On the Consistency of Circuit Lower Bounds for Non-Deterministic Time*

Albert Atserias[†] Sam Buss[‡] Moritz Müller[§]

March 5, 2023

Abstract

We prove the first unconditional consistency result for superpolynomial circuit lower bounds with a relatively strong theory of bounded arithmetic. Namely, we show that the theory V^0_2 is consistent with the conjecture that $\mathsf{NEXP} \notin \mathsf{P/poly}$, i.e., some problem that is solvable in non-deterministic exponential time does not have polynomial size circuits. We suggest this is the best currently available evidence for the truth of the conjecture. The same techniques establish the same results with NEXP replaced by the class of problems decidable in non-deterministic barely superpolynomial time such as $\mathsf{NTIME}(n^{O(\log\log\log n)})$. Additionally, we establish a magnification result on the hardness of proving circuit lower bounds.

*An extended abstract of part of this work will appear in the Proceedings of the 55th ACM Symposium on Theory of Computation (STOC 2023).

[†]Universitat Politècnica de Catalunya i Centre de Recerca Matemàtica, Barcelona, Spain. Supported in part by Project PID2019-109137GB-C22 (PROOFS) and the Severo Ochoa and María de Maeztu Program for Centers and Units of Excellence in R&D (CEX2020-001084-M) of the Spanish State Research Agency.

[‡]University of California, San Diego, USA. Supported in part by Simons Foundation grant 578919.

[§]Universität Passau, Passau, Germany.

1 Introduction

Bounded arithmetics are fragments of Peano arithmetic that formalize reasoning with concepts and constructions of bounded computational complexity. Their language is tailored so that natural classes of bounded formulas define important complexity classes. For example, the set of all bounded formulas defines precisely the problems in PH and the set of Σ^b_1-formulas those in NP. The central theories are comprised in Buss' hierarchy [5]

$$\mathsf{S}^1_2 \subseteq \mathsf{T}^1_2 \subseteq \mathsf{S}^2_2 \subseteq \mathsf{T}^2_2 \subseteq \cdots \subseteq \mathsf{T}_2 \subseteq \mathsf{V}^0_2 \subseteq \mathsf{V}^1_2 \tag{1}$$

The theory S^1_2 can be understood as formalizing P-reasoning, and V^1_2 as formalizing EXP-reasoning. The levels of T_2 are determined by induction schemes for properties of bounded computational complexity. E.g., T^1_2 has induction for NP, and T_2 for PH. Intuitively, these theories can construct and reason with polynomially large objects of various computational complexities. The theories V^0_2 and V^1_2 are extensions with a second sort of variables ranging over bounded sets of numbers and are given by comprehension schemes. Intuitively, these sets represent exponentially large objects.

Low levels of the bounded arithmetic hierarchy formalize a considerable part of contemporary complexity theory. This includes some advanced topics such as the Arthur-Merlin hierarchy [17], hardness amplification [16], or the PCP Theorem [30]. We refer to [26, Section 5] for a list of successful formalizations. Concerning circuit complexity, the topic of this paper, Jeřábek proved that his theory of approximate counting [15, 16, 17], which sits below T^2_2, formalizes Rabin's primality test, and proves that it is in $\mathsf{P/poly}$ [16, Example 3.2.10, Lemma 3.2.9]. Concerning lower bounds, many of the known (weak) circuit lower bounds can be formalized in a theory of approximate counting [26] and thus also in the theory T^2_2. For example, the AC^0 lower bound for parity has been formalized in [26, Theorem 1.1] via probabilistic reasoning with Furst, Saxe and Sipser's random restrictions [12], and in [22, Theorem 15.2.3] via Razborov's [32] proof of Håstad's switching lemma.

Razborov asked in his seminal work from 1995 for the "right fragment capturing the kind of techniques existing in Boolean complexity" [32, p.344]. Showing that any theory that is strong enough to capture these techniques cannot prove lower bounds for general circuits would give a precise sense in which current techniques are insufficient. This however seems to be very difficult. We refer to [34, Introduction] or [23, Ch.27-30] for a description of the resulting research program, and to [31] for a recent result.

In contrast to unprovability, the first and final words of Krajíček's 1995 monograph [22] ask for consistency results[1], namely to prove the conjecture in question "for nonstandard models of systems of bounded arithmetic". These are "not ridiculously pathological structures, and a part of the difficulty in constructing them stems exactly from the fact that it is hard to distinguish these structures, by the studied properties, from natural numbers" [22, p.xii]. In particular, showing that a given conjecture is consistent with certain bounded arithmetics, already low ones, would exhibit a world where both the conjecture and a considerable part of complexity theory are true.

[1] The citations to follow refer not to circuit lower bounds but to $\mathsf{P} \neq \mathsf{NP}$.

We therefore interpret consistency results as giving precise evidence for the *truth* of the conjecture. This is without doubt preferable to appealing to intuitions, or alluding to the experience that the conjectures appear to be theoretically coherent, exactly because a consistency result gives a precise meaning to this coherence.

1.1 Previous consistency results

Being well motivated, consistency results are also hard to come by, and not much is known. In particular, it is unknown whether $\mathsf{NP} \notin \mathsf{P/poly}$ is consistent with S^1_2.

It is not straightforward to formalize $\mathsf{NP} \notin \mathsf{P/poly}$ because exponentiation is not provably total in bounded arithmetics. On the formal level, call a number n *small* if 2^n exists. A size-n^c circuit can be coded by a binary string of length at most $10 \cdot n^c \cdot \log(n^c)$, and hence by a number below $2^{10 \cdot n^c \cdot \log(n^c)}$; this bound exists for small n.

On the formal level, an NP-problem is represented by a Σ^b_1-formula $\varphi(x)$. A sentence expressing that the problem defined by $\varphi(x)$ has size-n^c circuits looks as follows:

$$\alpha^c_\varphi := \ \forall n{\in}Log_{>1}\ \exists C{<}2^{n^c}\ \forall x{<}2^n\ (C(x){=}1 \leftrightarrow \varphi(x)).$$

Here, the quantifier on n ranges over small numbers above 1. We think of the quantifier on C as ranging over circuits of encoding-size n^c, and of the quantifier on x as ranging over length-n binary strings. Counting the $\exists$ hidden in φ, this is a bounded $\forall\exists\forall\exists$-sentence (namely a $\forall\Sigma^b_3$-sentence).

Now more precisely, the central question whether S^1_2 is consistent with $\mathsf{NP} \notin \mathsf{P/poly}$ asks for a Σ^b_1-formula $\varphi(x)$ such that $\mathsf{S}^1_2 + \left\{\neg\alpha^c_\varphi \mid c \in \mathbb{N}\right\}$ is consistent. As mentioned a model witnessing this consistency would be a world where a considerable part of complexity theory is true and the NP-problem defined by φ does not have polynomial-size circuits. This is faithful in that there also exists an NP-machine M that cannot be simulated by small circuits in the model. Namely, S^1_2 proves that $\varphi(x)$ is equivalent to a formula

$$\exists y{<}2^{n^d}\ \text{“}y \textit{ is an accepting computation of } M \textit{ on } x\text{”} \tag{2}$$

for a suitable NP-machine M, namely a *model-checker* for φ. Here, the constant d stems from the polynomial running time of M. We write $\alpha^c_M := \alpha^c_\varphi$ for $\varphi(x)$ equal to (2). One can also fix the machine M in advance to a *universal* one, namely a model-checker M^* for an S^1_2-provably NP-complete problem (e.g., SAT).

The predominant approach to the consistency of circuit lower bounds is based on witnessing theorems: a proof of α^c_M in some bounded arithmetic implies a low-complexity algorithm that computes a witness C from 1^n. E.g., if the theory has feasible witnessing in P, then it does not prove α^c_φ for any c unless the problem defined by $\varphi(x)$ is in P. However, S^1_2 is only known to have feasible witnessing in P for bounded $\forall\exists$-sentences and α^c_φ is a $\forall\exists\forall\exists$-sentence.

Fortunately, a self-reducibility argument implies that the quantifier complexity of this formula can be reduced. Up to suitable changes of c, the formula $\alpha^c_{M^*}$ is S^1_2-provably equivalent to the following sentence of lower quantifier complexity:

$$\beta^c_{M^*} := \begin{array}{l} \forall n \in Log_{>1}\ \exists C < 2^{n^c}\ \exists D < 2^{n^c}\ \forall x < 2^n\ \forall y < 2^{n^d} \\ \quad (C(x)=0 \to \neg\,\text{“}y \textit{ is an accepting computation of } M^* \textit{ on } x\text{”}) \wedge \\ \quad (C(x)=1 \to \text{“}D(x) \textit{ is an accepting computation of } M^* \textit{ on } x\text{”}), \end{array}$$

where d stems from the polynomial runtime of M^*. We define

$$\text{“}\mathsf{NP} \notin \mathsf{P/poly}\text{”} := \left\{\neg\beta^c_{M^*} \mid c \in \mathbb{N}\right\}.$$

Note, $\beta^c_{M^*}$ is a bounded $\forall\exists\forall$-sentence (namely a $\forall\Sigma^b_2$-sentence). For such sentences, S^2_2 has feasible witnessing in P^{NP} [5], and S^1_2 has feasible witnessing by certain interactive polynomial-time computations [21]. This was exploited by Cook and Krajíček [10] to prove[2] that "$\mathsf{NP} \notin \mathsf{P/poly}$" is consistent with S^2_2 unless $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}}$, and with S^1_2 unless $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}}_{\mathrm{tt}}$. Since the complexity of witnessing increases with the strength of the theory, it seems questionable whether this method yields insights for much stronger theories: by the Karp-Lipton Theorem [19], $\mathsf{PH} \notin \mathsf{NP}^{\mathsf{NP}}$ implies that "$\mathsf{NP} \notin \mathsf{P/poly}$" is true, and true sentences are consistent with any true theory. Moreover, the focus of this work is on unconditional consistency results.

