# Provability of the Circuit Size Hierarchy and Its Consequences

**Marco Carmosino**
IBM Research, Cambridge, MA, USA

**Valentine Kabanets**
Simon Fraser University, Burnaby, Canada

**Antonina Kolokolova**
Memorial University of Newfoundland, St. John, Canada

**Igor C. Oliveira**
University of Warwick, UK

**Dimitrios Tsintsilidas**
University of Warwick, UK

## Abstract

The *Circuit Size Hierarchy* ($\mathsf{CSH}^a_b$) states that if $a > b \geq 1$ then the set of functions on $n$ variables computed by Boolean circuits of size $n^a$ is strictly larger than the set of functions computed by circuits of size $n^b$. This result, which is a cornerstone of circuit complexity theory, follows from the *non-constructive* proof of the existence of functions of large circuit complexity obtained by Shannon in 1949.

Are there more "constructive" proofs of the Circuit Size Hierarchy? Can we quantify this? Motivated by these questions, we investigate the provability of $\mathsf{CSH}^a_b$ in theories of bounded arithmetic. Among other contributions, we establish the following results:

**(i)** Given any $a > b > 1$, $\mathsf{CSH}^a_b$ is provable in Buss's theory $\mathsf{T}^2_2$.

**(ii)** In contrast, if there are constants $a > b > 1$ such that $\mathsf{CSH}^a_b$ is provable in the theory $\mathsf{T}^1_2$, then there is a constant $\varepsilon > 0$ such that $\mathsf{P}^{\mathsf{NP}}$ requires non-uniform circuits of size at least $n^{1+\varepsilon}$.

In other words, an improved *upper bound* on the proof complexity of $\mathsf{CSH}^a_b$ would lead to new *lower bounds* in complexity theory.

We complement these results with a proof of the *Formula Size Hierarchy* ($\mathsf{FSH}^a_b$) in $\mathsf{PV}_1$ with parameters $a > 2$ and $b = 3/2$. This is in contrast with typical formalizations of complexity lower bounds in bounded arithmetic, which require $\mathsf{APC}_1$ or stronger theories and are not known to hold even in $\mathsf{T}^1_2$.

**2012 ACM Subject Classification** Theory of computation → Complexity theory and logic; Theory of computation → Proof complexity; Theory of computation → Circuit complexity; Theory of computation → Proof theory

**Keywords and phrases** Bounded Arithmetic, Circuit Complexity, Hierarchy Theorems

**Digital Object Identifier** 10.4230/LIPIcs.ITCS.2025.30

**Related Version** *Full Version*: https://eccc.weizmann.ac.il/report/2024/145/

**Funding** This work received support from the Royal Society University Research Fellowship URF\R1\191059; the UKRI Frontier Research Guarantee Grant EP/Y007999/1; the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick, and the Natural Sciences and Engineering Research Council of Canada.

**Acknowledgements** We thank Emil Jeřábek for a discussion about witnessing theorems in bounded arithmetic. We are also grateful to Hanlin Ren for a suggestion that improved our bounds in Corollary 10.

# 1 Introduction

## 1.1 Context and Motivation

The existence of Boolean functions requiring large circuits can be shown by a non-constructive counting argument, as established by Shannon in 1949 [24]. It follows from Shannon's seminal result and a simple padding argument that if $a > b \geq 1$ there are functions computable by circuits of size $n^a$ that cannot be computed by circuits of size $n^b$. In other words, the classification of Boolean functions by their minimum circuit size forms a strict *hierarchy*.

Obtaining a "constructive" form of these results has been a holy grail in computational complexity theory for several decades due to its connections to derandomization and as an approach to separating $\mathsf{P}$ and $\mathsf{NP}$. For instance, if there is a polynomial-time algorithm that given $1^n$ outputs the truth-table of a function $f\colon \{0,1\}^{\log n} \to \{0,1\}$ that requires circuits of size $n^{\Omega(1)}$, then $\mathsf{P} = \mathsf{BPP}$ [9]. In results of this form, a constructive form of the (non-constructive) proof of the existence of hard functions is interpreted *computationally* as the existence of an algorithm of bounded complexity that computes a hard function.

In this paper, rather than focusing on the existence of algorithms to capture the constructiveness of a statement, we explore this notion from the perspective of mathematical logic, specifically concerning its *provability* in certain mathematical theories. We are interested in identifying the weakest theory capable of establishing the aforementioned circuit size hierarchy for Boolean circuits and related results.

As one of our contributions, we present a tight connection between the computational and proof-theoretic perspectives. We demonstrate that proving the non-uniform circuit size hierarchy in a theory known as $\mathsf{T}^1_2$ implies the existence of a function in $\mathsf{P}^{\mathsf{NP}}$ that requires Boolean circuits of size at least $n^{1+\varepsilon}$. The latter is a frontier question in complexity theory (see, e.g., [5]). Thus, in a precise sense, developing more constructive proofs of the circuit size hierarchy would lead to significant progress on explicit circuit lower bounds.

We now proceed to describe this result and other contributions of this work in detail.

## 1.2 Results

We will be concerned with standard theories of bounded arithmetic. These theories are designed to capture proofs that manipulate and reason with concepts from a specified complexity class. Notable examples include Cook's theory $\mathsf{PV}_1$ [7], which formalizes polynomial-time reasoning; Jeřábek's theory $\mathsf{APC}_1$ [10, 11, 13], which extends $\mathsf{PV}_1$ by incorporating the dual weak pigeonhole principle for polynomial-time functions and formalizes probabilistic polynomial-time reasoning; and Buss's theories $\mathsf{T}^i_2$ [2], which incorporate induction principles corresponding to various levels of the polynomial-time hierarchy.

For an introduction to bounded arithmetic, we refer to [3]. For its connections to computational complexity and a discussion on the formalization of complexity theory, we refer to [23].[1] Here we only recall that theory $\mathsf{PV}_1$ corresponds essentially to $\mathsf{T}^0_2$ [12], and that $\mathsf{T}^0_2 \subseteq \mathsf{T}^1_2 \subseteq \mathsf{T}^2_2$ correspond to the first levels of Buss's hierarchy. A brief overview of the theories is provided in Section 2.

For a given $n \in \mathbb{N}$, we use $\mathsf{CIRCUIT}[s(n)]$ to denote the set of Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ computed by circuits of size at most $s(n)$. Similarly, when referring to formula size, we write $\mathsf{FORMULA}[s(n)]$. We use $\mathsf{SIZE}[s(n)]$ to denote the set of languages $L \subseteq \{0,1\}^*$ that admit a sequence of circuits of size at most $s(n)$.

[1] In particular, the reference [23] contains a detailed discussion of some aspects of the formalization of the statements appearing below.

## Circuit Size Hierarchy

For rationals $a > b \geq 1$ and $n_0$, we consider the following sentence:[2]

$$\mathsf{CSH}[a,b,n_0] \equiv \forall n \geq n_0 \in \mathsf{Log},\ \exists \text{ circuit } D\colon \{0,1\}^n \to \{0,1\} \text{ of size} \leq n^a,$$
$$\forall \text{ circuit } C\colon \{0,1\}^n \to \{0,1\} \text{ of size} \leq n^b,\ \exists x \in \{0,1\}^n \text{ such that } D(x) \neq C(x).$$

In other words, $\mathsf{CSH}[a,b,n_0]$ states that $\mathsf{CIRCUIT}[n^a] \not\subseteq \mathsf{CIRCUIT}[n^b]$ whenever $n \geq n_0$.

Next, we state our first result.

▶ **Theorem 1.** *The following results hold:*

**(i)** *For every choice of rationals $a$ and $b$ with $a > b > 1$, and for every large enough $n_0 \in \mathbb{N}$,*

$$\mathsf{T}^2_2 \vdash \mathsf{CSH}[a,b,n_0]\,.$$

**(ii)** *If there are rationals $a > b > 1$ and a constant $n_0 \in \mathbb{N}$ such that*

$$\mathsf{T}^1_2 \vdash \mathsf{CSH}[a,b,n_0]\,,$$

*then there is a constant $\varepsilon > 0$ and a language $L \in \mathsf{P}^{\mathsf{NP}}$ such that $L \notin \mathsf{SIZE}[n^{1+\varepsilon}]$.*

**(iii)** *Similarly to the previous item, if $\mathsf{PV}_1 \vdash \mathsf{CSH}[a,b,n_0]$, there is $L \in \mathsf{P}$ such that $L \notin \mathsf{SIZE}[n^{1+\varepsilon}]$.*

To put it another way, we can establish a circuit size hierarchy within the theory $\mathsf{T}^2_2$. If this result could also be proven in the theory $\mathsf{T}^1_2$, it would lead to a significant breakthrough in circuit lower bounds. Thus, by enhancing the proof complexity upper bound for the provability of the circuit size hierarchy, we can achieve new circuit lower bounds.

The proof technique of Item (ii) also applies to the theory $\mathsf{T}^2_2$, which combined with Item (i) gives us a superlinear lower bound for a language in $\mathsf{P}^{\Sigma^p_2}$, but this is already known by Kannan's theorem [15].

Note that in Theorem 1 Items (ii) and (iii) we obtain a lower bound against circuits of size $n^{1+\varepsilon}$, where the constant $\varepsilon > 0$ depends on the proof of $\mathsf{CSH}[a,b,n_0]$ in the corresponding theory. In other words, while the sentence claims the existence of hardness against circuits of size $n^b$, we are only able to extract a weaker lower bound for an explicit problem.

In our next result, we describe a setting where we can extract all the hardness from a proof of the corresponding sentence.

## Succinct Circuit Size Hierarchy

We define what we call the succinct version of the circuit size hierarchy, where we substitute the upper bound circuit with a collection of labelled examples for the function, which can always represent a circuit. For rationals $a > b \geq 1$ and $n_0$, we consider the following sentence:

$$\mathsf{SCSH}[a,b,n_0] \equiv \forall n \geq n_0 \in \mathsf{Log},\ \exists \text{ collection } \{(x^1,b^1),\ldots,(x^\ell,b^\ell)\} \text{ of size } \ell \leq n^a \text{ with}$$
$$|x^i| = n \wedge |b^i| = 1 \text{ for each } i \in [\ell] \text{ and } x^i \neq x^j \text{ for distinct } i,j \in [\ell]\,,$$
$$\forall \text{ circuit } C\colon \{0,1\}^n \to \{0,1\} \text{ of size} \leq n^b,\ \exists i \in [\ell] \text{ s.t. } C(x^i) \neq b^i.$$

In other words, $\mathsf{SCSH}[a,b,n_0]$ states that for every $n \geq n_0$ there is a collection of $\ell \leq n^a$ labelled examples such that every circuit of size at most $n^b$ disagrees with at least one of its labels. The truth of this statement can be validated by a counting argument, similarly with the circuit size hierarchy proof.

[2] The abbreviation $n \in \mathsf{Log}$ denotes that $n$ is the length of a variable $N$ (see, e.g., [23] for more details).

We obtain the following results on the proof complexity of the succinct circuit size hierarchy.

▶ **Theorem 2.** *The following results hold:*

**(i)** *For every choice of rationals $a > b > 1$ and for every large enough $n_0 \in \mathbb{N}$,*

$$\mathsf{T}^2_2 \vdash \mathsf{SCSH}[a, b, n_0]\,.$$

**(ii)** *If there are rationals $a > b > 1$ and a constant $n_0 \in \mathbb{N}$ such that*

$$\mathsf{T}^1_2 \vdash \mathsf{SCSH}[a, b, n_0]\,,$$

*then there is a language $L \in \mathsf{P}^{\mathsf{NP}}$ such that $L \notin \mathsf{SIZE}[n^b]$.*

In our final result, we investigate the provability of size hierarchies for more restricted computational models in $\mathsf{T}^1_2$ and weaker theories.

