log n, \Pi_t \text{LOGTIME}] \) by Theorem 6.2, it is now in \( NC^d \) as well.

Since \( p-\Pi_t \text{FSAT} \) is complete for \( \text{paraWNC}^d[t] \) under \( \text{paraNC}^d \) many-one reductions by Theorem 6.3, it suffices to show that \( p-\Pi_t \text{FSAT} \) is in \( \text{paraNC}^d \). Furthermore, by Lemma 3.10, it suffices to show that there is a small parameter \( NC^d \) many-one reduction from \( \Pi_t \text{FSAT} \) to \( \Pi_t \text{CSAT} \).

Let \( \{R_n\} \) be the family of circuits computing the function \( \phi \mapsto C_\phi \), where \( \phi \) is a \( \Pi_t \)-normalized Boolean formula and \( C_\phi \) is the natural \( \Pi_t \) Boolean circuit induced by that formula. If we can show that \( R_n \) is in \( NC^d \), then the parameter upper bound \( i \) and \( \{R_n\} \) together comprise a small parameter \( NC^d \) many-one reduction from \( p-\Pi_t \text{FSAT} \) to \( \Pi_t \text{CSAT} \).

Transforming a formula to its equivalent circuit is certainly computable in logarithmic space, so it is certainly in \( NC^2 \). However, we can transform a Boolean formula into an equivalent Polish notation (prefix) Boolean formula in alternating logarithmic time [6], which is in \( NC^1 \), and from there we can write the adjacency matrix of the tree given by the prefix Boolean formula with an \( NC^1 \) algorithm.

Now we prove the forward implication. Assume \( \text{paraNC}^d = \text{paraWNC}^d[t] \). Since \( p-\Pi_t \text{FSAT} \) is in \( \text{paraWNC}^d[t] \) by Theorem 6.3, it is now in \( \text{paraNC}^d \) as well. We will use Lemma 3.12 to show a reduction meeting the criteria of that lemma from \( \Pi_t \text{CSAT} \), which is complete for the class \( GC[i(n) \log n, \Pi_t \text{LOGTIME}] \), to \( p-\Pi_t \text{FSAT} \). TODO Move that lemma down here, since it’s only used here now?

Let \( \{C_{m,k'}\} \) be the deterministic circuit family deciding \( p-\Pi_t \text{FSAT} \) and \( f \) the circuit-computable function such that

- for each \( \phi \), we have \( \phi \) is satisfiable if and only \( C_{m,k'}(\phi) = 1 \)
- \( \text{size}(C_{m,k'}) \leq f(k')m^{O(1)} \)
- \( \text{depth}(C_{m,k'}) \leq f(k') + O(\log^d m) \).

Assume without loss of generality that \( f \) is increasing. Choose \( i \) to be the “lower inverse” function \( i_{f,d} \) guaranteed by Lemma 5.14.

For the chosen function \( i \), consider the problem \( \Pi_t \text{CSAT} \). By Lemma 6.4, there is an \( NC^d \) reduction, \( \{R_n\} \), from \( \Pi_t \text{CSAT} \) to \( \Pi_t \text{FSAT} \), the decision problem underlying the parameterized problem \( p-\Pi_t \text{FSAT} \). Furthermore, since the variables and the satisfying assignments are identical, we have

\[
f(k'(R_n(C,k))) = f(k')f(\phi,k) \leq f(k) \leq f(i(n)) \leq f(i_{f,d}(n)) \leq \min(n, \log^d n).
\]
Thus the many-one reduction from Lemma 6.4 and the paraNC\textsuperscript{d} algorithm for $p$-$\Pi_t$FSAT meet the conditions in the premise of Lemma 3.12. We conclude that $\Pi_t$CSAT is in NC\textsuperscript{d}.

6.5 Is paraWNC\textsuperscript{[t]} in paraP?

TODO Translate everything from subsection 5.5.

6.6 Does paraWNC\textsuperscript{[t]} equal paraWP\textsuperscript{[t]}?

TODO Translate everything from subsection 5.6.

References


31


32