Using similar methods, a recent line of works [24, 6, 7, 8] achieved unconditional consistency results for fixed-polynomial lower bounds, even for P instead of NP (based on [36]). For example, the main result in [7] implies that $\mathsf{S}^2_2 + \neg\alpha^c_\varphi$ and $\mathsf{S}^1_2 + \neg\alpha^c_\psi$ are consistent for certain formulas $\varphi(x)$ and $\psi(x)$ that define problems in P^{NP} and NP, respectively. Again it seems questionable whether the underlying methods can yield insights for much stronger theories: by Kannan [18], the lower bound stated by $\neg\alpha^c_\chi$ is true for some formula $\chi(x)$ defining a problem in $\mathsf{NP}^{\mathsf{NP}}$. Moreover, the formulas above depend on c and new ideas seem to be required to reach the unconditional consistency of superpolynomial lower bounds.

1.2 New consistency results

The purpose of this paper is to prove the unconditional consistency of $\mathsf{NEXP} \notin \mathsf{P/poly}$ with the comparatively strong theory V^0_2. Consistency results for V^0_2 are meaningful, since V^0_2 is stronger than T^2_2 which, as discussed earlier, can formalize many results in complexity theory. Our approach is not via witnessing but via *simulating comprehension.* We explain this idea in a simple setting.

The problems in NEXP are naturally represented on the formal level by $\hat{\Sigma}^{1,b}_1$-formulas $\varphi(x)$: an existentially quantified set variable followed by a bounded formula. Then, the following is a direct formalization of the consistency of $\mathsf{NEXP} \notin \mathsf{P/poly}$:

Proposition 1. *There exists $\varphi(x) \in \hat{\Sigma}^{1,b}_1$ such that $\mathsf{V}^0_2 + \left\{\neg\alpha^c_\varphi \mid c \in \mathbb{N}\right\}$ is consistent.*

[2] $\mathsf{P}^{\mathsf{NP}}_{\mathrm{tt}}$ denotes polynomial time with non-adaptive queries to an NP-oracle. In [10] a distinct but similar formalization of $\mathsf{NP} \notin \mathsf{P/poly}$ is used.

In hindsight this is not hard to prove. For $\varphi(x)$ take a formula negating the pigeonhole principle: it states that there exists a set coding an injection from $\{0,\dots,x{+}1\}$ into $\{0,\dots,x\}$. If this formula were computed by circuits, then we could use quantifier-free induction to show that the pigeonhole principle is provable in V^0_2. But it is well known that this is not the case (see [22, Corollary 12.5.5]).

Concerning the faithfulness of the direct formalization we get, as before, a model of V^0_2 where a certain NEXP-machine cannot be simulated by small circuits. For a NEXP-machine M we can write the formula (2) using instead of $\exists y$ a quantification $\exists Y$ for a set variable Y. It turns out that V^0_2 proves that every $\hat{\Sigma}^{1,b}_1$-formula $\varphi(x)$ is equivalent to such a formula for a suitable NEXP-machine M, namely a model-checker for $\varphi(x)$. Proving this is not trivial because V^0_2 is agnostic about the existence of computations of exponential-time machines. One of the contributions of this work is to prove it; we give the details in Section 3.

Intuitively, V^0_2 does not know whether non-trivial exponential-size sets exist, namely sets not given by bounded formulas. But then, how meaningful is the above consistency? Counting only set quantifiers, α^c_M is equivalent to the conjunction of a $\forall$- and a $\exists$-sentence, so α^c_M does possibly assert the existence of non-trivial large sets. It turns out that we can move again to a suitably modified sentence β^c_M of lower quantifier complexity, namely a $\forall$-sentence (i.e., $\forall\Pi^{1,b}_1$): such sentences do not entail the existence of non-trivial large sets. This does not follow from simple self-reducibility arguments but is a deep result of complexity theory, namely the Easy Witness Lemma of Impagliazzo, Kabanets and Wigderson [14, Theorem 31]. We use Williams' version as stated in [38, Lemma 3.1] (see [39, Theorem 3.1] for the equivalence):

Lemma 2 (Easy Witness Lemma). *If* $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, *then every* NEXP*-machine has polynomial-size oblivious witness circuits.*

An *oblivious witness circuit* for a machine M and input length n is a circuit D with at least n inputs such that for every x of length n, if M accepts x, then $tt(D_x)$ encodes an accepting computation of M on x. Here, the circuit D_x is obtained from D by fixing the first n inputs to the bits of x, and $tt(D_x)$ is the truth table of D_x. In the statement of the lemma, *polynomial-size* refers to polynomial in n, and the qualifier *oblivious* refers to the fact that D does not depend on x.

In the language of two-sorted bounded arithmetic the string $tt(D_x)$ corresponds to the set $D_x(\cdot)$ of numbers accepted by D_x. Hence, for an NEXP-machine M we define β^c_M by replacing $D(x)$ by $D_x(\cdot)$ and $\forall y$ by $\forall Y$:

$$\begin{aligned}\beta^c_M \;:=\; & \forall n{\in}Log_{>1}\ \exists C{<}2^{n^c}\ \exists D{<}2^{n^c}\ \forall x{<}2^n\ \forall Y\\ & (C(x){=}0 \;\to\; \neg\text{“}Y \textit{ is an accepting computation of } M \textit{ on } x\text{”}) \wedge\\ & (C(x){=}1 \;\to\; \text{“}D_x(\cdot) \textit{ is an accepting computation of } M \textit{ on } x\text{”}).\end{aligned}$$

We define

$$\text{“}\mathsf{NEXP} \notin \mathsf{P/poly}\text{”} := \{\neg\beta^c_{M_0} \mid c \in \mathbb{N}\}$$

for a suitable universal NEXP-machine M_0. We stress that Lemma 2 and the fact that $\alpha^c_{M_0}$ and $\beta^c_{M_0}$ are equivalent up to a suitable change of c are argued outside the theories we consider. In particular, the proof of their equivalence depends on the Easy Witness Lemma and we do not know if the Easy Witness Lemma is provable in V^0_2.

The purpose of this paper is to prove:

Theorem 3. V^0_2 + "$\mathsf{NEXP} \notin \mathsf{P/poly}$" *is consistent.*

We emphasize here that our formalization of $\mathsf{NEXP} \notin \mathsf{P/poly}$ through the universal machine M_0 and the $\beta^c_{M_0}$ sentences refers exclusively to the setting of non-relativized complexity classes.

Second we show that NEXP can be lowered to just above NP. For $k \in \mathbb{N}$, define $\log^{(k)} n$ inductively by $\log^{(1)} n := \log n$, and $\log^{(k+1)} n := \log\log^{(k)} n$. We prove:

Theorem 4. V^0_2 + "$\mathsf{NTIME}(n^{O(\log^{(k)} n)}) \notin \mathsf{P/poly}$" *is consistent for every positive* $k \in \mathbb{N}$.

The formalization and proof proceeds similarly and relies on an Easy Witness Lemma for barely superpolynomial time by Murray and Williams [27]. Theorem 4 "almost" settles the central question for the consistency of $\mathsf{NP} \notin \mathsf{P/poly}$ with a strong bounded arithmetic. Closing the tiny gap, however, seems to require some new ideas.

1.3 Simulating comprehension

The idea behind our approach to the consistency of circuit lower bounds is the following. By the Easy Witness Lemma, the inclusion $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ implies that a rich collection of sets is represented by circuits (via their truth tables). A weak theory can quantify over circuits and hence implicitly over this collection. Thus, intuitively, $\beta^c_{M_0}$ should enable a weak theory to simulate a two-sorted theory of considerable strength. More precisely, we show that $\beta^c_{M_0}$ can be used to simulate a considerable fragment of $\Sigma^{1,b}_1$-comprehension, i.e., a considerable fragment of V^1_2. In particular, the simulated fragment proves the pigeonhole principle. This implies Theorem 3 as it is well-known that V^0_2 cannot prove this principle. Thereby, Theorem 3 is ultimately based on the exponential lower bound for the propositional translation of this principle in bounded depth Frege systems [1, 3]. On a high level, while the approach based on witnessing uses complexity theoretic methods, our approach is based on methods from mathematical logic, in particular forcing (cf. [2]).

The sketched idea can be encapsulated as follows. By $\mathsf{S}^1_2(\alpha)$ we denote the two-sorted variant of S^1_2. Its models consist of two universes M and $\mathcal{X}$ interpreting the number and the set sort, respectively. Given such a model that additionally satisfies $\beta^c_{M_0}$ for some $c \in \mathbb{N}$, we show that shrinking $\mathcal{X}$ to the sets represented by circuits in M yields a model of V^1_2. This has two interesting corollaries. The first is:

Corollary 5. *Let* T *be a theory that contains* $\mathsf{S}^1_2(\alpha)$ *but does not prove all number-sort consequences of* V^1_2*. Then* T + "$\mathsf{NEXP} \notin \mathsf{P/poly}$" *is consistent.*

By a *number-sort* formula we mean one that does not use set-sort variables. Note that the corollary refers to number-sort sentences of arbitrary unbounded quantifier complexity. It is conjectured that V^1_2 has more number-sort consequences than all other theories mentioned so far. But this is known only for S^1_2 [37, 20], and there even for $\forall\Pi^b_1$-sentences. Corollary 5 directly infers evidence for the truth of "$\mathsf{NEXP} \notin \mathsf{P/poly}$" from progress in mathematical logic on understanding independence. Loosely speaking, we view it in line with the belief that it is mathematical logic that ultimately bears on fundamental complexity-theoretic conjectures (see e.g. again the preface of [22]).