### Formula Size Hierarchy

For rationals $a > b \geq 1$ and $n_0$, we consider the following sentence:

$$\mathsf{FSH}[a, b, n_0] \equiv \forall n \geq n_0 \in \mathsf{Log},\ \exists \text{ formula } F\colon \{0,1\}^n \to \{0,1\} \text{ of size} \leq n^a,$$
$$\forall \text{ formula } G\colon \{0,1\}^n \to \{0,1\} \text{ of size} \leq n^b,\ \exists x \in \{0,1\}^n \text{ such that } F(x) \neq G(x).$$

In other words, $\mathsf{FSH}(a, b, n_0)$ states that $\mathsf{FORMULA}[n^a] \not\subseteq \mathsf{FORMULA}[n^b]$ whenever $n \geq n_0$.

We establish that for some parameters a formula size hierarchy is provable already in $\mathsf{PV}_1$.

▶ **Theorem 3.** *Consider rationals $a > 2$ and $b = 3/2$, and let $n_0$ be a large enough positive integer. Then*

$$\mathsf{PV}_1 \vdash \mathsf{FSH}[a, b, n_0]\,.$$

While many lower bounds can be proven in $\mathsf{APC}_1$ and stronger theories (see [22, 23, 4] and references therein), Theorem 3 provides an example of a non-trivial lower bound (under a "Log" formalization; see [23, Section 4.1]) that can be established in $\mathsf{PV}_1$, which might be of independent interest.

## 1.3 Techniques

The proofs of Items (ii) and (iii) in Theorem 1 are inspired by arguments from [18, 17] that rely on a combination of a witnessing theorem with a term elimination strategy. Recall that the witnessing theorem allows us to extract computational information from a proof of the sentence in the theory. Roughly speaking, in our context this implies that the first existential quantifier in the sentence $\mathsf{CSH}[a, b, n_0]$, which corresponds to a circuit computing a hard function, can be witnessed by a finite number of terms $t_1, \ldots, t_k$ of the corresponding theory. In $\mathsf{PV}_1$, a term yields a polynomial-time function, while in $\mathsf{T}^1_2$ a term yields a polynomial-time function with access to an $\mathsf{NP}$ oracle. The main difficulty is that (1) for a given input length $n$ it is not clear which term among $t_1, \ldots, t_k$ succeeds in constructing a hard function, and (2) for a term to succeed we must provide counter-examples to the candidate witnesses provided by previous terms.

As in previous papers, we assume that the conclusion of the theorem does not hold, and use this assumption to rule out the correctness of each term. This leads to a contradiction, meaning that the original sentence is not provable in the corresponding theory. Implementing this plan requires a careful argument, and we are currently only able to carry it out under a complexity inclusion in $\mathsf{SIZE}[n^{1+\varepsilon}]$ as opposed to $\mathsf{SIZE}[n^b]$. The proof of the result is given in Section 3.1.

On the other hand, in the case of the succinct circuit size hierarchy, the argument for Item (ii) of Theorem 2 is simpler and allows us to start with the weaker assumption that $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{SIZE}[n^b]$. Without getting into the technical details, the main reason for not losing hardness in this result is that given a labelled list of examples and access to an NP oracle, we can efficiently compute a minimum size circuit that agrees with this list of inputs. Consequently, we can check if a candidate labelled list provided by a term is indeed hard, or produce a counter-example when this is not the case. The same computation is not available in the case of Theorem 1, since it is not clear how to efficiently compute with access to an NP oracle if a given circuit admits a smaller equivalent circuit. The proof of Item (ii) of Theorem 2 appears in Section 3.2.

The proofs of Theorem 1 Item (i) and Theorem 2 Item (i) are given in Section 3.3. The formalization of these hierarchies in $\mathsf{T}^2_2$ is easily done with access to the dual Weak Pigeonhole Principle for polynomial-time functions, a principle which is known to be available in $\mathsf{T}^2_2$. In more detail, CSH follows from SCSH in $\mathsf{PV}_1$, while SCSH can be established in theory $\mathsf{APC}_1$, which is contained in $\mathsf{T}^2_2$.

Finally, in the proof of Theorem 3 we formalize in $\mathsf{PV}_1$ that the parity function on $n$ bits can be computed by formulas of size $O(n^2)$ and require formulas of size $\Omega(n^{3/2})$. This yields in $\mathsf{PV}_1$ a proof of $\mathsf{FSH}[a, b, n_0]$ for any choice of parameters $a > 2$, large enough $n_0$, and $b = 3/2$. The upper bound on the complexity of parity follows from a straightforward formalization of the correctness of the formula obtained via a divide-and-conquer procedure. On the other hand, in order to show the formula lower bound we formalize Subbotovskaya's argument [25] based on the method of restrictions. To implement the proof in $\mathsf{PV}_1$, we directly define an efficient refuter that given a small formula outputs an input string where it fails to compute the parity function. The correctness of the refuter is established by induction using an induction principle available in the theory $\mathsf{S}^1_2$. We then rely on a conservation result showing that the proof can also be done in $\mathsf{PV}_1$. A detailed exposition of the argument appears in Section 4.

# 2 Preliminaries

## 2.1 Complexity Theory

We employ standard definitions from complexity theory, such as basic complexity classes, Boolean circuits, and Boolean formulas (see, e.g., [1]).

Let $\mathbb{N}$ represent the set of non-negative integers. For any $a \in \mathbb{N}$, let $|a|$ denote the length of its binary representation, defined as $|a| \triangleq \lceil \log_2(a+1) \rceil$. For a constant $k \geq 1$, a function $f \colon \mathbb{N}^k \to \mathbb{N}$ is said to be computable in polynomial time if $f(x_1, \ldots, x_k)$ can be computed in time polynomial in $|x_1|, \ldots, |x_k|$. For convenience, we might write $|\vec{x}| \triangleq |x_1|, \ldots, |x_k|$. The class FP denotes the set of polynomial-time computable functions. Although the definition of polynomial time typically refers to a machine model, FP can also be defined in a machine-independent manner as the closure of a set of base functions $\mathcal{F}$ (not described here) under *composition* and *limited recursion on notation.* A function $f(\vec{x}, y)$ is defined from functions $g(\vec{x})$, $h(\vec{x}, y, z)$, and $k(\vec{x}, y)$ by *limited recursion on notation* if

$$f(\vec{x}, 0) = g(\vec{x})$$
$$f(\vec{x}, y) = h(\vec{x}, y, f(\vec{x}, \lfloor y/2 \rfloor))$$
$$f(\vec{x}, y) \leq k(\vec{x}, y)$$

for every sequence $(\vec{x}, y)$ of natural numbers. Cobham [6] established that $\mathsf{FP}$ is the smallest class of functions that contains the base functions $\mathcal{F}$ and is closed under composition and limited recursion on notation.

## 2.2 Bounded Arithmetic

### 2.2.1 Logical Theories

We recall the definitions of some standard theories of bounded arithmetic. For more details, the reader can consult [16, 8, 20].

#### 2.2.1.1 Cook's Theory PV [7]

The theory $\mathsf{PV}_1$ is designed to model the set $\mathbb{N}$ of natural numbers with the standard interpretations for constants and function symbols like $0, +, \times$, etc. The vocabulary (language) of $\mathsf{PV}$, denoted $\mathcal{L}_{\mathsf{PV}}$, includes a function symbol for each polynomial-time algorithm $f \colon \mathbb{N}^k \to \mathbb{N}$, where $k$ is any constant. These function symbols and their defining axioms are derived using Cobham's characterization of polynomial-time functions discussed above. While Cook's $\mathsf{PV}$ was an equational theory, it was later extended in [19] to a first-order theory $\mathsf{PV}_1$, which includes an induction axiom scheme that simulates binary search. It can be shown that $\mathsf{PV}_1$ allows induction over quantifier-free formulas (i.e., polynomial-time predicates).

$\mathsf{PV}_1$ can be formulated with all axioms as universal formulas (i.e., $\forall \vec{x}\, \phi(\vec{x})$, where $\phi$ is free of quantifiers). Thus, $\mathsf{PV}_1$ is a *universal theory.* Although the definition of $\mathsf{PV}_1$ is quite technical, the theory is fairly robust and the details of its definition are often unnecessary for practical purposes. In particular, $\mathsf{PV}_1$ has an equivalent formalizations that does not rely on Cobham's result, e.g. [12].

#### 2.2.1.2 Jeřábek's Theory $\mathsf{APC}_1$ [10, 11, 13]

$\mathsf{APC}_1$ extends $\mathsf{PV}_1$ with the *dual Weak Pigeonhole Principle* ($\mathsf{dWPHP}$) for $\mathsf{PV}_1$ functions:

$$\mathsf{APC}_1 \triangleq \mathsf{PV} \cup \{\mathsf{dWPHP}(f) \mid f \in \mathcal{L}_{\mathsf{PV}}\}.$$

Each sentence $\mathsf{dWPHP}(f)$ postulates that, for every length $n = |N|$ and for every choice of $\vec{z}$, there is $y < (1 + 1/n) \cdot 2^n$ such that $f(\vec{z}, x) \neq y$ for every $x < 2^n$. It is known that $\mathsf{APC}_1$ is contained in $\mathsf{T}^2_2$ [21].

#### 2.2.1.3 Buss's Theories $\mathsf{S}^i_2$ and $\mathsf{T}^i_2$ [2]

The language $\mathcal{L}_B$ for these theories includes predicate symbols $=$ and $\leq$, constant symbols 0 and 1, and function symbols $S$ (successor), $+$, $\cdot$, $\lfloor x/2 \rfloor$, $|x|$ (interpreted as the length of $x$), and $\#$ (interpreted as $x\#y = 2^{|x|\cdot|y|}$, known as "smash").

Recall that a *bounded quantifier* is a quantifier of the form $Qy \leq t$, where $Q \in \{\exists, \forall\}$ and $t$ is a term not involving $y$. Similarly, a *sharply bounded quantifier* is one of the form $Qy \leq |t|$. A formula where each quantifier appears bounded (or sharply bounded) is called a bounded (or sharply bounded) formula.

We can create a hierarchy of formulas by counting alternations of bounded quantifiers. The class $\Pi^b_0 = \Sigma^b_0$ contains the sharply bounded formulas. Recursively, for each $i \geq 0$, the classes $\Sigma^b_i$ and $\Pi^b_i$ are defined by the quantifier structure of the sentence, ignoring sharply bounded quantifiers. For instance, if $\varphi \in \Sigma^b_0$ and $\psi \triangleq \exists y \leq t(\vec{x})\ \varphi(y, \vec{x})$, then $\psi \in \Sigma^b_1$. For the general case of the definition, see [16]. It is known that for each $i \geq 1$, a predicate $P(\vec{x})$ is in $\Sigma_i^p$ (the $i$-th level of the polynomial hierarchy) if and only if there is a $\Sigma_i^b$-formula that agrees with it over $\mathbb{N}$.

These theories share a common set of finitely many axioms, BASIC, which postulate the expected arithmetic behavior of the constants, predicates, and function symbols. The only difference among the theories is the type of induction axiom scheme each one postulates.

$\mathsf{T}_2^i$ is a theory in the language $\mathcal{L}_B$ that extends BASIC by including the induction axiom IND:

$$\varphi(0) \wedge \forall x\,(\varphi(x) \to \varphi(x+1)) \to\ \forall x\, \varphi(x)$$

for all $\Sigma_i^b$-formulas $\varphi(a)$. The formula $\varphi(a)$ may contain other free variables in addition to $a$.

$\mathsf{S}_2^i$ is a theory in the language $\mathcal{L}_B$ that extends BASIC by including the polynomial induction axiom PIND:

$$\varphi(0) \wedge \forall x\,(\varphi(\lfloor x/2 \rfloor) \to \varphi(x)) \to\ \forall x\, \varphi(x)$$

for all $\Sigma_i^b$-formulas $\varphi(a)$. The formula $\varphi(a)$ may contain other free variables in addition to $a$.