The second corollary is:

Corollary 6. *If $\mathsf{S}^1_2(\alpha)$ does not prove "$\mathsf{NEXP} \notin \mathsf{P/poly}$", then V^1_2 does not prove "$\mathsf{NEXP} \notin \mathsf{P/poly}$".*

This is a *magnification result* on the hardness of proving circuit lower bounds: it infers strong hardness (for V^1_2) from weak hardness (for $\mathsf{S}^1_2(\alpha)$). The term magnification has been coined in [28] in the context of circuit lower bounds where such results are currently intensively investigated (cf. [9]). In proof complexity such results are rare so far. An example in propositional proof complexity appears in [26, Proposition 4.14]. Magnification results are interesting because they reveal inconsistencies in common beliefs about what is and what is not within the reach of currently available techniques. Corollary 6 might foster hopes to complete Razborov's program to find a precise barrier in circuit complexity (cf. Remark 41).

2 Consistency of the direct formalization

In this section we provide the details of the simple proof of Proposition 1. We begin by recalling the necessary preliminaries on bounded arithmetic. This will be needed also in later sections. We refer to [22, Ch.5] for the missing details.

2.1 Preliminaries: bounded arithmetic

Bounded arithmetics have language $x{\leqslant}y$, 0, 1, $x{+}y$, $x{\cdot}y$, $\lfloor x/2\rfloor$, $x\#y$, $|x|$, and built-in equality $x{=}y$. Note that Cantor's pairing $\langle x,y\rangle$ is given by a term. Iterating it gives $\langle x_1,\ldots,x_k\rangle$ for $k>2$. A number x is called *small* if it satisfies the formula $\exists y\ x{=}|y|$. We abbreviate $\exists y\ x{=}|y|$ by $x{\in}Log$ and $x{\in}Log \wedge 1{<}x$ by $x{\in}Log_{>1}$. The quantifiers $\forall x{\in}Log_{>1}$ and $\exists x{\in}Log_{>1}$ range over small numbers above 1. If $x=|y|$, we write 2^x for $1\#y$ and similarly for other exponential functions. E.g., a formula of the form $\forall x{\in}Log_{>1}\ \ldots\ 2^{x^2}\ldots$ stands for the formula $\forall x\forall y\ (1{<}x \wedge x{=}|y| \to \ldots\ y\#y\ldots)$.

Theories The theories of bounded arithmetic are given by a set BASIC of universal sentences determining the meaning of the symbols, plus induction schemes. For a set of formulas Φ, the set (of the universal closures) of formulas

$$\varphi(\bar{x},0) \wedge \forall y{<}z\ (\varphi(\bar{x},y) \to \varphi(\bar{x},y+1)) \to \varphi(\bar{x},z),$$

for $\varphi\in\Phi$, is the scheme of Φ-*induction*. Restricting to small numbers z gives the scheme of Φ-*length induction*; formally, replace z by $|z|$ above. Here, and throughout, when writing a formula ψ as $\psi(\bar{x})$ we mean that *all* free variables of ψ are among $\bar{x}$.

The set Σ^b_∞ contains all bounded formulas, and Σ^b_i, Π^b_i, for $i \in \mathbb{N}$, are subsets of Σ^b_∞ defined by counting alternations of bounded quantifiers $\exists x{\leqslant}t, \forall x{\leqslant}t$, not counting sharply bounded ones $\exists x{\leqslant}|t|, \forall x{\leqslant}|t|$. In particular, $\Sigma^b_0 = \Pi^b_0$ is the set of sharply bounded formulas. The theories T^i_2 are defined by $\mathsf{BASIC}+\Sigma^b_i$-induction. The theories S^i_2 are defined by $\mathsf{BASIC}+\Sigma^b_i$-length-induction. Full bounded arithmetic $\mathsf{T}_2 \coloneqq \bigcup_{i\in\mathbb{N}} \mathsf{T}^i_2$ has Σ^b_∞-induction.

Two-sorted theories Two-sorted bounded arithmetics are obtained by adding a new set of variables $X, Y, \ldots$ of the *set sort*. Original variables $x, y, \ldots$ are of the *number sort*. We shall use capital letters also for number-sort variables. Therefore, for clarity, from now on we write $\exists_2 X$ and $\forall_2 X$ for quantifiers on set-sort variables X. The language is enlarged by adding a binary relation $x{\in}X$ between the number and the set sort. A *number-sort* formula is one that uses only the number sort. In particular, it has no set-sort parameters. By a *term* we mean a term in the number sort. We write $X{\leqslant}z$ for $\forall y\ (y{\in}X \to y{\leqslant}z)$.

Models have the form $(M, \mathcal{X})$ where M is a universe for the number sort and $\mathcal{X}$ is a universe for the set sort. The symbol $\in$ is interpreted by a subset of $M \times \mathcal{X}$. The standard model is $(\mathbb{N}, [\mathbb{N}]^{<\omega})$ where $[\mathbb{N}]^{<\omega}$ is the set of finite subsets of $\mathbb{N}$; the number sort symbols are interpreted as usual over $\mathbb{N}$ and $\in$ by actual element-hood.

The sets $\Sigma^b_\infty(\alpha), \Sigma^b_i(\alpha), \Pi^b_i(\alpha)$ are defined as $\Sigma^b_\infty, \Sigma^b_i, \Pi^b_i$, allowing free set-variables and the symbol $\in$, but not allowing set-sort quantifiers, nor set-sort equalities $X{=}Y$. Another name for the set $\Sigma^b_\infty(\alpha)$ is $\Sigma^{1,b}_0$. The theories $\mathsf{T}^i_2(\alpha)$, $\mathsf{S}^i_2(\alpha)$, and $\mathsf{T}_2(\alpha)$, are given by BASIC and analogous induction schemes as before, namely $\Sigma^b_i(\alpha)$-induction, $\Sigma^b_i(\alpha)$-length induction, and $\Sigma^b_\infty(\alpha)$-induction, respectively. Additionally, we add the following axioms with the set sort. Recalling the notation $X{\leqslant}z$ introduced above, the new axioms are (the universal closures of):

set-boundedness axiom: $\exists z\ X{\leqslant}z$.
extensionality axiom: $X{\leqslant}z \wedge Y{\leqslant}z \wedge \forall y{\leqslant}z\ (y{\in}X \leftrightarrow y{\in}Y) \to X{=}Y$.

We add the scheme of (bounded) $\Delta^b_1(\alpha)$-*comprehension*, given by (the universal closures of) the formulas

$$\exists_2 Y{\leqslant}z\ \forall y{\leqslant}z\ \big(y \in Y \leftrightarrow \varphi(\bar{X}, \bar{x}, y)\big), \tag{3}$$

where $\varphi(\bar{X}, \bar{x}, y)$ is $\Delta^b_1(\alpha)$ with respect to the theory defined over the two-sorted language as BASIC plus $\Sigma^b_1(\alpha)$-length-induction, i.e., this theory proves $\varphi(\bar{X}, \bar{x}, y)$ equivalent to both a $\Pi^b_1(\alpha)$-formula and a $\Sigma^b_1(\alpha)$-formula.

For example, this scheme implies that there is a set Y as described when $\varphi(\bar{X}, \bar{x}, y)$ is $f^{\bar{X}}(\bar{x}, y){=}1$ where $f^{\bar{X}}(\bar{x}, y)$ is a function that is $\Sigma^b_1(\alpha)$-definable in $\mathsf{S}^1_2(\alpha)$. The superscript indicates that $\bar{X}$ comprises all the free variables of the set sort that appear in the $\Sigma^b_1(\alpha)$-formula that defines $f^{\bar{X}}(\bar{x}, y)$. It is well known [5] that these are precisely the functions that are computable in polynomial time with oracles denoted by the set variables. We do not distinguish S^1_2 (or $\mathsf{S}^1_2(\alpha)$) from its variant in the language PV (resp., $\mathsf{PV}(\alpha)$) which has a symbol for all polynomial time functions (resp., with oracles denoted by the set variables). We shall often use that $\mathsf{S}^1_2(\alpha)$ proves induction for quantifier-free $\mathsf{PV}(\alpha)$-formulas (cf. [22, Lemma 5.2.9]). We write quantifier-free $\mathsf{PV}(\alpha)$-formulas with latin capital letters; e.g., $F(\bar{X}, \bar{x})$.

A piece of notation For formulas $\varphi(Y,\bar{X},\bar{x})$ and $\psi(\bar{Z},\bar{z},u)$ we write

$$\varphi\big(\psi(\bar{Z},\bar{z},\cdot),\bar{X},\bar{x}\big)$$

for the formula obtained from φ by replacing every atomic subformula of the form $t\in Y$, for t a term, by the formula $\psi(\bar{Z},\bar{z},t)$, preceded by any necessary renaming of the bound variables of φ to avoid the capturing of free variables. We use this notation only for formulas φ without set equalities.

Genuine two-sorted theories It is easy to see that the theories $\mathsf{T}^i_2(\alpha),\mathsf{S}^i_2(\alpha)$ have the same number sort consequences as $\mathsf{T}^i_2,\mathsf{S}^i_2$, respectively. Also $\mathsf{T}^i_2(\alpha),\mathsf{S}^i_2(\alpha)$ are conservative over their subtheories without $\Delta^b_1(\alpha)$-comprehension. Intuitively, the two-sorted versions of bounded arithmetics are the usual ones plus syntactic sugar. Genuine set-sorted theories are obtained from $\mathsf{T}_2(\alpha)$ by adding *(bounded)* Φ*-comprehension* for certain sets of formulas Φ, i.e., (3) for $\varphi(\bar{X},\bar{x},y)$ in Φ.

The set $\Sigma^{1,b}_\infty$ contains all two-sorted formulas with quantifiers of both sorts, but bounded number-sort quantifiers. Again we disallow set equalities. The sets $\Sigma^{1,b}_i,\Pi^{1,b}_i$, for $i\in\mathbb{N}$, are subsets of $\Sigma^{1,b}_\infty$ defined by counting the alternations of set quantifiers (and not counting number quantifiers). A $\hat{\Sigma}^{1,b}_1$-formula is of the form

$$\exists_2 Y\ \varphi(\bar{X},Y,\bar{x}) \tag{4}$$

where $\varphi(\bar{X},Y,\bar{x})$ is a $\Sigma^{1,b}_0$-formula.

For $i\in\mathbb{N}$ the theory V^i_2 is given by $\Sigma^{1,b}_i$-comprehension. In particular, V^0_2 is given by $\Sigma^{1,b}_0$-comprehension. It has the same number-sort consequences as T_2.

Remark 7. Sometimes, the sets $\Sigma^{1,b}_i(\alpha)$ are defined with bounded set quantifiers $\exists X\leqslant t$ and $\forall X\leqslant t$. The difference is not essential: for every $\Sigma^{1,b}_\infty$-formula $\varphi(\bar{X},Y,\bar{x})$ there is a term $t(\bar{x})$ such that $\mathsf{S}^1_2(\alpha)$ proves

$$t(\bar{x})\leqslant y\to\big(\varphi(\bar{X},Y,\bar{x})\leftrightarrow\varphi(\bar{X},Y^{\leqslant y},\bar{x})\big)$$

where $Y^{\leqslant y}$ stands for $\psi(Y,y,\cdot)$ with $\psi(Y,y,u):=(u\leqslant y\wedge u\in Y)$. By $\Delta^b_1(\alpha)$-comprehension, $\exists_2 Y\varphi$ is $\mathsf{S}^1_2(\alpha)$-provably equivalent to $\exists_2 Y\leqslant t(\bar{x})\ \varphi$. It follows that every $\Sigma^{1,b}_i(\alpha)$-formula is $\mathsf{S}^1_2(\alpha)$-provably equivalent to one with bounded set sort quantifiers.

Remark 8. Disallowing set equalities is convenient but inessential in the sense that V^i_2 does not change when set equalities are allowed in $\Sigma^{1,b}_i$. Indeed, let $\varphi(\bar{X},\bar{x})$ be a $\Sigma^{1,b}_i$-formula except that set equalities are allowed. Then there is a $\Sigma^{1,b}_i$-formula $\varphi^*(\bar{X},\bar{x},u)$ (without set equalities and) with bounded set quantifiers such that $\mathsf{S}^1_2(\alpha)$ proves

$$\exists u\ \big(\varphi(\bar{X},\bar{x})\leftrightarrow\varphi^*(\bar{X},\bar{x},u)\big).$$

Proof. The formula φ^* is defined by a straightforward recursion on φ. For example, if φ is $X_1{=}X_2$, then φ^* is $\forall y\leqslant u\ (y\in X_1\to y\in X_2)\wedge\forall y\leqslant u\ (y\in X_2\to y\in X_1)$; a u witnessing the equivalence is any common upper bound on X_1 and X_2. If φ is $\exists_2 Y\psi(\bar{X},Y,\bar{x})$ and $\psi^*=\psi^*(\bar{X},Y,\bar{x},u)$ is already defined, then φ^* is $\exists_2 Y\leqslant t(\bar{x},u)\ \psi^*(\bar{X},Y,\bar{x},u)$ where the term t is chosen according to the previous remark. $\square$

Circuits A circuit with s gates is coded by a number below $2^{10\cdot s\cdot|s|}$. On the formal level we shall only consider small circuits, i.e., $s \in Log$, so $2^{10\cdot s\cdot|s|}$ exists. We use capital letters C, D, E for number variables when they are intended to range over circuits. There is a PV-function $eval(C, x)$ that (in the standard model) takes a circuit C with, say, $n \leqslant |C|$ input gates, and evaluates it on inputs $x < 2^n$. This means that the input gates of C are assigned the bits of the length-n binary representation of x; we assume $eval(C, x) = 0$ if $x \geqslant 2^n$ or if C does not code a circuit.

It is notationally convenient to have circuits that take finite tuples $\bar{x} = (x_1, \dots, x_k)$ as inputs; formally, such a circuit has k sequences of input gates, the i-th taking the bits of x_i. Again, $eval(C, \bar{x})$ denotes the evaluation function; it outputs 0 if any x_i has length bigger than the length its allotted input sequence. Our circuits have exactly one output gate, so S^1_2 proves $eval(C, \bar{x}){<}2$. We write $C(\bar{x})$ for the quantifier-free PV-formula $eval(C, \bar{x}){=}1$; in some places we also write $C(\bar{x}){=}1$ and $C(\bar{x}){=}0$ instead of $C(\bar{x})$ and $\neg C(\bar{x})$, respectively.

For a circuit C taking $(\ell + k)$-tuples as inputs and an ℓ-tuple $\bar{x}$ we let $C_{\bar{x}}$ be the circuit obtained by fixing the first ℓ inputs to $\bar{x}$; it takes k-tuples as inputs. Formally, $C_{\bar{x}}$ is a PV-term with variables $C, \bar{x}$ and $\mathsf{S}^1_2(\alpha)$ proves $(C_{\bar{x}}(\bar{y}) \leftrightarrow C(\bar{x}, \bar{y}))$ and $|C_{\bar{x}}|{\leqslant}|C|$.

Lemma 9. *For every quantifier-free PV-formula $F(\bar{x})$ there is a $c \in \mathbb{N}$ such that S^1_2 proves*

$$\forall n{\in}Log_{>1}\ \exists C{<}2^{n^c}\ \forall \bar{x}{<}2^n\ \big(C(\bar{x}) \leftrightarrow F(\bar{x})\big).$$

On the formal level, if Y is a set and C is a circuit, then we say that Y is *represented* by C if $\forall y\ (C(y) \leftrightarrow y{\in}Y)$. In our notation, such set Y is written $C(\cdot)$, or $eval(C, \cdot){=}1$. More precisely, for a formula $\varphi(Y, \bar{X}, \bar{x})$ and a circuit C we write

$$\varphi\big(C(\cdot), \bar{X}, \bar{x}\big),$$

for the formula obtained from φ by replacing every formula of the form $t{\in}Y$ by $C(t)$, i.e., by $eval(C, t){=}1$. Note that if the set Y is represented by a circuit with n inputs, then $Y{<}2^n$, provably in S^1_2. For example, we shall use circuits to represent computations of exponential-time machines M. Using the notation introduced in Section 3.1,

$$\text{“}C(\cdot)\ \textit{is a halting computation of}\ M\ \textit{on}\ \bar{x}\text{”}$$

is a Π^b_1-formula with free variables $C, \bar{x}$ stating that the circuit C represents a halting computation of M on $\bar{x}$.

2.2 Direct consistency for NEXP

The set of $\hat{\Sigma}^{1,b}_1$-formulas without free variables of the set sort is a natural class of formulas defining, in the standard model, all the problems in NEXP. For such a formula ψ it is straightforward to write down a set of sentences (a.k.a. a theory) stating that ψ does not have polynomial size circuits. We explicitly define this formalization of $\mathsf{NEXP} \notin \mathsf{P/poly}$ as the set of all sentences of the form $\neg\alpha^c_\psi$, for $c \in \mathbb{N}$, for the sentence α^c_ψ defined in the introduction, and then argue that its consistency with V^0_2 follows from known lower bounds in proof complexity.

We repeat the definition of α^c_ψ from the introduction:

Definition 10. Let $c \in \mathbb{N}$ and let $\psi = \psi(x)$ be a $\hat{\Sigma}_1^{1,b}$-formula (with only one free variable x, and in particular without free variables of the set sort). Define

$$\alpha_\psi^c \;:=\; \forall n{\in}Log_{>1}\;\exists C{\leqslant}2^{n^c}\,\forall x{<}2^n\;\big(C(x) \leftrightarrow \psi(x)\big).$$

We are ready to prove Proposition 1.

Proof of Proposition 1: The *(functional) pigeonhole principle* $PHP(x)$ is the following $\Pi_1^{1,b}$-formula:

$$\begin{aligned}\forall_2 X\;\big(&\exists y{\leqslant}x{+}1\;\forall z{\leqslant}x\;\neg\langle y,z\rangle{\in}X\;\vee\\ &\exists y{\leqslant}x{+}1\;\exists z{\leqslant}x\;\exists z'{\leqslant}x\;(\neg z{=}z' \wedge \langle y,z\rangle{\in}X \wedge \langle y,z'\rangle{\in}X)\;\vee\\ &\exists y{\leqslant}x{+}1\;\exists y'{\leqslant}x{+}1\;\exists z{\leqslant}x\;(\neg y{=}y' \wedge \langle y,z\rangle{\in}X \wedge \langle y',z\rangle{\in}X)\big).\end{aligned}$$

Note that $\psi = \psi(x) := \neg PHP(x)$ is (logically equivalent to) a $\hat{\Sigma}_1^{1,b}$-formula. For the sake of contradiction assume that $\mathsf{V}_2^0 + \big\{\neg\alpha_\psi^c \mid c \in \mathbb{N}\big\}$ is inconsistent. By compactness, there exists $c \in \mathbb{N}$ such that V_2^0 proves α_ψ^c.

Claim: $\mathsf{V}_2^0 + \alpha_\psi^c$ proves $PHP(x)$.

The claim implies the theorem: it is well known [22, Corollary 12.5.5] that there is an expansion (M, R^M) of a model M of BASIC by an interpretation $R^M \subseteq M$ of a new predicate R such that R^M is bounded and witnesses $\neg PHP(n)$ for some (nonstandard) $n \in M$, and, further, (M, R^M) models induction for bounded formulas. Let $\mathcal{Y}$ be the collection of bounded sets definable in (M, R^M) by bounded formulas. Then $(M, \mathcal{Y})$ is a model of V_2^0 with $R^M \in \mathcal{Y}$, so $(M, \mathcal{Y}) \models \neg PHP(n)$.