#### 2.2.1.4 Theory $\mathsf{S}_2^1(\mathsf{PV})$

When proving some results in $\mathsf{S}_2^1$, it is often convenient to use a more expressive vocabulary that easily describes any polynomial-time function. This can be done in a *conservative* manner, meaning the power of the theory is not increased. Specifically, let $\Gamma$ be a set of $\mathcal{L}_B$-formulas. We say that a polynomial-time function $f\colon \mathbb{N}^k \to \mathbb{N}$ is $\Gamma$-*definable* in $\mathsf{S}_2^1$ if there exists a formula $\psi(\vec{x}, y) \in \Gamma$ such that the following conditions are met:

**(i)** For every $\vec{a} \in \mathbb{N}^k$, $f(\vec{a}) = b$ if and only if $\mathbb{N} \models \varphi(\vec{a}, b)$.

**(ii)** $\mathsf{S}_2^1 \vdash \forall \vec{x}\,(\exists y\,(\varphi(\vec{x}, y) \wedge \forall z\,(\varphi(\vec{x}, z) \to y = z)))$.

Every function $f \in \mathsf{FP}$ is $\Sigma_1^b$-definable in $\mathsf{S}_2^1$. By incorporating all functions in FP into the vocabulary of $\mathsf{S}_2^1$ and extending the axioms of $\mathsf{S}_2^1$ with their defining equations, we obtain a theory $\mathsf{S}_2^1(\mathsf{PV})$. This theory allows polynomial-time predicates to be referred to using quantifier-free formulas. $\mathsf{S}_2^1(\mathsf{PV})$ remains conservative over $\mathsf{S}_2^1$, meaning any $\mathcal{L}_B$-sentence provable in $\mathsf{S}_2^1(\mathsf{PV})$ is also provable in $\mathsf{S}_2^1$. Finally, it is known that $\mathsf{S}_2^1(\mathsf{PV})$ proves the polynomial induction scheme for both $\Sigma_1^b$-formulas and $\Pi_1^b$-formulas within the extended vocabulary.

### 2.2.2 The KPT Witnessing Theorem

The following witnessing theorem (a variant of Herbrand's theorem) is proved in [19] (cf. also [16, Theorem 7.4.1]) for universal theories (like the theory $\mathsf{PV}_1$).

▶ **Theorem 4** (KPT Theorem for ∀∃∀∃ sentences)**.** *Let* T *be a universal theory with vocabulary* $\mathcal{L}$. *Let* $\varphi$ *be an open* $\mathcal{L}$*-formula, and suppose that*

$$\mathsf{T} \vdash \forall x\, \exists y\, \forall z\, \exists w\ \varphi(x, y, z, w).$$

*Then there is a finite sequence* $s_1, \ldots, s_k$ *of* $\mathcal{L}$*-terms such that*

$$\mathsf{T} \vdash \forall x, z_1, \ldots, z_k\, \big(\psi(x, s_1(x), z_1) \vee \psi(x, s_2(x, z_1), z_2) \vee \cdots \vee \psi(x, s_k(x, z_1, \ldots, z_{k-1}), z_k)\big),$$

*where*

$$\psi(x, y, z) \triangleq \exists w\ \varphi(x, y, z, w).$$

We can also apply the KPT Theorem to each theory $\mathsf{T}_2^i$ (for $i \geq 1$) using a conservative extension of the theory that admits a universal axiomatization. The corresponding theory is called $\mathsf{PV}_{i+1}$ [19]. In $\mathsf{PV}_{i+1}$, each term is equivalent to an $\mathsf{FP}^{\Sigma_i^p}$ function over the standard model. This leads to the following result.

▶ **Theorem 5** (Consequence of the KPT Theorem for Theory $\mathsf{T}_2^i$)**.** *Let $i \geq 1$, $\varphi(x, y, w, z)$ be a $\Pi_i^b$-formula, and suppose that*

$$\mathsf{T}_2^i \vdash \forall x \, \exists y \, \forall z \, \exists w \; \varphi(x, y, w, z).$$

*Then there is a finite sequence $f_1, \ldots, f_k$ of function symbols, each corresponding to an $\mathsf{FP}^{\Sigma_i^p}$ function, such that*

$$\mathbb{N} \models \forall x, z_1, \ldots, z_k \left(\psi(x, f_1(x), z_1) \vee \psi(x, f_2(x, z_1), z_2) \vee \cdots \vee \psi(x, f_k(x, z_1, \ldots, z_{k-1}), z_k)\right),$$

*where*

$$\psi(x, y, z) \triangleq \exists w \; \varphi(x, y, z, w).$$

# 3 Circuit Size Hierarchies in Bounded Arithmetic

## 3.1 Explicit Circuit Lower Bounds from Provability in $\mathsf{PV}_1$ and $\mathsf{T}_2^1$

In this section, we prove Theorem 1 Items (ii) and Items (iii).

▶ **Theorem 6** (Theorem 1 Item (iii))**.** *If there are rationals $a > b > 1$ and $n_0 \in \mathbb{N}$ such that*

$$\mathsf{PV}_1 \vdash \mathsf{CSH}[a, b, n_0]\,,$$

*then there is a constant $\varepsilon > 0$ and a language $L \in \mathsf{P}$ such that $L \notin \mathsf{SIZE}[n^{1+\varepsilon}]$.*

**Proof.** Towards a contradiction, suppose that $\mathsf{PV}_1 \vdash \mathsf{CSH}[a, b, n_0]$ for rationals $a > b > 1$ and some constant $n_0$ and that $P \subseteq \bigcap_{\varepsilon>0} \mathsf{SIZE}[n^{1+\varepsilon}]$. The sentence $\mathsf{CSH}[a, b, n_0]$ has the form $\forall\exists\forall\exists$:

$$\mathsf{CSH}[a, b, n_0] \triangleq \forall n \geq n_0 \in \mathsf{Log},\; \exists \text{ circuit } D \; \forall \text{ circuit } C \;\; \psi_{a,b}(n, D, C)\,,$$

where $\psi_{a,b}(n, D, C)$ is the existential formula:

$$\psi_{a,b}(n, D, C) \triangleq \exists x \; |x| \leq n \wedge \mathsf{SIZE}(D) \leq n^a \wedge (\mathsf{SIZE}(C) \leq n^b \rightarrow \; D(x) \neq C(x)).$$

Therefore, we can apply the KPT Theorem (Theorem 4), which provides $\mathsf{PV}_1$-terms, equivalently $\mathsf{FP}$ functions, $s_1, \ldots, s_k$, where $k$ is a constant, such that

$$\mathbb{N} \models \psi_{a,b}(n, s_1(1^{(n)}), C_1) \vee \psi_{a,b}(n, s_2(1^{(n)}, C_1), C_2) \vee \cdots \vee \psi_{a,b}(n, s_k(1^{(n)}, C_1, \ldots, C_{k-1}), C_k). \quad (1)$$

In the formula above the circuits $C_1, \ldots, C_k$ are universally quantified.

Next, we use $P \subseteq \bigcap_{\varepsilon>0} \mathsf{SIZE}[n^{1+\varepsilon}]$ to refute each of these disjuncts. We start by considering the following language, $D$-Eval:

$D$-Eval is in $\mathsf{P}$ due to the fact that $s_1, \ldots, s_k \in \mathsf{FP}$ and circuit evaluation is in $\mathsf{FP}$. By our assumption on the circuit complexity of the complexity class $\mathsf{P}$, for every input length $m$ and every $\varepsilon > 0$, $D$-Eval $\in \mathsf{SIZE}[m^{1+\varepsilon}]$, so we can choose

$$\varepsilon_0 \triangleq b^{1/(2k)} - 1 > 0$$

**Algorithm 1** The pseudocode of an algorithm that decides the language $D$-Eval.

**Input :** A string $x$ and a sequence $\langle C_1, C_2, \ldots, C_r\rangle$ of $r \leq k-1$ circuits

1. Define $n \triangleq |x|$;
2. Simulate $s_{r+1}(1^{(n)}, C_1, \ldots, C_r)$ and interpret the output as a Boolean circuit $D\colon \{0,1\}^n \to \{0,1\}$;
   `// We assume w.l.o.g. that` $D$ `is a valid` $n$`-bit circuit of size` $\leq n^a$`, since otherwise the disjunct is trivially false.`
3. Evaluate $D$ on input $x$ and output the result.

and have $D$-Eval $\in \mathsf{SIZE}[m^{b^{1/(2k)}}]$. We also define the constants

$$\epsilon_i \triangleq b^{i/k} \quad \text{and} \quad \delta_i \triangleq b^{(2i-1)/(2k)}$$

for $i = 1, \ldots, k$. Note that $\epsilon_i = (1+\varepsilon_0)\delta_i$ and $\delta_{i+1} > \epsilon_i$.

We start by refuting $\psi_{a,b}(n, s_1(1^{(n)}), C_1)$. We consider inputs of the form $x, \lambda$ to $D$-Eval, where $\lambda$ is the empty sequence. Then the input has length $n + c$ , where $c = O(\log n)$ accounts for the overhead in the encoding of the input. We consider the circuit $C_1^* \in \mathsf{CIRCUIT}[(n+c)^{1+\varepsilon_0}]$, which evaluates as $D$-Eval on inputs of length $n + c$, and we fix the input variables not related to $x$ to represent the empty sequence. The resulting circuit has as input an $n$-bit string $x$ and computes according to $s_1(1^{(n)})$ by definition of the $D$-Eval algorithm. For sufficiently large $n$, we have that $n+c \leq n^{\delta_1} \Rightarrow (n+c)^{1+\varepsilon_0} \leq n^{(1+\varepsilon_0)\delta_1} = n^{\epsilon_1}$, therefore we have the circuit $C_1^* \in \mathsf{CIRCUIT}[n^{\epsilon_1}]$ which agrees with the circuit $s_1(1^{(n)})$ on all $n$-bit inputs. Since $\epsilon_1 \leq b$, we have that $\mathbb{N} \not\models \psi_{a,b}(n, s_1(1^{(n)}), C_1^*)$.

We can apply a similar argument to the next disjunct using the aforementioned circuit $C_1^*$. In more detail, we consider the input $(x, \langle C_1^*\rangle)$ on $D$-Eval, which has length $m = n + 9n^{\epsilon_1}\log(n^{\epsilon_1}) + c \leq n^{\delta_2}$ for sufficiently large $n$ due to $\delta_2 > \epsilon_1$, and a corresponding circuit $C_2^* \in \mathsf{CIRCUIT}[m^{1+\varepsilon_0}]$ provided by the circuit upper bound hypothesis. Similarly, we can fix the $9n^{\epsilon_1}\log(n^{\epsilon_1}) + c$ variables not related to the input string $x$. This provides an $n$-bit circuit $C_2^* \in \mathsf{CIRCUIT}[n^{\epsilon_2}]$ that computes according to the circuit $s_2(1^{(n)}, C_1^*)$, due to the definition of the $D$-Eval algorithm. Since $\epsilon_2 < b$, we have that $\mathbb{N} \not\models \psi_{a,b}(n, s_2(1^{(n)}, C_1^*), C_2^*)$.