We are left to prove the claim. Argue in V_2^0 and set $n := \max\{|x|, 2\}$. Then α_ψ^c gives a circuit C such that

$$\forall u{\leqslant}x\;(\neg C(u) \leftrightarrow PHP(u)).$$

We observe that V_2^0 proves that $PHP(x)$ is inductive, i.e.,

$$PHP(0) \wedge \forall u{<}x\;(PHP(u) \to PHP(u+1)). \tag{5}$$

Indeed, if X is a set that witnesses $\neg PHP(u+1)$, then we construct a set Y that witnesses $\neg PHP(u)$ as follows. If there does not exist any $v{\leqslant}u{+}1$ with $\langle v,u\rangle{\in}X$, then the set $Y := X$ itself is the witness we want. On the other hand, if there exists $v{\leqslant}u{+}1$ with $\langle v,u\rangle{\in}X$, then let Y be the set of pairs $z = \langle x,y\rangle$ such that the two projections $x = \pi_1(z)$ and $y = \pi_2(z)$ satisfy the formula $\varphi(x,y,u,v)$ below, for the fixed parameters u and v:

$$\varphi(x,y,u,v) := x{\leqslant}u \wedge y{<}u \wedge \big((x{>}v \wedge \langle x{-}1,y\rangle{\in}X) \vee (x{<}v \wedge \langle x,y\rangle{\in}X)\big).$$

Here, $x{-}1$ denotes the (truncated) predecessor PV-function. In the definition of Y we used the two projections π_1 and π_2, also as PV-functions. Since the definition of Y is a quantifier-free $\mathsf{PV}(\alpha)$-formula, the set Y exists by quantifier-free $\mathsf{PV}(\alpha)$-comprehension, and it is clear by construction that it witnesses $\neg PHP(u)$.

To complete the proof, plug $\neg C(u)$ for $PHP(u)$ in (5) and quantifier-free $\mathsf{PV}(\alpha)$-induction gives $\neg C(x)$, and hence $PHP(x)$. $\square$

Remark 11. The model $(M, \mathcal{X})$ that witnesses the above consistency is a model of V_2^0 where $PHP(n)$ fails for some nonstandard $n \in M$: otherwise $\alpha^1_{\neg PHP}$ would be true and witnessed by trivial circuits that always reject.

3 Formally verified model-checkers

We shall need to formally reason about certain straightforwardly defined exponential time machines, namely model-checkers and universal machines. A model-checker M_φ for a formula $\varphi(\bar{X}, \bar{x})$ has oracle access to $\bar{X}$ and, on input $\bar{x}$, decides whether $\varphi(\bar{X}, \bar{x})$ is true. For example, by nesting a loop for each bounded quantifier, $\Sigma_0^{1,b}$-formulas have straightforward model-checkers that run in exponential time and polynomial space. We define such model-checkers with care, so that $\mathsf{S}^1_2(\alpha)$ verifies their time and space bounds as well as their correctness. This correctness statement has to be formulated carefully because, in general, $\mathsf{S}^1_2(\alpha)$ cannot prove that a halting computation of $M_\varphi^{\bar{X}}$ on $\bar{x}$ exists. Thus, proving correctness means to show that *if* a computation exists, *then* it does what it is supposed to do. To prove this we use some constructions that are similar in spirit to those in [4].

3.1 Preliminaries: explicit machines

In short, a machine will be called *explicit* if the theory $\mathsf{S}^1_2(\alpha)$ proves that its halting computations terminate within a specified number of steps, using no more than a specified amount of space in its work tapes, and by querying its oracles no further than a specified position.

Machine model Our model of computation is the multi-tape oracle Turing machine with one-sided infinite tapes (i.e., cells indexed by $\mathbb{N}$) and an alphabet containing $\{0, 1\}$. The content of cell 0 is fixed to a fixed symbol marking the end of the tape. At the start, the heads scan cell 1. The machines can be deterministic or non-deterministic. Such a machine M has read-only input tapes, and work tapes and oracle tapes. If there are k input tapes, then its inputs are k-tuples $\bar{x} = (x_1, \dots, x_k)$ of numbers with the length-$|x_i|$ binary representation of x_i written on the i-th input tape. The length of the input is $|\bar{x}| = \max_i |x_i|$. If M does not have oracle tapes, then it is a machine *without oracles*. If M has $\ell \geqslant 1$ oracle tapes, then we write $M^{\bar{X}}$ for the machine with oracles $\bar{X} = (X_1, \dots, X_\ell)$. When the machine enters a special query state, it moves to one out of 2^ℓ many special answer states which codes the answers to the ℓ queries written on the ℓ oracle tapes, i.e., whether the number written (in binary) on the i-th oracle tape belongs to X_i or not.

A *partial space-s time-t query-q computation of $M^{\bar{X}}$ on $\bar{x}$* comprises $t + 1$ configurations, the first one being the starting configuration, every other being a successor of the previous one, and repeating halting configurations, if any. Being *space-s* means that the largest visited cell on each tape is at most s, and being *query-q* means that the largest visited cell on each oracle tape is at most most $|q|$.

Coding computations Fix a machine M. Let $s,t,q \in \mathbb{N}$ and consider a partial space-s, time-t, query-q computation of M on an unspecified input with unspecified oracles. A configuration is coded by an $(s+1)$-tuple $(q, c_0, \ldots, c_{s-1})$ of numbers: q codes the current state of the machine; c_i codes, for each tape, a *position bit* indicating whether the index of the currently scanned cell is at most i and, for each work or oracle tape, the content of cell i. We assume that these numbers are smaller than M (the machine is (coded by) a number), so we get an $(s+1) \times (t+1)$ matrix of such numbers. This matrix is coded by the set Y of numbers bounded by $\langle s, t, |M| \rangle$ that contains exactly those $\langle i, j, k \rangle$ such that $i \leqslant s$, $j \leqslant t$, $k < |M|$ and the (i,j)-entry of the matrix has k-bit 1.

The details of the encoding are irrelevant. What is required is that there is a $\mathsf{PV}(\alpha)$-function f^Y such that $f^Y(t,s,q,j)$ gives, about the j-th configuration, a number coding the state, the positions of the heads, the contents of the cells they scan, and the numbers that are written in binary in the first $|q|$ cells of the oracle tapes. In the encoding sketched above, to find the position of a specific head, f^Y uses binary search to find $i \leqslant s$ where its position bit flips; computing the oracle queries is possible because the oracle tapes contain numbers below $2^{|q|}$.

Having f^Y, it is straightforward to write a natural $\Pi^b_1(\alpha)$-formula stating

$$\text{“}Y \text{ is a partial space-}s \text{ time-}t \text{ query-}q \text{ computation of } M^{\bar{X}} \text{ on } \bar{x}\text{”}. \tag{6}$$

The free variables of this formula are $Y, \bar{X}, \bar{x}, s, t, q$. Exceptionally, we shall also consider M on the formal level, in which case M is an additional free *number variable*. All quantifiers in the $\Pi^b_1(\alpha)$-formula (6) can be $\mathsf{S}^1_2(\alpha)$-provably bounded by $p(s,t,|q|,|M|,|\bar{x}|)$ for a polynomial p, where $|\bar{x}|$ stands for $|x_1|, \ldots, |x_k|$. If M is a machine without oracles, the formula is $\mathsf{S}^1_2(\alpha)$-provably equivalent to the one with $q = 0$, and we omit ‘query-q’. We also omit ‘space-s’ if $s = t$. Further, replacing ‘partial’ by ‘halting’ or ‘accepting’ or ‘rejecting’ are obvious modifications of the formula.

Explicit machines Binary search gives a $\mathsf{PV}(\alpha)$-function $\mathit{time}^Y(s,t)$ such that, provably in S^1_2, if Y is a halting time-t space-s query-q computation of $M^{\bar{X}}$ on $\bar{x}$, then $\mathit{time}^Y(s,t)$ is the minimal $j \leqslant t$ such that the j-th configuration in Y is halting. We make the further assumption that M never writes blank (but can write a copy of this symbol), so heads leave marks on visited cells. Binary search can then compute the maximal non-blank cell in the j-th configuration on any tape. By quantifier-free induction for $\mathsf{PV}(\alpha)$-formulas, S^1_2 proves that this cell number is non-decreasing for $j = 0, 1, \ldots, t$. Hence, there is a $\mathsf{PV}(\alpha)$-function $\mathit{space}^Y(s,t)$ such that, provably in S^1_2, if Y is a halting time-t space-s query-q computation of $M^{\bar{X}}$ on $\bar{x}$, then $\mathit{space}^Y(s,t)$ is the maximal cell visited in Y on any tape. Similarly, there is a $\mathsf{PV}(\alpha)$-function $\mathit{query}^Y(s,t)$ that computes the maximal cell visited on a query tape.

Definition 12. A machine M is *explicit* if there are terms $s(\bar{x}), t(\bar{x}), q(\bar{x})$ such that

$$\mathsf{S}^1_2(\alpha) \vdash \begin{array}{l} \text{“}Y \text{ is a halting space-}s' \text{ time-}t' \text{ query-}q' \text{ computation of } M^{\bar{X}} \text{ on } \bar{x}\text{”} \to \\ \mathit{time}^Y(s',t') \leqslant t(\bar{x}) \wedge \mathit{space}^Y(s',t') \leqslant s(\bar{x}) \wedge \mathit{query}^Y(s',t') \leqslant q(\bar{x}). \end{array}$$

We say that the terms $s = s(\bar{x}), t = t(\bar{x}), q = q(\bar{x})$ *witness* that M is explicit. Further, if $r(\bar{x})$ is another term, then we say that $r = r(\bar{x})$ *witnesses* that M is an

explicit NEXP*-machine*	if it is non-deterministic	with $t = s = q = r$;
explicit EXP*-machine*	if it is deterministic	with $t = s = q = r$;
explicit PSPACE*-machine*	if it is deterministic	with $t = q = r$ and $s = \|r\|$;
explicit NP*-machine*	if it is non-deterministic	with $t = s = \|r\|$ and $q = r$;
explicit P*-machine*	if it is deterministic	with $t = s = \|r\|$ and $q = r$.

Observe that, if s, t, q witness that M is explicit, and $s' = s'(\bar{x}), t' = t'(\bar{x}), q' = q'(\bar{x})$ are terms such that $\mathsf{S}^1_2 \vdash s(\bar{x}) \leqslant s'(\bar{x}) \wedge t(\bar{x}) \leqslant t'(\bar{x}) \wedge q(\bar{x}) \leqslant q'(\bar{x})$, then also s', t', q' witness that M is explicit. E.g., if r witnesses that M is an explicit P-machine, then r also witnesses that M is an explicit PSPACE-machine.

Given an explicit machine M, we omit 'space-s time-t query-q' in (6) and its variations with 'halting', 'accepting' or 'rejecting'. E.g. for an explicit EXP-machine M, say witnessed by $r = r(\bar{x})$, we have a $\Pi^b_1(\alpha)$-formula

$$\text{“}Y \textit{ is an accepting computation of } M^{\bar{X}} \textit{ on } \bar{x}\text{”}. \tag{7}$$

This means that Y is a space-$r(\bar{x})$ time-$r(\bar{x})$ query-$r(\bar{x})$ computation of $M^{\bar{X}}$ on $\bar{x}$ that ends in an accepting halting configuration, and all queries *“$z \in X$?”* during the computation satisfy $z < 2^{|r(\bar{x})|}$. In particular,

$$Y \leqslant \langle r(\bar{x}), r(\bar{x}), |M| \rangle \tag{8}$$

provably in S^1_2. Furthermore, all quantifiers in the $\Pi^b_1(\alpha)$-formula (7) can be $\mathsf{S}^1_2(\alpha)$-provably bounded by $p(r(\bar{x}), |M|, |\bar{x}|)$ for a polynomial p, where $|\bar{x}|$ stands for $|x_1|, \ldots, |x_k|$.

Thereby, our mode of speech follows [22, Definition 8.1.2] in that the time bound is used to determine the bound on the oracle tapes.

Polynomial-time computations It is well-known that S^1_2 formalizes polynomial time computations. We shall use this in the form of the following lemma.

For an explicit P-machine M, its computations Y can be coded by numbers y and we get a $\Pi^b_1(\alpha)$-formula

$$\text{“}y \textit{ is a halting computation of } M^{\bar{X}} \textit{ on } \bar{x}\text{”}.$$

Here, y is a number sort variable, and the free variables are $\bar{X}, \bar{x}, y$. If M has a special output tape, we agree that the output of a computation is the number whose binary representation is written in cells $1, 2, \ldots$ up to the first cell not containing a bit. We have a $\mathsf{PV}(\alpha)$-function out_M such that, provably in $S^1_2(\alpha)$, if y is a halting computation of $M^{\bar{X}}$ on $\bar{x}$, then $out_M(y, j)$ is the content of cell j of the output tape in the halting configuration in case this is a bit; otherwise $out_M(y, j)$=2. In particular, $\mathsf{S}^1_2(\alpha)$ proves $out_M(y, j) \leqslant 2$,

Lemma 13. *For every* $\mathsf{PV}(\alpha)$*-function* $f^{\bar{X}}(\bar{x})$ *there are an explicit* P*-machine* M *and a* $\mathsf{PV}(\alpha)$*-function* $g^{\bar{X}}(\bar{x})$ *such that* $\mathsf{S}^1_2(\alpha)$ *proves*

$$\begin{aligned}&\big(\text{“}y \text{ is a halting computation of } M^{\bar{X}} \text{ on } \bar{x}\text{”} \leftrightarrow y{=}g^{\bar{X}}(\bar{x})\big) \wedge\\ &\big(j{<}|f^{\bar{X}}(\bar{x})| \to out_M(g^{\bar{X}}(\bar{x}), j{+}1){=}bit(f^{\bar{X}}(\bar{x}), j)\big) \wedge\\ &\big(j{\geqslant}|f^{\bar{X}}(\bar{x})| \to out_M(g^{\bar{X}}(\bar{x}), j{+}1){=}2\big).\end{aligned}$$

In the statement of the lemma, $bit(n,i)$ is a PV-function computing the i-bit of the binary representation of n, i.e., $bit(n,i) = \lfloor n/2^i \rfloor \bmod 2$ (in the standard model). In particular, we have $bit(n,i) = 0$ for $i \geqslant |n|$.

3.2 Deterministic model-checkers

For every $\Sigma^{1,b}_0$-formula $\varphi = \varphi(\bar{X}, \bar{x})$ in the language $\mathsf{PV}(\alpha)$ we define its *bounding term* $bt_\varphi(\bar{x})$ as follows:

1. $bt_\varphi = 0$ if φ is atomic,
2. $bt_\varphi = bt_\psi$ if $\varphi = \neg\psi$,
3. $bt_\varphi = bt_\psi + bt_\theta$ if $\varphi = (\psi \wedge \theta)$,
4. $bt_\varphi = bt_\psi(\bar{x}, t(\bar{x})) + t(\bar{x})$ if $\varphi = \exists y{\leqslant}t(\bar{x})\ \psi(\bar{X}, \bar{x}, y)$.

Below, given a tuple $\bar{x} = (x_1, \ldots, x_k)$, we write $|\bar{x}|$ for $(|x_1|, \ldots, |x_k|)$.

Lemma 14. *For every* $\Sigma^{1,b}_0$*-formula* $\varphi = \varphi(\bar{X}, \bar{x})$ *there are an explicit* PSPACE*-machine* $M^{\bar{X}}_\varphi$*, two terms* $r_\varphi(\bar{x})$ *and* $s_\varphi(\bar{x})$*, a* $\Sigma^{1,b}_0$*-formula* $C_\varphi(\bar{X}, \bar{x}, u)$*, and a polynomial* $p_\varphi(m, \bar{n})$*, such that*

(a) $\mathsf{S}^1_2(\alpha) \vdash$ *“Y is an accepting computation of $M^{\bar{X}}_\varphi$ on $\bar{x}$”* $\to \varphi(\bar{X}, \bar{x})$*,*

(b) $\mathsf{S}^1_2(\alpha) \vdash$ *“Y is a rejecting computation of $M^{\bar{X}}_\varphi$ on $\bar{x}$”* $\to \neg\varphi(\bar{X}, \bar{x})$*,*

(c) $\mathsf{S}^1_2(\alpha) \vdash$ *“$C_\varphi(\bar{X}, \bar{x}, \cdot)$ is a halting computation of $M^{\bar{X}}_\varphi$ on $\bar{x}$”,*

(d) $\mathsf{S}^1_2(\alpha) \vdash\ r_\varphi(\bar{x}) \leqslant p_\varphi(bt_\varphi(\bar{x}), |\bar{x}|)$ *,*

(e) $r_\varphi(\bar{x}), s_\varphi(\bar{x})$ *witness* $M^{\bar{X}}_\varphi$ *as explicit* EXP*- and* PSPACE*-machines, respectively.*

In addition, if $\varphi = \varphi(\bar{X}, \bar{x})$ *is a* $\Pi^b_1(\alpha)$*-formula, then there are a term* $t_\varphi(\bar{x})$ *and a quantifier-free* $\mathsf{PV}(\alpha)$*-formula* $C_\varphi(\bar{X}, \bar{x}, w, u)$ *such that*

(f) $\mathsf{T}^1_2(\alpha) \vdash\ \exists w{\leqslant}t_\varphi(\bar{x})$ *“$C_\varphi(\bar{X}, \bar{x}, w, \cdot)$ is a halting computation of $M^{\bar{X}}_\varphi$ on $\bar{x}$”,*

(g) $\mathsf{S}^1_2(\alpha) \vdash\ \varphi(\bar{X}, \bar{x}) \to$ *“$C_\varphi(\bar{X}, \bar{x}, t_\varphi(\bar{x}), \cdot)$ is an accepting computation of $M^{\bar{X}}_\varphi$ on $\bar{x}$”.*

Proof. A $\Sigma^{1,b}_0$-formula $\varphi = \varphi(\bar{X}, \bar{x})$ is called *good* if it satisfies (a)–(e). Observe that all $\Sigma^b_0(\alpha)$-formulas are good: they are $\mathsf{S}^1_2(\alpha)$-provably equivalent to formulas of the form $f^{\bar{X}}(\bar{x}){=}1$ for some $\mathsf{PV}(\alpha)$-function $f^{\bar{X}}(\bar{x})$, and we can choose a machine according to Lemma 13. Recall that an explicit P-machine is also an explicit PSPACE-machine and explicit EXP-machine (in this case, all three witnessed by the same term).

We leave it to the reader to check that the good formulas are closed under Boolean combinations. We are then left to show that if

$$\varphi(\bar{X},\bar{x}) \;=\; \exists y \leqslant t(\bar{x})\ \psi(\bar{X},\bar{x},y) \tag{9}$$

for a term $t(\bar{x})$ and a good formula $\psi = \psi(\bar{X},\bar{x},y)$, then φ is good. To lighten the notation, in the following we drop any reference to the set-parameters $\bar{X}$ in the formulas, and to the oracles $\bar{X}$ in machines, since they remain fixed throughout the proof.

The machine M_φ runs a loop searching for a y in $\{0,\ldots,t(\bar{x})\}$ that satisfies ψ. On input $\bar{x}$, it writes $y := 0$ on a work tape and then loops: it checks whether $y \leqslant t(\bar{x})$ and, if so, it updates $y := y+1$ and runs M_ψ on $(\bar{x},y)$; otherwise it halts. It accepts or rejects according to a *flag* bit b stored in its state space: b is initially set to 0, and it is set to 1 when and if an M_ψ-run accepts.