Inductively, if we have circuits $C_1^*, C_2^*, \ldots, C_i^*$ for some $i \leq k-1$ of sizes at most $n^{\epsilon_1}, n^{\epsilon_2}, \ldots, n^{\epsilon_i}$, respectively, we consider the input $(x, \langle C_1^*, \ldots, C_i^*\rangle)$ to $D$-Eval, which has length $m = n + 9n^{\epsilon_1}\log(n^{\epsilon_1}) + \cdots + 9n^{\epsilon_i}\log(n^{\epsilon_i}) + c \leq n^{\delta_{i+1}}$ for sufficiently large $n$. Therefore, by taking a corresponding $m^{1+\varepsilon_0}$-size circuit for $D$-Eval and fixing all the inputs except for $x$, we get the circuit $C_{i+1}^* \in \mathsf{CIRCUIT}[n^{\epsilon_{i+1}}] \subseteq \mathsf{CIRCUIT}[n^b]$ which agrees with the circuit $s_{i+1}(1^{(n)}, C_1^*, \ldots, C_i^*)$ on all $n$-bit inputs. Consequently, $\mathbb{N} \not\models \psi_{a,b}(n, s_{i+1}(1^{(n)}, C_1^*, \ldots, C_i^*), C_{i+1}^*)$.

Overall, we can refute all disjuncts in Equation (1), which gives us a contradiction. This completes the proof. ◀

▶ **Theorem 7** (Theorem 1 Item (ii)). *If there are rationals $a > b > 1$ and $n_0 \in \mathbb{N}$ such that*

$$\mathsf{T}_2^1 \vdash \mathsf{CSH}[a, b, n_0]\,,$$

*then there is a constant $\varepsilon > 0$ and a language $L \in \mathsf{P}^{\mathsf{NP}}$ such that $L \notin \mathsf{SIZE}[n^{1+\varepsilon}]$.*

**Proof.** In this case, provability in $\mathsf{T}_2^1$ provides by the KPT Theorem (Theorem 5) functions $s_1, \ldots, s_k$ which are in $\mathsf{FP}^{\mathsf{NP}}$ instead of $\mathsf{FP}$ as in the previous proof. Therefore, the algorithm $D$-Eval is in $\mathsf{P}^{\mathsf{NP}}$ and we use the upper bound $\mathsf{P}^{\mathsf{NP}} \subseteq \bigcap_{\varepsilon>0} \mathsf{SIZE}[n^{1+\varepsilon}]$ to get a contradiction in the same way as above. ◀

Note that in the arguments above we have no control over the constant $\varepsilon > 0$. It depends on the number of disjuncts obtained from the KPT Theorem, which depends on the supposed proof of the hierarchy sentence.

## 3.2 Extracting All the Hardness from Proofs of a Succinct Hierarchy Theorem

In this section, we prove Theorem 2 Item (ii).

▶ **Theorem 8** (Theorem 2 Item (ii)). *If there are rationals $a > b > 1$ and a constant $n_0 \in \mathbb{N}$ such that*

$$\mathsf{T}^1_2 \vdash \mathsf{SCSH}[a, b, n_0] \,,$$

*then there is a language $L \in \mathsf{P}^{\mathsf{NP}}$ such that $L \notin \mathsf{SIZE}[n^b]$.*

**Proof.** The main idea here is to use the proof of $\mathsf{SCSH}$ in order to define a Turing machine $M$ which runs in polynomial time using an $\mathsf{NP}$ oracle and its language is hard against $n^b$-size circuits.

Starting from $\mathsf{T}^1_2 \vdash \mathsf{SCSH}[a, b, n_0]$, we see that the structure of the sentence is $\forall\exists\forall\exists$:

$$\mathsf{SCSH}[a, b, n_0] \triangleq \forall n \geq n_0 \in \mathsf{Log},\ \exists \text{ collection } \mathcal{F},\ \forall \text{ circuit } C\ \ \phi_{a,b}(n, \mathcal{F}, C),$$

where $\phi_{a,b}(n, \mathcal{F}, C)$ is the formula that states that $\mathcal{F}$ is a collection $\{(x^1, b^1), \ldots, (x^\ell, b^\ell)\}$ with $\ell \leq n^a$, where $|x^i| = n$ and $|b^i| = 1$, and that if $C$ is a circuit on $n$ variables and of size $\leq n^b$, then there is some $i \in [\ell]$ such that $C(x^i) \neq b^i$ (we can move the existential quantifier at the front of the formula).

Thus, by the KPT Theorem (Theorem 5), there are $\mathsf{FP}^{\mathsf{NP}}$ functions $f_1, \ldots, f_k$, where $k$ is a fixed constant, such that

$$\mathbb{N} \models \phi_{a,b}(n, f_1(1^{(n)}), C_1) \vee \phi_{a,b}(n, f_2(1^{(n)}, C_1), C_2) \vee \cdots \vee \phi_{a,b}(n, f_k(1^{(n)}, C_1, \ldots, C_{k-1}), C_k). \tag{2}$$

From the relation above, we can see that one of the functions $f_1, \ldots, f_k$ will output a collection that refutes every circuit of size $\leq n^b$. If it is not $f_1$, then there is a counterexample circuit $C_1$, which is used as extra input in $f_2$ and so on. Since $f_1, \ldots, f_k$ are in $\mathsf{FP}^{\mathsf{NP}}$, we can simulate this procedure in a $\mathsf{P}^{\mathsf{NP}}$ Turing machine $M_{a,b}$, described below.

▶ Remark. In contrast with Algorithm 1, the algorithm of the Turing machine $M_{a,b}$ does not need to have the counterexample circuits as input, since it can guess and check them during its process, using the $\mathsf{NP}$ oracle. This difference in the input size is what gives us the $n^b$ lower bound instead of $n^{1+\epsilon}$.

It is easy to see that the language $L(M_{a,b})$ recognised by the Turing machine $M_{a,b}$, is in $\mathsf{P}^{\mathsf{NP}}$. It suffices to show that $L(M_{a,b}) \notin \mathsf{SIZE}[n^b]$.

Consider a circuit $C \in \mathsf{CIRCUIT}[n^b]$. We will show that it fails to recognise $L(M_{a,b})$. Assume that the for-loop in Algorithm 2 ends in the $r$-th iteration with $r \leq k$. We fix the circuits $C_1, C_2, \ldots, C_{r-1}$ found by the algorithm. Then the formula $\phi_{a,b}(n, f_r(1^{(n)}, C_1, \ldots, C_{r-1}), C)$ always holds. If $r < k$ and $C$ did not satisfy it, then the $\mathsf{NP}$ oracle would find $C$ as a counterexample and it would continue to the $(r+1)$-th iteration. If $r = k$, then by the construction of $C_1, C_2, \ldots, C_{k-1}$, the formulas $\phi_{a,b}(n, f_i(1^{(n)}, C_1, \ldots, C_{i-1}), C_i)$ for $i < k$ do not hold, which means by Equation (2) that $\phi_{a,b}(n, f_k(1^{(n)}, C_1, \ldots, C_{k-1}), C)$ is true.

**Algorithm 2** The Turing machine $M_{a,b}$, whose language is hard for $n^b$-size circuits.

**Input :** A bit-string $x$
Define $n \triangleq |x|$;
**for** $i = 1, \ldots, k$ **do**
  Simulate $f_i$ with input $1^{(n)}$ and, if $i > 1$, $C_1, \ldots, C_{i-1}$. Interpret the output as a collection $\mathcal{F} = \{(x^1, b^1), \ldots, (x^\ell, b^\ell)\}$ with $\ell = n^a$;
  Check with an NP oracle whether there exists a circuit $C$ of size $\leq n^b$, such that $C(x^i) = b^i$ for all $i \in [\ell]$;
  If not or if $i = k$, exit the for-loop with the current $\mathcal{F}$;
  If there is such a circuit, then use the NP oracle to find it and name it $C_i$.
**end**
If the pair $(x, 1)$ is in the collection $\mathcal{F}$, then **accept**. Else **reject**.

Since $\mathcal{F} \equiv f_r(1^{(n)}, C_1, \ldots, C_{r-1})$, from $\phi_{a,b}(n, \mathcal{F}, C)$ we get that there is some $i \in [\ell]$, such that $C(x^i) \neq b^i$. However, if $b^i = 1$, then $x^i \in L(M_{a,b})$, and if $b^i = 0$, then $x^i \notin L(M_{a,b})$. In both cases, the circuit $C$ fails to recognise the language $L(M_{a,b})$, and the proof is complete. ◀

## 3.3 Formalization in $\mathsf{T}^2_2$

In this section, we prove Theorem 1 Item (i) and Theorem 2 Item (i). To achieve this, we show that the succinct circuit size hierarchy is provable in $\mathsf{APC}_1$, which is contained in $\mathsf{T}^2_2$. We then observe that the circuit size hierarchy is easily provable from the succinct circuit size hierarchy.

▶ **Theorem 9.** *For every choice of rationals $a > b > 1$ and for every large enough $n_0 \in \mathbb{N}$,*

$$\mathsf{APC}_1 \vdash \mathsf{SCSH}[a, b, n_0]\,.$$

*In particular, $\mathsf{SCSH}[a, b, n_0]$ is provable in $\mathsf{T}^2_2$.*

**Proof.** We define the polynomial-time function, $f$, which takes as input the description of a circuit, $C$, of size $n^b$, which means that the length of the description of $C$ is $9n^b \log n^b$, and outputs a bit string $y$ of length $n^a$ with the property that for all $i = 0, 1, \ldots, n^a - 1$, $y_i = C(i)$.

The correctness of the polynomial-time algorithm $f$ is provable in $\mathsf{PV}_1$. In other words,

$$\mathsf{PV}_1 \vdash \forall n \in \mathsf{Log}\ (\,|x| \leq 9n^b \log n^b\ \wedge\ |y| \leq n^a\,) \rightarrow \\ (\,|f(x)| \leq n^a\ \wedge\ (f(x) = y \leftrightarrow \forall i < n^a\ y_i = \mathsf{Eval}(x, i))). \quad (3)$$

The quantifier $\forall i \leq n^a$ is sharply bounded, so this formula is provable in $\mathsf{PV}_1$.

The theory $\mathsf{APC}_1$ includes the dWPHP axiom for all PV functions with input length $n$ and output length $n + 1$, or equivalently input length $n$ and output length $m$ with $n < m$. From the first part of Equation (3), the input length of $f$ is $9n^b \log n^b$, while the output length is $n^a$. Furthermore, it is provable in $\mathsf{PV}_1$ that there is some constant $n_0$, such that $\forall n \geq n_0\ n^a > 9n^b \log n^b$. Therefore, we can use the axiom:

$$\mathsf{dWPHP}(f) \triangleq \forall n \geq n_0\ \exists y\ (|y| = n^a)\ \forall x\ (|x| = 9n^b \log n^b)\ f(x) \neq y \quad (4)$$

Every circuit of size $n^b$ can be described by a string of size $9n^b \log n^b$, which means that

$$\forall C \in \mathsf{CIRCUIT}[n^b]\ |C| \leq 9n^b \log n^b.$$

Also, from the second part of Equation (3), using the notation for the circuit $C$, we get that

$$f(C) \neq y \leftrightarrow \exists i < n^a \; C(i) \neq y_i.$$

Substituting the last two relations to Equation (4), we get that

$$\mathsf{APC}_1 \vdash \forall n \geq n_0 \in \mathsf{Log}\; \exists y\; (|y| = n^a)\; \forall C \in \mathsf{CIRCUIT}[n^b]\; \exists i < n^a\; C(i) \neq y_i,$$

which is equivalent with $\mathsf{SCSH}[a, b, n_0]$, if we interpret $y$ as the collection

$$\mathcal{F}_y \triangleq \{(0, y_0), (1, y_1), \ldots\}.$$ ◀

▶ **Corollary 10.** *For every choice of rationals $a > b > 1$ and for every large enough $n_0 \in \mathbb{N}$,*

$$\mathsf{T}^2_2 \vdash \mathsf{CSH}[a, b, n_0]\,.$$

**Proof.** Since $a > b$, there is some rational $\epsilon > 0$, such that $a - \epsilon > b$. From Theorem 9, we have got a collection $\mathcal{F} = \{(x^1, b^1), \ldots, (x^\ell, b^\ell)\}$ of size $\ell \leq n^{a-\epsilon}$, such that for all circuits $C$ of size less than $n^b$, there exists $i \in [\ell]$ such that $C(x^i) \neq b^i$. So, we only need to prove that

$$\mathsf{PV}_1 \vdash \exists \text{ circuit } D\colon \{0,1\}^n \to \{0,1\} \text{ of size} \leq n^a,\ \forall i \in [\ell]\; D(x^i) = b^i,$$

and then we can easily deduce that $\mathsf{APC}_1 \vdash \mathsf{CSH}[a, b, n_0]$. The same holds also for $\mathsf{T}^2_2$.

It is sufficient to argue in $\mathsf{PV}_1$ that there is a polynomial-time function $\mathsf{Circuit}(\mathcal{F})$ such that given the collection $\mathcal{F}$ from Theorem 9 outputs a circuit $D\colon \{0,1\}^n \to \{0,1\}$ of the required size such that $\forall i \in [\ell]\; D(x^i) = b^i$. In order to optimize the circuit size, we use that the obtained collection has a specific structure. More precisely, we have that for any $i \in [\ell]$, the strings $x^i$ is the $n$-bit binary representation of the integer $i - 1$. Therefore, we can construct the circuit $D$ in the following way: For every $n$-bit string $x^i$ such that $(x^i, 1) \in \mathcal{F}$, we construct the term $T^i$, which is the conjunction of the first $|\ell|$ least significant bits of $x^i$ (we put the literal $z_j$ if the $j$-th bit of $x^i$ is 1 and $\neg z_j$ if the $j$-th bit of $x^i$ is 0, where $j \leq |\ell|$). Then we make the DNF

$$D \triangleq \bigvee_{(x^i,1) \in \mathcal{F}} T^i.$$

It is easy to see that $D$ agrees with all the pairs of the collection $\mathcal{F}$. For an arbitrary pair $(x^i, b^i)$, if $b^i = 1$, then the bits of $x^i$ satisfy the term $T^i$, hence $D(x^i) = 1$. Otherwise, if $b^i = 0$, we know that the first $|\ell|$ least significant bits of $x^i$ do not satisfy any term of the disjunction (since for all $i$, $x^i \leq \ell$), thus we get that $D(x^i) = 0$.

The DNF $D$ can be viewed as a circuit and its correctness is easily provable in $\mathsf{PV}_1$. This circuit has size at most $n^{a-\epsilon}|\ell|$ (derived by $|\ell| - 1$ $\wedge$-gates for each one of the at most $n^{a-\epsilon}$ terms and at most $n^{a-\epsilon}$ $\vee$-gates for the final disjunction), which is at most $n^{a-\epsilon}(\log n^{a-\epsilon} + 1)$. For large enough $n_0$, we can prove that $\forall n \geq n_0$, $n^{a-\epsilon}(\log n^{a-\epsilon} + 1) \leq n^a$, hence we have the desired result. ◀

## 4 Provability of Formula Size Bounds in $\mathsf{PV}_1$

In this section, we prove Theorem 3. To achieve this, we establish that:

1. The parity function on $n$ bits requires formulas of size $\geq n^{3/2}$ (Section 4.1).
2. The parity function on $n$ bits can be computed by formulas of size $O(n^2) \leq n^a$ for any fixed rational $a > 2$ and large enough $n$ (Section 4.2).
3. Consequently, the formula size hierarchy holds with parameters $a > 2$ and $b = 3/2$, provided that $n_0$ is large enough (Section 4.3).

## 4.1 Subbotovskaya's Lower Bound

### 4.1.1 High-Level Details of the Formalization

In this section, we sketch a formalization in $\mathsf{PV}_1$ of the proof that the parity function on $n$ bits requires Boolean formulas of size $\geq n^{3/2}$ [25].[3] We adapt the argument presented in [14, Section 6.3], which proceeds as follows:

1. [14, Lemma 6.8]: Given a Boolean formula $F$ on $n$-bit inputs, it is possible to fix one of its variables so that the resulting formula $F_1$ satisfies

$$\mathsf{Size}(F_1) \leq (1 - 1/n)^{3/2} \cdot \mathsf{Size}(F).$$

In order to pick the variable to be restricted and its value, one first "normalizes" the formula $F$, as implicitly described in [14, Claim 6.9] (see more details below).

2. [14, Theorem 6.10]: By applying this result $\ell \triangleq n - k$ times, it is possible to obtain a formula $F_\ell$ on $k$-bit inputs such that

$$\mathsf{Size}(F_\ell) \leq \mathsf{Size}(F)\cdot(1-1/n)^{3/2}\cdot(1-1/(n-1))^{3/2}\ldots(1-1/(k+1))^{3/2} = \mathsf{Size}(F)\cdot(k/n)^{3/2}.$$

3. [14, Example 6.11]: If the initial formula $F$ computes the parity function, by setting $\ell = n - 1$ we obtain

$$1 \leq \mathsf{Size}(F_\ell) \leq (1/n)^{3/2} \cdot \mathsf{Size}(F),$$

and consequently $\mathsf{Size}(F) \geq n^{3/2}$.

We recommend reading this section with [14, Section 6.3] at hand. We will slightly modify the argument when formalizing the lower bound in $\mathsf{PV}_1$. In more detail, given a small formula $F$, we recursively construct (and establish correctness by induction) an $n$-bit input $y$ witnessing that $F$ does not compute the parity function. (Actually, for technical reasons related to the induction step, we will simultaneously construct an $n$-bit input $y^0_n$ witnessing that $F$ does not compute the parity function and an $n$-bit input $y^1_n$ witnessing that $F$ does not compute the negation of the parity function.)

Let $s(n)$ be a size bound and $\oplus(x)$ be a $\mathsf{PV}$ function that computes the parity of the binary string described by $x$, i.e., $\oplus(x) \triangleq x_1 \oplus x_2 \oplus \ldots \oplus x_n$, where $x_i$ denotes the $i$-th bit of $x$. To simplify notation, we tacitly view $x$ as a binary string. We assume that the formalization employs a well-behaved function symbol $\oplus$ such that $\mathsf{PV}_1$ proves the basic properties of the parity function, e.g., $\mathsf{PV}_1 \vdash \oplus(x1) = 1 - \oplus(x)$ and $\mathsf{PV}_1 \vdash \oplus(x0) = \oplus(x)$.

We consider the following $\mathcal{L}_{\mathsf{PV}}$-sentence stating that the parity function requires formulas of size at least $s(n)$ for every input length $n \geq 1$:

$$\mathsf{FLB}_s \triangleq \forall N\, \forall n\, \forall F\, (n = |N| \geq 1 \wedge \mathsf{Size}(F) < s(n) \rightarrow \exists x\, (|x|_\ell = n \wedge \mathsf{Eval}(F, x) \neq \oplus(x))\,,$$ [4]

where for convenience of notation we use the function symbol $|w|_\ell$ to compute the bit-length of the string represented by $w$ (under some reasonable encoding).

▶ **Theorem 11.** *Let* $s(n) \triangleq n^{3/2}$. *Then* $\mathsf{PV}_1 \vdash \mathsf{FLB}_s$.

[3] For concreteness, we let the size of a Boolean formula $F$ be the number of leaves of $F$ labeled by an input literal. We allow leaves that are labeled by constants, but we do not charge for them. Consequently, a constant function has formula complexity 0, while a non-constant function has formula complexity at least 1.

**Proof.** Given $b \in \{0,1\}$, we introduce the function $\oplus^b(x) \triangleq \oplus(x) + b \pmod 2$. In order to prove $\mathsf{FLB}_s$ in $\mathsf{PV}_1$, we explicitly consider a polynomial-time function $R(1^{(n)}, F, b)$ with the following properties:[5]

1. Let $b \in \{0,1\}$.
2. If $\mathsf{Size}(F) < s(n)$ then $R(1^{(n)}, F, b)$ outputs an $n$-bit string $y_n^b$ such that $\mathsf{Eval}(F, y_n^b) \neq \oplus^b(y_n^b)$.

In other words, $R(1^{(n)}, F, b)$ witnesses that the formula $F$ does not compute the function $\oplus^b$ over $n$-bit strings. Note that the correctness of $R$ is captured by the bounded universal sentence:

$$\mathsf{Ref}_{R,s} \triangleq \forall 1^{(n)}\, \forall F\, (\mathsf{Size}(F) < s(n) \to |y_n^0|_\ell = |y_n^1|_\ell = n \wedge F(y_n^0) \neq \oplus^0(y_n^0) \wedge F(y_n^1) \neq \oplus^1(y_n^1)),$$

where we employed the abbreviations $y_n^0 \triangleq R(1^{(n)}, F, 0)$ and $y_n^1 \triangleq R(1^{(n)}, F, 1)$. Our plan is to define $R$ and show that $\mathsf{PV}_1 \vdash \mathsf{Ref}_{R,s}$. Note that this implies $\mathsf{FLB}_s$ in $\mathsf{PV}_1$. Jumping ahead, the correctness of $R(1^{(n)}, F, b)$ will be established by polynomial induction on $N$ (equivalently, induction on $n = |N|$). Since $\mathsf{Ref}_{R,s}$ is a universal sentence and $\mathsf{S}^1_2$ is $\forall\Sigma^b_1$-conservative over $\mathsf{PV}_1$, polynomial induction for $\mathsf{NP}$ and $\mathsf{coNP}$ predicates (admissible in $\mathsf{S}^1_2$; see, e.g., [16, Section 5.2]) is available during the formalization. More details follow.

The procedure $R(1^{(n)}, F, b)$ makes use of a few polynomial-time sub-routines (discussed below) and is defined in the following way:

**Algorithm 3** Refuter Algorithm $R(1^{(n)}, F, b)$.

**Input :** $1^{(n)}$ for some $n \geq 1$, formula $F$ over $n$-bit inputs, $b \in \{0,1\}$.
1 Let $s(n) \triangleq n^{3/2}$. If $\mathsf{Size}(F) \geq s(n)$ **return** *"error"*;
2 If $\mathsf{Size}(F) = 0$, $F$ computes a constant function $b_F \in \{0,1\}$. In this case, **return** *the $n$-bit string $y_n^b \triangleq y_1^b 0^{n-1}$ such that $\oplus^b(y_1^b 0^{n-1}) \neq b_F$*;
3 Let $\widetilde{F} \triangleq \mathsf{Normalize}(1^{(n)}, F)$;
  // $\widetilde{F}$ satisfies [14, Claim 6.9], $\mathsf{Size}(\widetilde{F}) \leq \mathsf{Size}(F)$, $\forall x \in \{0,1\}^n\; F(x) = \widetilde{F}(x)$.
4 Let $\rho \triangleq \mathsf{Find\text{-}Restriction}(1^{(n)}, \widetilde{F})$, where $\rho\colon [n] \to \{0,1,\star\}$ and $|\rho^{-1}(\star)| = n-1$;
  // $\rho$ restricts a suitable variable $x_i$ to a bit $c_i$, as in [14, Lemma 6.8].
5 Let $F' \triangleq \mathsf{Apply\text{-}Restriction}(1^{(n)}, \widetilde{F}, \rho)$. Moreover, let $b' \triangleq b \oplus c_i$ and $n' \triangleq n-1$;
  // $F'$ is an $n'$-bit formula; $\forall z \in \{0,1\}^{\rho^{-1}(\star)}\; F'(z) = \widetilde{F}(z \cup x_i \mapsto c_i)$.
6 Let $y_{n'}^{b'} \triangleq R(1^{n'}, F', b')$ and **return** *the $n$-bit string $y_n^b \triangleq y_{n'}^{b'} \cup y_i \mapsto c_i$*;

#### 4.1.1.1 $\mathsf{Normalize}(1^{(n)}, F)$ and its properties (in $\mathsf{S}^1_2$)

We say that a subformula $G$ of $F$ is a *neighbor* of a leaf $z$ if either $z \wedge G$ or $z \vee G$ is a subformula of $F$. We say that a formula $F$ over variables $\{x_1, \ldots, x_n\}$ is in *normal form* if for every $i \in [n]$ and every literal $z \in \{x_i, \overline{x_i}\}$, if $z$ is a leaf of $F$ and $G$ is a neighbor of $z$ in $F$, then $G$ does not contain the variable $x_i$.