To prove (a)–(e) we want a quantifier-free $\mathsf{PV}(\alpha)$-formula $D(Y,\bar{x},y,u)$ that extracts the M_ψ-computation simulated in the y-loop. More precisely, we want $\mathsf{S}^1_2(\alpha)$ to prove that, if Y is a halting computation of M_φ on $\bar{x}$, then $D(Y,\bar{x},y,\cdot)$ is a halting computation of M_ψ on $(\bar{x},y)$. For this, we design the details of M_φ in a way so that the j-th step of the computation of M_ψ on $(\bar{x},y)$ is simulated by M_φ at a time easily computed from $\bar{x},y,j$.

Description of M_φ. Set $r(\bar{x}) = r_\psi(\bar{x},t(\bar{x}))$ where $r_\psi(\bar{x},y)$ is the term claimed to exist for ψ. Note that $\mathsf{S}^1_2(\alpha)$ proves that $r_\psi(\bar{x},y) \leqslant r(\bar{x})$ for $y \leqslant t(\bar{x})$. Additionally to properties (a)–(e) for ψ, we assume inductively that $\mathsf{S}^1_2(\alpha)$ proves that the halting configuration of M_ψ on $(\bar{x},y)$ equals the initial configuration except for the state, that is, M_ψ cleans all worktapes and moves all heads back to cell 1 before it halts.

Our machine initially computes $t = t(\bar{x})$ and $r = r(\bar{x})$ and two binary *clocks* initially set to $0^{|t|}$ and $0^{|r|}$. The terms are evaluated using explicit P-machines according to Lemma 13. The initial settings of the clocks are simply computed by scanning the binary representations of t and r that were computed at the start. This initial computation of terms, and initialization of clocks, takes time exactly $\mathit{ini}(\bar{x})$ for some PV-function $\mathit{ini}(\bar{x})$. Further, $\mathsf{S}^1_2(\alpha)$ proves $\mathit{ini}(\bar{x}) \leqslant |t_i(\bar{x})|$ for a suitable term $t_i(\bar{x})$.

The y-loop is implemented as follows. First update y, the value of the first clock. To do this, sweep over the first clock, and then back, in exactly $(2|t|+2)$ steps, doing the following: copy y without leading 0's to some tape, so this tape holds the length-$|y|$ binary representation of y (as expected by M_ψ); increase the clock by 1 if $y < t$, and reset it to $0^{|t|}$ if $y = t$; in the latter case store a bit signaling this; this signal bit halts the computation (in the next y-loop) instead of doing the y-update. After this y-update, simulate r steps of M_ψ on $(\bar{x},y)$ by an inner loop: in $2|r|+2$ steps sweep twice over the second clock. If its value was smaller that r, then increase it by 1 and simulate the next step of M_ψ's computation; this can mean repeating the halting computation. If its value was not smaller than r, then set the clock back to $0^{|r|}$. Thus, exactly $2|r|+3$ steps are spent for one step of M_ψ and one y-loop takes exactly $t_\ell(\bar{x}) := (r(\bar{x})+1)\cdot(2|r(\bar{x})|+3)$ steps.

If the signal bit halts the computation, then our machine first cleans all tapes and moves heads back to cell 1, before halting. We omit a description of this final polynomial time computation. It can be implemented to take exactly $\mathit{fin}(\bar{x})$ steps for a PV-function $\mathit{fin}(\bar{x})$, and S^1_2 proves $\mathit{fin}(\bar{x}) \leqslant |t_f(\bar{x})|$ for a suitable term $t_f(\bar{x})$.

Thus M_φ runs in time exactly $\mathit{ini}(\bar{x}) + (t(\bar{x})+1) \cdot t_\ell(\bar{x}) + \mathit{fin}(\bar{x})$. It simulates r steps of M_ψ on $(\bar{x}, y)$ at times

$$t(\bar{x}, y, j) \coloneqq \mathit{ini}(\bar{x}) + y \cdot t_\ell(\bar{x}) + (j+1) \cdot (2|r(\bar{x})| + 3) \tag{10}$$

for $j < r(\bar{x})$.

Explicitness. Let $s_\psi(\bar{x}, y)$ be the term that witnesses M_ψ as an explicit PSPACE-machine. Let Y be a halting computation of M_φ on $\bar{x}$. There is a $\mathsf{PV}(\alpha)$-function that from $\bar{x}$ computes (a number coding) the initial computation of terms and clocks, and $\mathsf{S}^1_2(\alpha)$ proves its halting configuration is as described. Clearly, $\mathsf{S}^1_2(\alpha)$ proves that the first $\mathit{ini}(\bar{x})$ steps of Y coincide with this computation. In particular, $\mathsf{S}^1_2(\alpha)$ proves that the clocks computed in Y have the desired length. Similarly, there is a $\mathsf{PV}(\alpha)$-function that from $\bar{x}, y, j$ computes (a number coding) the space-$|s_\psi(\bar{x}, y)|$ configuration of M_ψ at time $t(\bar{x}, y, j)$ in Y.

We prove, by quantifier-free induction, that the computation Y simulates the steps of M_ψ at times $t(y, j) \coloneqq t(\bar{x}, y, j)$ for $y \leqslant t$ and $j < r$. Assume this holds for time $t(y, j)$. We verify it for time $t(y, j+1)$ or time $t(y+1, 0)$ depending on whether $j < r$ or $j = r$. Assume the former; the latter case is similar. Compute the time-$(2|r|+3)$ computation (that sweeps twice over the clock and simulates one more step of M_ψ) starting at the configuration at time $t(y, j)$; then Y must coincide with this computation between time $t(y, j)$ and time $t(y, j+1)$. Hence, Y simulates a step of M_ψ at time $t(y, j+1)$. Similarly, quantifier-free induction proves that the M_ψ-configurations at the times $t(y, j)$ in Y are successors of each others. This yields a quantifier-free $\mathsf{PV}(\alpha)$-formula $D(Y, \bar{x}, y, u)$ as desired.

From the configuration at time $\mathit{ini}(\bar{x}) + (t+1) \cdot t_\ell(\bar{x})$ one can compute the final $\mathit{fin}(\bar{x})$ steps of the clean-up computation before M_φ halts, and the last $\mathit{fin}(\bar{x})$ steps of Y must coincide with that. Hence, $\mathsf{S}^1_2(\alpha)$ proves that the configuration of Y at time $\mathit{ini}(\bar{x}) + (t+1) \cdot t_\ell + \mathit{fin}(\bar{x})$ is halting. Recalling that $\mathit{ini}(\bar{x}) \leqslant |t_i(\bar{x})|$ and $\mathit{fin}(\bar{x}) \leqslant |t_f(\bar{x})|$, this implies that the terms

$$r_0(\bar{x}) \coloneqq |t_i(\bar{x})| + (t(\bar{x}) + 1) \cdot t_\ell(\bar{x}) + |t_f(\bar{x})|, \tag{11}$$

$$s_0(\bar{x}) \coloneqq |t_i(\bar{x})| + (|t(\bar{x})| + 1) + (|r(\bar{x})| + 1) + |s_\psi(\bar{x}, t(\bar{x}))| + |t_f(\bar{x})| \tag{12}$$

witness that M_φ is explicit with respect to time and space, respectively. As query-bound term $q_0(\bar{x})$ we can take $2^{s_0(\bar{x})}$, which exists because, provably in $\mathsf{S}^1_2(\alpha)$, the term $s_0(\bar{x})$ is small, i.e., bounded by $|s'(\bar{x})|$ for some other term $s'(\bar{x})$. Since S^1_2 proves $t_\ell(\bar{x}) \leqslant 16 r(\bar{x})^2$ and hence also $r_0(\bar{x}) \leqslant q_0(\bar{x})$, the term $s_\varphi(\bar{x}) \coloneqq q_0(\bar{x})$ itself witnesses M_φ as explicit PSPACE-machine. Similarly, $r_\varphi(\bar{x}) \coloneqq r_0(\bar{x})$ witnesses M_φ as an explicit EXP machine. This proves (e). Note that the term r_φ does not serve also as witness that M_φ is an explicit PSPACE-machine because $r_\varphi(\bar{x})$ is actually smaller than $s_\varphi(\bar{x})$; we need the tighter time-bound in (11) to be able to prove (d) later.

Proof of (a)–(e). For (a) argue in $\mathsf{S}^1_2(\alpha)$ and suppose Y is an accepting computation of M_φ on $\bar{x}$. Being accepting means that the final state has flag $b = 1$, while the starting state has flag $b = 0$. By binary search we find a time when b flips from 0 to 1. This time determines $y_0 \leqslant t$ such that the y_0 loop accepts. Then $Z := D(Y, \bar{x}, y_0, \cdot)$ is an accepting computation of M_ψ on $(\bar{x}, y_0)$. Note that Z exists by $\Delta_1^b(\alpha)$-comprehension. Then (a) for ψ, implies $\psi(\bar{x}, y_0)$ and thus $\varphi(\bar{x})$.

For (b), argue in $\mathsf{S}_2^1(\alpha)$ and suppose Y is a rejecting computation of M_φ on $\bar{x}$, so the flag is 0 in the final configuration. Let $y \leqslant t$. Then $D(Y, \bar{x}, y, \cdot)$ is a rejecting computation of M_ψ on $(\bar{x}, y)$: otherwise the y loop sets the flag to 1 and then binary search finds a time where the flag flips from 1 to 0 in Y which contradicts the working of M_φ. Then (b) for ψ implies $\neg\psi(\bar{x}, y)$. As y was arbitrary, we get $\neg\varphi(\bar{x})$.