[5] For convenience, we often write $1^{(n)}$ instead of explicitly considering parameters $N$ and $n = |N|$. We might also write just $F(x)$ instead of $\mathsf{Eval}(F, x)$.

▶ **Lemma 12.** *There is a polynomial-time function* $\mathsf{Normalize}(1^{(n)}, F)$ *that given a Boolean formula* $F$ *over* $n$ *input variables, outputs a formula* $\widetilde{F}$ *over* $n$ *input variables such that the following holds:*

**(i)** $\mathsf{Size}(\widetilde{F}) \leq \mathsf{Size}(F)$.

**(ii)** *For every input* $x \in \{0,1\}^n$, $\widetilde{F}(x) = F(x)$.

**(iii)** $\widetilde{F}$ *is in normal form.*

**(iv)** $\widetilde{F}$ *is either a constant* $0$ *or* $1$, *or* $\widetilde{F}$ *contains no leaves labeled by constants* $0$ *and* $1$.

*Moreover, the correctness of* $\mathsf{Normalize}(1^{(n)}, F)$ *is provable in* $\mathsf{S}^1_2$.

**Proof Sketch.** It is enough to verify that the proof of [14, Claim 6.9] provides such a polynomial-time function and that its correctness can be established in $\mathsf{S}^1_2$. In more detail, if $F$ is not in normal form, we can efficiently compute a literal $z \in \{x_i, \overline{x_i}\}$ and a neighbor $G$ of $z$ that violates the corresponding property. As shown in [14, Claim 6.9], we can fix any leaf $z' \in \{x_i, \overline{x_i}\}$ in $G$ by an appropriate constant $c$ so that the resulting formula $F_1$ satisfies conditions (i) and (ii) of Lemma 12. After at most $\ell \triangleq \mathsf{Size}(F)$ iterations, we obtain a sequence $F_1, \ldots, F_\ell$ of formulas such that $\widetilde{F} \triangleq F_\ell$ satisfies conditions (i), (ii), and (iii) of the lemma. Moreover, condition (iv) can always be guaranteed by simplifying the final formula, i.e., by replacing subformulas $0 \vee G$ by $G$, $1 \vee G$ by $1$, $0 \wedge G$ by $0$, and $1 \wedge G$ by $G$. The correctness of $\widetilde{F} \triangleq \mathsf{Normalize}(1^{(n)}, F)$ can be established by polynomial induction for $\mathsf{coNP}$ predicates (i.e., $\Pi^b_1$ formulas), which is available in $\mathsf{S}^1_2$. ◀

#### 4.1.1.2 Find-Restriction$(1^{(n)}, \widetilde{F})$ and its properties (in $\mathsf{S}^1_2$)

We argue in $\mathsf{S}^1_2$ and follow the argument from the proof of [14, Lemma 6.8]. Let $\widetilde{F}$ be a formula over $n$ input variables in normal form. We focus on the non-trivial case, and assume that $n \geq 2$, $\mathsf{Size}(\widetilde{F}) \geq 2$, and that $\widetilde{F}$ contains no leaves labeled by constants. Let $\mathsf{Count}(1^{(n)}, F, i)$ be a polynomial-time algorithm that outputs the number of leaves of $F$ that contain the variable $x_i$ (including its appearances as $\overline{x_i}$). Let $w = (w_1, \ldots, w_n)$ be the corresponding sequence of multiplicities, i.e., $w_i \triangleq \mathsf{Count}(1^{(n)}, F, i)$. Note that $\sum_i w_i = \widetilde{s}$, where $\widetilde{s} \triangleq \mathsf{Size}(\widetilde{F})$.

We claim that $\mathsf{S}^1_2$ proves the existence of an index $i \in [n]$ such that $w_i \geq \widetilde{s}/n$. First, for each $j \in [n]$, we define the cumulative sum $v_j \triangleq \sum_{i \leq j} w_j$. Let $v \triangleq (v_0, v_1, \ldots, v_n)$ be the corresponding sequence, where we set $v_0 \triangleq 0$. Notice that $v_n = \widetilde{s}$. Since $v$ contains $n+1$ elements, it can be efficiently computable from $w$. We now argue by induction on $n$ that for some index $j \in [n]$ we have $v_j - v_{j-1} \geq v_n/n$. This implies that $w_j = v_j - v_{j-1} \geq v_n/n = \widetilde{s}/n$, as desired.

If $n = 1$, then $v_1 - v_0 = v_1 = v_1/1$ and the result holds for $j = 1$. Assume the result holds for $n-1$, and consider $v_n$. If $v_n - v_{n-1} \geq v_n/n$, we can pick $j = n$ and we are done. Otherwise, $v_{n-1} \geq v_n - v_n/n = v_n(n-1)/n$. By the induction hypothesis, there is an index $j \in [n-1]$ such that $v_j - v_{j-1} \geq v_{n-1}/(n-1)$. Using the lower bound on $v_{n-1}$, we get that $v_j - v_{j-1} \geq v_n/n$, which concludes the proof.

Consequently, $\mathsf{S}^1_2$ proves the existence of a variable $x_i$ which appears $t \geq \widetilde{s}/n$ times as a leaf of $\widetilde{F}$. Let $z_1, \ldots, z_t$ be the leaves of $\widetilde{F}$ labeled by either $x_i$ or $\overline{x_i}$. Recall that we assume that $n \geq 2$, $\mathsf{Size}(\widetilde{F}) \geq 2$, and that $\widetilde{F}$ satisfies conditions (iii) and (iv) of Lemma 12. Therefore, each leaf $z_j$ has a neighbor subformula $G_j$ in $\widetilde{F}$ that contains some leaf labeled by a literal not in $\{x_i, \overline{x_i}\}$. For this reason, if we set $x_i$ to an appropriate constant $c_j$, $G_j$ will disappear from $F$, thereby erasing at least another leaf not among $z_1, \ldots, z_t$. As in the proof of [14, Lemma 6.8], if we let $c \in \{0,1\}$ be the constant that appears more often among

$c_1, \ldots, c_t$ and set $x_i \mapsto c$ in the restriction $\rho$, all the leaves $z_1, \ldots, z_t$ will be eliminated from $\widetilde{F}$ together with at least $t/2$ additional leaves.[6] Thus the total number of eliminated leaves, which we specify using a polynomial-time function $\mathsf{NumRemoved}(1^{(n)}, \widetilde{F}, \rho)$, satisfies

$$\mathsf{NumRemoved}(1^{(n)}, \widetilde{F}, \rho) \geq t + \frac{t}{2} \geq \frac{3\widetilde{s}}{2n}.$$

Overall, it follows that

$$\mathsf{S}^1_2 \vdash \widetilde{F} = \mathsf{Normalize}(1^{(n)}, F) \wedge \rho = \mathsf{Find\text{-}Restriction}(1^{(n)}, \widetilde{F}) \rightarrow \\ \mathsf{NumRemoved}(1^{(n)}, \widetilde{F}, \rho) \geq \frac{3}{2n} \cdot \mathsf{Size}(\widetilde{F})\,.$$

#### 4.1.1.3 Apply-Restriction$(1^{(n)}, \widetilde{F}, \rho)$ and its properties (in $\mathsf{S}^1_2$)

We only sketch the details. This is simply a polynomial-time algorithm that, given a formula $\widetilde{F}$ on $n$ input variables and a restriction $\rho\colon [n] \to \{0, 1, *\}$ with $|\rho^{-1}(\star)| = n - 1$ (i.e., $\rho$ restricts a single variable $x_i$ to a constant $c_i \in \{0, 1\}$), outputs a formula $F'$ over $n - 1$ input variables that sets every literal $z \in \{x_i, \overline{x_i}\}$ to the corresponding constant and simplifies the resulting formula, e.g., replaces subformulas $0 \vee G$ by $G$, $1 \vee G$ by 1, $0 \wedge G$ by 0, and $1 \wedge G$ by $G$. Additionally, for $F' = \mathsf{Apply\text{-}Restriction}(1^{(n)}, \widetilde{F}, \rho)$, we have

$$\mathsf{S}^1_2 \vdash \mathsf{Size}(F') \leq \mathsf{Size}(\widetilde{F}) - \mathsf{NumRemoved}(1^{(n)}, \widetilde{F}, \rho) \wedge \\ \forall z \in \{0, 1\}^{\rho^{-1}(\star)}\; F'(z) = \widetilde{F}(z \cup x_i \mapsto c_i)\,. \tag{5}$$

Using the computed bound on $\mathsf{NumRemoved}(1^{(n)}, \widetilde{F}, \rho)$ for $\rho = \mathsf{Find\text{-}Restriction}(1^{(n)}, \widetilde{F})$, we obtain that for $\widetilde{F}$ and $F'$ defined as above (with $s' \triangleq \mathsf{Size}(F')$ and $\widetilde{s} \triangleq \mathsf{Size}(\widetilde{F})$), and assuming that $n \geq 2$,

$$\mathsf{S}^1_2 \vdash s' \leq \widetilde{s} - \frac{3}{2n} \cdot \widetilde{s} = \widetilde{s} \cdot \left(1 - \frac{3}{2n}\right) \leq \widetilde{s} \cdot \left(1 - \frac{1}{n}\right)^{3/2}. \tag{6}$$

The last inequality uses that $\mathsf{S}^1_2 \vdash \forall a,\; a \geq 2 \rightarrow (1 - 3/(2a))^2 \leq (1 - 1/a)^3$, which one can easily verify.

Note that $R(1^{(n)}, F, b)$ runs in time polynomial in $n + |F| + |b|$ and that it is definable in $\mathsf{S}^1_2$. Next, we establish the correctness of $R(1^{(n)}, F, b)$ in $\mathsf{S}^1_2$ .

▶ **Lemma 13.** *Let $s(n) \triangleq n^{3/2}$. Then $\mathsf{S}^1_2 \vdash \mathsf{Ref}_{R,s}$.*

**Proof.** We consider the formula $\varphi(N)$ defined as

$$\forall F\, \forall n = |N| \geq 1 (\mathsf{Size}(F) < s(n)) \rightarrow (|y^0_n|_\ell = |y^1_n|_\ell = n \wedge F(y^0_n) \neq \oplus^0(y^0_n) \wedge F(y^1_n) \neq \oplus^1(y^1_n))\,,$$

where as before we use $y^0_n \triangleq R(1^{(n)}, F, 0)$ and $y^1_n \triangleq R(1^{(n)}, F, 1)$. Note that $\varphi(N)$ is a $\Pi^b_1$ formula. Below, we argue that

$$\mathsf{S}^1_2 \vdash \varphi(1) \quad \text{and} \quad \mathsf{S}^1_2 \vdash \forall N\, \varphi(\lfloor N/2 \rfloor) \rightarrow \varphi(N)\,.$$

Then, by polynomial induction for $\Pi^b_1$ formulas (available in $\mathsf{S}^1_2$) and using that $\varphi(0)$ trivially holds, it follows that $\mathsf{S}^1_2 \vdash \forall N\, \varphi(N)$. In turn, this yields $\mathsf{S}^1_2 \vdash \mathsf{Ref}_{R,s}$.

[6] The existence of such a constant $c$ can be proved in $\mathsf{S}^1_2$ in a way that is similar to the proof that some variable $x_i$ appears in at least $\widetilde{s}/n$ leaves.

**Base Case: $\mathsf{S}^1_2 \vdash \varphi(1)$.** In this case, for a given formula $F$ and length $n$, the hypothesis of $\varphi(1)$ is satisfied only if $n = 1$ and $\mathsf{Size}(F) = 0$. Let $y_1^0 \triangleq R(1, F, 0)$ and $y_1^1 \triangleq R(1, F, 1)$. We need to prove that

$$|y_1^0|_\ell = |y_1^1|_\ell = 1 \wedge F(y_1^0) \neq \oplus^0(y_1^0) \wedge F(y_1^1) \neq \oplus^1(y_1^1)\,.$$

Since $n = 1$ and $\mathsf{Size}(F) = 0$, $F$ evaluates to a constant $b_F$ on every input bit. The statement above is implied by Line 2 in the definition of $R(n, F, b)$.

**(Polynomial) Induction Step: $\mathsf{S}^1_2 \vdash \forall N\, \varphi(\lfloor N/2 \rfloor) \rightarrow \varphi(N)$.** Fix an arbitrary $N$, let $n \triangleq |N|$, and assume that $\varphi(\lfloor N/2 \rfloor)$ holds. By the induction hypothesis, for every formula $F'$ with $\mathsf{Size}(F') < n'^{3/2}$, where $n' \triangleq n - 1$, we have

$$|y_{n'}^0|_\ell = |y_{n'}^1|_\ell = n' \;\wedge\; F'(y_{n'}^0) \neq \oplus^0(y_{n'}^0) \;\wedge\; F'(y_{n'}^1) \neq \oplus^1(y_{n'}^1)\,, \tag{7}$$

where $y_{n'}^0 \triangleq R(1^{n'}, F', 0)$ and $y_{n'}^1 \triangleq R(1^{n'}, F', 1)$.

Now let $n \geq 2$, and let $F$ be a formula over $n$-bit inputs of size $< n^{3/2}$. By the size bound on $F$, $R(1^{(n)}, F, b)$ ignores Line 1. If $\mathsf{Size}(F) = 0$, then similarly to the base case it is trivial to check that the conclusion of $\varphi(N)$ holds. Therefore, we assume that $\mathsf{Size}(F) \geq 1$ and $R(1^{(n)}, F, b)$ does not stop at Line 2. Let $\widetilde{F} \triangleq \mathsf{Normalize}(1^{(n)}, F)$ (Line 3), $\rho \triangleq \mathsf{Find\text{-}Restriction}(1^{(n)}, \widetilde{F})$ (Line 4), $F' \triangleq \mathsf{Apply\text{-}Restriction}(1^{(n)}, \widetilde{F}, \rho)$ (Line 5), $n' \triangleq n - 1$ (Line 5), and $b' \triangleq b \oplus c_i$ (Line 5), where $\rho$ restricts the variable $x_i$ to the bit $c_i$. Moreover, for convenience, let $s \triangleq \mathsf{Size}(F)$, $\widetilde{s} \triangleq \mathsf{Size}(\widetilde{F})$, and $s' \triangleq \mathsf{Size}(F')$. By Lemma 12 Item (i), Equation (6), and the bound $s < n^{3/2}$,

$$\mathsf{S}^1_2 \vdash s' \leq \widetilde{s} \cdot (1 - 1/n)^{3/2} \leq s \cdot (1 - 1/n)^{3/2} < n^{3/2} \cdot (1 - 1/n)^{3/2} = (n - 1)^{3/2}\,.$$

Thus $F'$ is a formula on $n'$-bit inputs of size $< n'^{3/2}$. Recall that for a given $b \in \{0, 1\}$ we have $b' = b \oplus c_i$. Let $y_{n'}^{b'} \triangleq R(1^{n'}, F', b')$ (Line 6). By the first condition in the induction hypothesis (Equation (7)) and the definition of each $y_n^b \triangleq y_{n'}^{b'} \cup y_i \mapsto c_i$, we have $|y_n^0|_\ell = |y_n^1|_\ell = n$. Below, we also rely on the last two conditions in the induction hypothesis (Equation (7)), Lemma 12 Item (ii), and the last condition in Equation (5). We derive the following statements, where $b \in \{0, 1\}$:

$$\begin{aligned} F'(y_{n'}^{b'}) &\neq \oplus^{b'}(y_{n'}^{b'})\,, \\ F(y_n^b) &= F'(y_{n'}^{b'})\,, \\ F(y_n^b) &\neq \oplus^{b'}(y_{n'}^{b'})\,. \end{aligned}$$

Notice that

$$\oplus^{b'}(y_{n'}^{b'}) = \oplus^{b \oplus c_i}(y_{n'}^{b'}) = c_i \oplus (\oplus^b(y_{n'}^{b'})) = c_i \oplus (\oplus^b(y_n^b) \oplus c_i) = \oplus^b(y_n^b)\,.$$

These statements imply that, for each $b \in \{0, 1\}$, $F(y_n^b) \neq \oplus^b(y_n^b)$. In other words, the conclusion of $\varphi(N)$ holds. This completes the proof of the induction step. ◀

As explained above, the provability of $\mathsf{Ref}_{R,s}$ in $\mathsf{S}^1_2$ implies its provability in $\mathsf{PV}_1$. Since $\mathsf{PV}_1 \vdash \mathsf{Ref}_{R,s} \rightarrow \mathsf{FLB}_s$, this completes the proof of Theorem 11. ◀

### 4.1.2 On the Low-Level Details of the Formalization

In order to make our presentation accessible to a broader audience, in this section we provide more details about the formalization of algorithms and about the proofs of their basic properties. However, due to space restriction, the section is included only in the full version.

## 4.2 Upper Bound

In this section, we show that the parity function on $n$ bits can be computed by formulas of size $O(n^2)$, provably in $\mathsf{PV}_1$. We can formalize this upper bound in the language of $\mathsf{PV}$, defining an $\mathcal{L}_{\mathsf{PV}}$-sentence stating that the parity function can be computed by a formula of size $s(n)$ for every input length $n \geq 1$:

$$\mathsf{FUB}_s \triangleq \forall N\, \forall n\, \exists F\, (n = |N| \geq 1 \land \mathsf{Size}(F) < s(n) \land \forall x\, (|x| \leq n \to \mathsf{Eval}(F, x) = \oplus_n^0(x))\,.$$

▶ **Theorem 14.** *Let $s(n) \triangleq 4n^2$. Then $\mathsf{PV}_1 \vdash \mathsf{FUB}_s$.*

**Proof.** $\mathsf{FUB}_s$ is a $\forall\Sigma_2^b$ sentence and our intended theory is $\mathsf{PV}_1$. In order to implement some inductive proofs, it will be helpful to reduce the complexity of the formula. For this, we introduce a new polynomial-time function, $\mathsf{ParForm}(1^{(n)})$, which generates the desired formula that computes the parity function on $n$ bits. Since it is a polynomial-time function, there is a symbol for it in $\mathsf{PV}$ and we can use it in the new formalization:

$$\begin{aligned}\mathsf{FUB}'_s \triangleq\, & \forall N\, \forall n\, (n = |N| \geq 1 \land \mathsf{Size}(\mathsf{ParForm}(1^{(n)})) < s(n) \land \\ & \forall x\, (|x| \leq n \to \mathsf{Eval}(\mathsf{ParForm}(1^{(n)}), x) = \oplus_n^0(x))\,.\end{aligned}$$

It is immediate that $\mathsf{FUB}'_s \Rightarrow \mathsf{FUB}_s$, thus we focus on proving $\mathsf{FUB}'_s$. We continue with the following steps:

1. We prove an upper bound of $n^2$ for the formulas calculating the parity function and its negation, when $n$ is a power of 2.
2. We use this construction to derive the $4n^2$ upper bound for any $n$.

Next, we define a polynomial-time algorithm $\mathsf{Par}(1^{(n)})$ which computes a formula that calculates the parity function on $n$ bits and a formula that calculates the negation of the parity function on $n$ bits, if $n$ is a power of 2.

**Algorithm 4** $\mathsf{Par}(1^{(n)})$ outputs Boolean formulas for $\oplus_n^0$ and $\oplus_n^1$ when $n$ is a power of 2.

```
Input: 1^(n) for some n ≥ 1.
Let k ≜ |n − 1|. If n ≠ 2^k (n is not a power of 2), then return "error";
// F will compute the parity function, while F̄ will compute its
   negation
if k = 0 then
    Define F to be the formula with one leaf x_1 and F̄ to be the formula with one leaf
      ¬x_1.
else if k ≥ 1 then
    // Construct a pair (F, F̄) of formulas on input bits x_1, ..., x_{2^k} as
       follows:
    Let (F_1, F̄_1) ≜ Par(1^(n/2)), and define a corresponding pair (F_2, F̄_2):
    In F_2 and F̄_2, relabel the leaves by putting x_{2^{k-1}+i} instead of x_i for every
      i = 1, ..., 2^{k-1};
    Now let F ≜ (F_1 ∨ F_2) ∧ (F̄_1 ∨ F̄_2) and F̄ ≜ (F_1 ∧ F_2) ∨ (F̄_1 ∧ F̄_2).
end
return (F, F̄).
```

▶ **Lemma 15.** *If $n$ is a power of 2, the algorithm $\mathsf{Par}(1^{(n)})$ correctly outputs two formulas $(F, \overline{F})$ of size $n^2$ which calculate the parity function and its negation, provably in $\mathsf{S}_2^1(\mathsf{PV})$.*

**Proof.** We split the proof of the correctness for the algorithm $\mathsf{Par}(1^{(n)})$ into 3 properties:

1. $\phi_1(n) \triangleq F, \overline{F} \in \mathsf{VALIDFORM}(n)$, where $\mathsf{VALIDFORM}(n)$ is the set of formulas on $n$ variables;
2. $\phi_2(n) \triangleq \mathsf{Size}(F) = \mathsf{Size}(\overline{F}) = n^2$;
3. $\phi_3(n) \triangleq \forall x\ |x| \leq n \to \mathsf{Eval}(F, x) = \oplus_n^0(x) \wedge \mathsf{Eval}(\overline{F}, x) = \oplus_n^1(x)$.

For now we only care about the case that $n$ is a power of 2, so we prove these properties conditionally (equivalently we prove $(n = (n-1)\#1) \to \phi(n)$).[7] That is why it suffices to use polynomial induction on $n$, which is available in $\mathsf{S}_2^1$, since our formulas are at most $\Pi_1^b$.

We skip the proof of $\phi_1$, which is proven by simple induction as below, using the fact that if $F_1, F_2$ are formulas then $F_1 \wedge F_2$ and $F_1 \vee F_2$ are also formulas.

**Property 2: $\mathsf{S}_2^1 \vdash \phi_2(n)$.** For the base case, $\phi_2(1)$, we have $k = 0$, which means that the output $(F, \overline{F}) \triangleq \mathsf{Par}(1^1)$ will be two formulas with one leaf each, hence

$$\mathsf{Size}(F) = \mathsf{Size}(\overline{F}) = 1.$$

For the induction step, we need $\mathsf{S}_2^1 \vdash \forall n\, \phi_2(\lfloor n/2 \rfloor) \to \phi_2(n)$. If $n$ is not a power of 2, then the statement is true by default. In the case of $n$ being a power of 2, we fix $k = |n-1|$ and we want to prove equivalently:

$$\mathsf{S}_2^1 \vdash \phi_2(2^{k-1}) \to \phi_2(2^k).$$

Assume that $\phi_2(2^{k-1}) \equiv \phi_2(n/2)$ holds. From Line 8 we have that

$$F = (F_1 \vee F_2) \wedge (\overline{F}_1 \vee \overline{F}_2) \text{ and } \overline{F} = (F_1 \wedge F_2) \vee (\overline{F}_1 \wedge \overline{F}_2), \tag{8}$$

where $(F_1, \overline{F_1})$ and $(F_2, \overline{F_2})$ are copies of $\mathsf{Par}(1^{n/2})$. From the induction hypothesis, this means that $\mathsf{Size}(F_1) = \mathsf{Size}(\overline{F_1}) = \mathsf{Size}(F_2) = \mathsf{Size}(\overline{F_2}) = (n/2)^2 = 2^{2(k-1)}$. Therefore, from (Equation (8)) and the properties of the function $\mathsf{Size}$, we get

$$\mathsf{Size}(F) = \mathsf{Size}(F_1) + \mathsf{Size}(\overline{F_1}) + \mathsf{Size}(F_2) + \mathsf{Size}(\overline{F_2}) = 4 \cdot 2^{2(k-1)} = 2^{2k} = n^2.$$

Similarly for $\overline{F}$, which means that $\phi_2(2^k) \equiv \phi_2(n)$ holds. This completes the proof of the induction for $\phi_2$.

**Property 3: $\mathsf{S}_2^1 \vdash \phi_3(n)$.** Here the base case is trivial: for $F \triangleq x_1$ and $x \in \{0, 1\}$, then $\mathsf{Eval}(F, x) = x = \oplus_1^0(x)$. Similarly for $\overline{F}$.

For the induction step, we assume as above that $n = 2^k$ and we want to prove:

$$\mathsf{S}_2^1 \vdash \phi_3(2^{k-1}) \to \phi_3(2^k).$$

We assume that $\phi_2(2^{k-1}) \equiv \phi_2(n/2)$ holds and we write $F$ in the form

$$F = (F_1 \vee F_2) \wedge (\overline{F}_1 \vee \overline{F}_2) \text{ and } \overline{F} = (F_1 \wedge F_2) \vee (\overline{F}_1 \wedge \overline{F}_2),$$

where $(F_1, \overline{F_1})$ and $(F_2, \overline{F_2})$ are copies of $\mathsf{Par}(1^{n/2})$. Therefore, instead of $\mathsf{Eval}(F, x)$, we can calculate

$$\mathsf{Eval}((F_1 \vee F_2) \wedge (\overline{F}_1 \vee \overline{F}_2), x).$$

[7] It is easy to check that this is true if and only if $n$ is a power of 2.

We need to prove that $\mathsf{Eval}(F, x) = \oplus_n^0(x)$ for all $x$ with $|x| \le n$. So, taking one such $x$ we can split its binary representation into two parts $x_1, x_2$ with lengths $|x_1|, |x_2| \le n/2$, such that $x = (x_2 x_1)_b = x_1 + 2^{n/2} x_2$.

The input to subformulas $F_2, \overline{F_2}$ from the definition are the bits $x_{2^k-1+i}$ for $i = 1, \ldots, 2^{k-1}$, which means that their input is $x_2$. Similarly, the input to subformulas $F_1, \overline{F_1}$ is $x_1$. Hence, we can define

$$b_1 \triangleq \mathsf{Eval}(F_1, x_1) b_3 \triangleq \mathsf{Eval}(\overline{F_1}, x_1)$$
$$b_2 \triangleq \mathsf{Eval}(F_2, x_2) b_4 \triangleq \mathsf{Eval}(\overline{F_2}, x_2)$$

From the properties of the evaluation function and the form of $F$, we can prove in $\mathsf{S}_2^1$ that $\mathsf{Eval}(F, x) = (b_1 \vee b_2) \wedge (b_3 \vee b_4)$, where the symbols $\vee, \wedge$ are used as Boolean symbols here.

However, since $|x_1|, |x_2| \le n/2$ and $(F_1, \overline{F_1}) = (F_2, \overline{F_2}) = \mathsf{Par}(1^{n/2})$, from the induction hypothesis we get that

$$b_1 = \oplus^0(x_1) b_3 = \oplus^1(x_1) = 1 - b_1$$
$$b_2 = \oplus^0(x_2) b_4 = \oplus^1(x_2) = 1 - b_2$$

Next, it is easy to prove by checking all the 4 cases that

$$\forall b_1, b_2 \in \{0, 1\} \; (b_1 \vee b_2) \wedge ((1 - b_1) \vee (1 - b_2)) = b_1 \oplus b_2,$$

and as a result, we get

$$\mathsf{Eval}(F, x) = (\oplus^0(x_1)) \oplus (\oplus^0(x_2)) = \oplus^0(x_2 x_1) = \oplus^0(x)$$

by the properties of the parity function. Similarly, we can prove that $\mathsf{Eval}(\overline{F}, x) = \oplus_n^1(x)$, which concludes the induction. ◀

For the general case, we use a simple padding argument. For a number $n$, we can define the number

$$\tilde{n} \triangleq (n - 1)\#1.$$

This number is the least power of 2 that is greater or equal to $n$. It is easy to see that

$$\mathsf{PV}_1 \vdash n \le \tilde{n} < 2n.$$

If we replace $\mathsf{ParForm}(1^{(n)})$ by $\mathsf{Par}_1(1^{\tilde{n}})$ (the first coordinate of $\mathsf{Par}(1^{\tilde{n}})$), we have by the above lemma that

1. $\mathsf{Size}(\mathsf{ParForm}(1^{(n)})) = \mathsf{Size}(\mathsf{Par}_1(1^{\tilde{n}})) = \tilde{n}^2 < (2n)^2 = s(n)$.
2. For all $x$ with $|x| \le n$, we have $|x| \le \tilde{n}$, which by the lemma gives us

   $$\mathsf{Eval}(\mathsf{ParForm}(1^n), x) = \mathsf{Eval}(\mathsf{Par}_1(1^{\tilde{n}}), x) = \oplus_{\tilde{n}}^0(x).$$

   Since $|x| \le n$, we also have $\oplus_{\tilde{n}}^0(x) = \oplus_n^0(x)$. Consequently, we have $\mathsf{Eval}(\mathsf{ParForm}(1^n), x) = \oplus_n^0(x)$.

These two together show that $\mathsf{PV}_1 \vdash \mathsf{FUB}'_s$ and the proof is complete. ◀

## 4.3 Formula Size Hierarchy

In this section, we provide the proof of Theorem 3.

▶ **Theorem 16** (Theorem 3). *Consider rationals $a > 2$ and $b = 3/2$, and let $n_0$ be a large enough positive integer. Then*

$$\mathsf{PV}_1 \vdash \mathsf{FSH}[a, b, n_0] .$$

**Proof.** We combine the results of Section 4.1 and Section 4.2. We argue in $\mathsf{PV}_1$. From Theorem 11, we get that

$$\forall n \in \mathsf{Log}\ \forall F \in \mathsf{FORMULA}[n^{3/2}]\ \exists x\ (|x| \leq n \land F(x) \neq \oplus_n(x)), \tag{9}$$

and from Theorem 14, we have that

$$\forall n \in \mathsf{Log}\ \exists G \in \mathsf{FORMULA}[4n^2]\ \forall x\ (|x| \leq n \to G(x) = \oplus_n(x)).$$

We can eliminate the constant 4 from the latter using that $a > 2$ and choosing a large enough $n_0$, such that for every $n \geq n_0$, $n^a \geq 4n^2$ (provably in $\mathsf{PV}_1$). Consequently,

$$\forall n \geq n_0 \in \mathsf{Log}\ \exists G \in \mathsf{FORMULA}[n^a]\ \forall x\ (|x| \leq n \to G(x) = \oplus_n(x)). \tag{10}$$

Finally, combining Equation (9) and Equation (10), we get that

$$\forall n \geq n_0 \in \mathsf{Log}\ \exists G \in \mathsf{FORMULA}[n^a]\ \forall F \in \mathsf{FORMULA}[n^{3/2}]\ \exists x\ (|x| \leq n\ \land\ F(x) \neq G(x)),$$

which is exactly the formula size hierarchy, $\mathsf{FSH}[a, b, n_0]$, for our choice of parameters $a > 2$ and $b = 3/2$. ◀

## References

1 Sanjeev Arora and Boaz Barak. *Computational Complexity – A Modern Approach*. Cambridge University Press, 2009. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.
2 Samuel R. Buss. *Bounded Arithmetic*. Bibliopolis, 1986.
3 Samuel R. Buss. Bounded arithmetic and propositional proof complexity. In *Logic of Computation*, pages 67–121. Springer Berlin Heidelberg, 1997.
4 Lijie Chen, Jiatu Li, and Igor C. Oliveira. Reverse mathematics of complexity lower bounds. In *Symposium on Foundations of Computer Science* (FOCS), 2024.
5 Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Relations and equivalences between circuit lower bounds and Karp-Lipton theorems. In *Computational Complexity Conference* (CCC), pages 30:1–30:21, 2019. doi:10.4230/LIPIcs.CCC.2019.30.
6 Alan Cobham. The intrinsic computational difficulty of functions. *Proc. Logic, Methodology and Philosophy of Science*, pages 24–30, 1965.
7 Stephen A. Cook. Feasibly constructive proofs and the propositional calculus (preliminary version). In *Symposium on Theory of Computing* (STOC), pages 83–97, 1975. doi:10.1145/800116.803756.
8 Stephen A. Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. Cambridge University Press, 2010. doi:10.1017/CBO9780511676277.
9 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on the Theory of Computing* (STOC), pages 220–229, 1997. doi:10.1145/258533.258590.
10 Emil Jeřábek. Dual weak pigeonhole principle, boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129(1-3):1–37, 2004. doi:10.1016/j.apal.2003.12.003.

11 Emil Jeřábek. *Weak pigeonhole principle and randomized computation*. PhD thesis, Charles University in Prague, 2005.

12 Emil Jeřábek. The strength of sharply bounded induction. *Mathematical Logic Quarterly*, 52(6):613–624, 2006. doi:10.1002/malq.200610019.

13 Emil Jeřábek. Approximate counting in bounded arithmetic. *Journal of Symbolic Logic*, 72(3):959–993, 2007. doi:10.2178/jsl/1191333850.

14 Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*. Springer, 2012. doi:10.1007/978-3-642-24508-4.

15 Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1):40–56, 1982. doi:10.1016/S0019-9958(82)90382-5.

16 Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1995.

17 Jan Krajíček. Small circuits and dual weak PHP in the universal theory of p-time algorithms. *ACM Transactions on Computational Logic* (TOCL), 22(2):1–4, 2021. doi:10.1145/3446207.

18 Jan Krajíček and Igor C. Oliveira. Unprovability of circuit upper bounds in Cook's theory PV. *Logical Methods in Computer Science*, 13(1), 2017. doi:10.23638/LMCS-13(1:4)2017.

19 Jan Krajíček, Pavel Pudlák, and Gaisi Takeuti. Bounded arithmetic and the polynomial hierarchy. *Annals of Pure and Applied Logic*, 52(1-2):143–153, 1991. doi:10.1016/0168-0072(91)90043-L.

20 Jan Krajíček. *Proof Complexity*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2019. doi:10.1017/9781108242066.

21 Alexis Maciel, Toniann Pitassi, and Alan R. Woods. A new proof of the weak pigeonhole principle. *Journal of Computer and System Sciences*, 64(4):843–872, 2002. doi:10.1006/jcss.2002.1830.

22 Moritz Müller and Ján Pich. Feasibly constructive proofs of succinct weak circuit lower bounds. *Annals of Pure and Applied Logic*, 171(2), 2020. doi:10.1016/j.apal.2019.102735.

23 Igor C. Oliveira. Meta-mathematics of computational complexity theory. *Preprint*, 2024.

24 Claude E. Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949. doi:10.1002/j.1538-7305.1949.tb03624.x.

25 Bella A. Subbotovskaya. Realization of linear functions by formulas using $+, \cdot, -$. In *Soviet Math. Dokl*, 1961.