For (c), it is easy to construct from C_ψ a formula $C_{\psi,0}$ such that $\mathsf{S}_2^1(\alpha)$ proves that the set $C_{\psi,0}(\bar{x}, y, \cdot)$ is the computation of the y-loop of M_φ on $\bar{x}$ with flag 0 stored in the state space. There is an analogous formula $C_{\psi,1}$ for flag 1. These formulas just stretch the computation described by C_ψ and interleave it with the trivial updates of the clocks. The desired formula $C_\varphi(\bar{x}, u)$ 'glues together' these computations, plus the initial $\mathit{ini}(\bar{x})$ steps of initialization, and the final $\mathit{fin}(\bar{x})$ steps of clean-up. We sketch the definition of $C_\varphi(\bar{x}, u)$: from u we can compute y such that the truth value of $C_\varphi(\bar{x}, u)$ is one of the bits in the code of the computation of the y-loop of M_φ on $\bar{x}$, or one of the bits in the code of the initial or final computation. Then $C_\varphi(\bar{x}, u)$ states

$$(\exists z{<}y\ \psi(\bar{x}, z) \wedge C_{\psi,1}(\bar{x}, y, u)) \vee (\neg\exists z{<}y\ \psi(\bar{x}, z) \wedge C_{\psi,0}(\bar{x}, y, u)). \tag{13}$$

For (d) and (e), recall the choice of $r_\varphi(\bar{x}) := r_0(\bar{x})$ with $r_0(\bar{x})$ in (11). Earlier we argued that $r_\varphi(\bar{x})$ witnesses M_φ as an explicit EXP-machine, which proves (e). For (d), recall that $t_\ell(\bar{x}) = (r(\bar{x}) + 1) \cdot (2|r(\bar{x})| + 3)$ and hence $r_\varphi(\bar{x}) = p(r(\bar{x}), t(\bar{x}), |\bar{x}|)$ for a suitable polynomial p, provably in S_2^1. Recalling that $r(\bar{x}) = r_\psi(\bar{x}, t(\bar{x}))$, and that by (d) for ψ we have $r_\psi(\bar{x}, y) \leqslant p_\psi(bt_\psi(\bar{x}, y), |\bar{x}|, |y|)$ provably in S_2^1, from $bt_\varphi(\bar{x}) = bt_\psi(\bar{x}, t(\bar{x})) + t(\bar{x})$ we get, also provably in S_2^1, that $r_\varphi(\bar{x}) \leqslant p_\varphi(bt_\varphi(\bar{x}), |\bar{x}|)$ for a suitable polynomial p_φ.

Proof of (f)–(g). Assume φ is a $\Pi_1^b(\alpha)$-formula. We modify the given construction as follows. Up to $\mathsf{S}_2^1(\alpha)$-provable equivalence we have

$$\varphi(\bar{X}, \bar{x}) = \forall y{\leqslant}t(\bar{x})\ g^{\bar{X}}(\bar{x}, y){=}1$$

where $t(\bar{x})$ is a term and $g^{\bar{X}}(\bar{x}, y)$ is a $\mathsf{PV}(\alpha)$-function. As before, we drop any reference to the set-parameters $\bar{X}$, and to the oracles $\bar{X}$, since they will stay fixed throughout the proof. We define M_φ similarly as before with the role of M_ψ played by a P-machine checking $g(\bar{x}, y){=}1$ according to Lemma 13. The only difference is in the flag bit: it is initially set to 1, and it is set to 0 when and if a y-loop rejects (meaning $\neg g(\bar{x}, y){=}1$).

In this case we can choose r small, i.e., equal to $|r'|$ for some term $r' = r'(\bar{x})$, so there is a $\mathsf{PV}(\alpha)$-function $h(\bar{x}, y)$ that computes (a number that codes) the computation of the y-loop of M_φ. Then $C_\varphi(\bar{x}, w, u)$ 'glues together' these computations plus suitable initial and final computations. The only problem is to determine the flag b stored in the states of M_φ. For this we need to know the minimal $w \leqslant t$ such that $\neg g(\bar{x}, w){=}1$ holds, or take $w = t + 1$ if $\varphi(\bar{x})$ holds. Such w exists provably in $\mathsf{T}_2^1(\alpha)$. This shows (f) for $t_\varphi(\bar{x}) := t(\bar{x}) + 1$. For (g), assuming $\varphi(\bar{x})$ we can take $w = t+1$ directly since in this case the flag bit is always 1 provably in $\mathsf{S}^1_2(\alpha)$. $\square$

Remark 15. The proof shows that the quantifier complexity of C_φ is close to that of φ. If $\varphi \in \Sigma^b_0(\alpha)$, then C_φ is a quantifier free $\mathsf{PV}(\alpha)$-formula. If $\varphi \in \Sigma^b_i(\alpha)$ for $i > 0$, then C_φ is a Boolean combination of $\Sigma^b_i(\alpha)$-formulas. Note that if the outer quantifier in (9) is sharply bounded, i.e., $t(\bar{x}) = |t'(\bar{x})|$ for some term $t'(\bar{x})$, then the y-bounded quantifiers in (13) are sharply bounded too.

3.3 Optimality remarks

This subsection offers some remarks stating that Lemma 14.f cannot be improved in certain respects. This material is not needed in the following.

M: new subsection – could be deleted

Remark 16. For our definition of $M^{\bar{X}}_\varphi$, one cannot replace $\mathsf{T}^1_2(\alpha)$ by $\mathsf{S}^1_2(\alpha)$ in Lemma 14.f unless $\mathsf{S}^1_2 = \mathsf{T}^1_2$.

Proof. Let $\varphi(x) = \exists y{\leqslant}x\ \psi(y,x)$ for ψ a quantifier-free PV-formula, and assume (f) holds for $\mathsf{S}^1_2(\alpha)$ instead $\mathsf{T}^1_2(\alpha)$. We show $\mathsf{S}^1_2(\alpha)$ proves that, if there is $y \leqslant x$ such that $\psi(y,x)$, then there is a minimal such y. Argue in $\mathsf{S}^1_2(\alpha)$ and suppose $\varphi(x)$. By $\Delta^b_1(\alpha)$-comprehension and (f) there is a halting computation Y of M_φ on x. By (b) it cannot be rejecting, so is accepting. Our proof of (a) gives $\psi(y_0,x)$ for $y_0 \leqslant x$ such that the flag b flips from 0 to 1 in loop y_0. We claim y_0 is minimal. This is clear if $y_0 = 0$. Otherwise we had $b = 0$ after the loop on $y_0 - 1$ (in Y). For contradiction, assume there is $y_1 < y_0$ with $\psi(y_1,x)$. Then the loop on y_1 would set $b = 1$. By quantifier-free induction we find a time between y_1 and $y_0 - 1$ where b flips from 1 to 0. This contradicts the working of M_φ. $\square$

Fix *any* machines M_φ satisfying the lemma. Call a formula *true* if its universal closure is true in the standard model.

Remark 17. In Lemma 14.f the auxiliary $\exists w$ cannot be omitted. There is a $\Sigma^b_1(\alpha)$-formula $\varphi(X,x)$ such that for all quantifier-free $\mathsf{PV}(\alpha)$-formulas $C(X,x,u)$ the following is not true:

$$\text{“}C(X,x,\cdot)\ \textit{is a halting computation of}\ M^X_\varphi\ \textit{on}\ x\text{”}.$$

Proof. Otherwise for every $\Sigma^b_1(\alpha)$-formula $\varphi(X,x)$ is equivalent to a quantifier-free $\mathsf{PV}(\alpha)$-formula $D(X,x)$. Let $A \subseteq \mathbb{N}$ be such that $\mathsf{NP}^A \notin \mathsf{P}^A$ and choose $Q \in \mathsf{NP}^A \smallsetminus \mathsf{P}^A$. Choose a $\Sigma^b_1(\alpha)$-formula $\varphi(X,x)$ defining Q in $(\mathbb{N},A)$, the model where X is interpreted by A. Note $D(X,x)$ defines in $(\mathbb{N},A)$ a problem in P^A. Then $(\varphi(X,x) \leftrightarrow D(X,x))$ fails in $(\mathbb{N},A)$ for some x, and hence also in $(\mathbb{N},A')$ for some bounded $A' \subseteq A$ (Remark 7). Thus, this equivalence is not true. $\square$

Remark 18. Lemma 14.f does not extend to much more complex formulas. There is a $\Pi^b_2(\alpha)$-formula $\varphi(X,x)$ such that for all terms t and all quantifier-free $\mathsf{PV}(\alpha)$-formulas C the following is not true:

$$\exists w{\leqslant}t(x)\ \text{“}C(X,x,w,\cdot)\ \textit{is a halting computation of}\ M^X_\varphi\ \textit{on}\ x\text{”}.$$

Proof. Note this is a $\Sigma_2^b(\alpha)$-formula, so for every $A \subseteq \mathbb{N}$ defines in $(\mathbb{N}, A)$ a problem in $(\Sigma_2^{\mathsf{P}})^A$. Choose A such that $(\Pi_2^{\mathsf{P}})^A \neq (\Sigma_2^{\mathsf{P}})^A$ and argue similarly as before. □

3.4 Non-deterministic model-checkers

We shall also need model-checkers for $\hat{\Sigma}_1^{1,b}$-formulas. As a first step we prove a technical lemma showing how to convert an explicit oracle PSPACE-machine M^Y into an explicit NEXP-machine N that first guesses the oracle Y on a *guess tape*, and then simulates M^Y. As usual, we need to show that $\mathsf{S}_2^1(\alpha)$ is able to prove that this construction does what is claimed.

Lemma 19. *For every explicit* PSPACE*-machine $M^{Y,\bar{X}}$ that, as explicit* EXP*-machine, is witnessed by term $r_M(\bar{x})$, there are an explicit* NEXP*-machine $N^{\bar{X}}$, a term $r_N(\bar{x})$, a polynomial $p_N(m,\bar{n})$, and quantifier-free* PV(α)*-formulas F, G, H such that*

(a) $\mathsf{S}_2^1(\alpha) \vdash$ *"Z is an accepting computation of $M^{Y,\bar{X}}$ on $\bar{x}$" →*
"$F(Z, Y, \bar{X}, \bar{x}, \cdot)$ is an accepting computation of $N^{\bar{X}}$ on $\bar{x}$".

(b) $\mathsf{S}_2^1(\alpha) \vdash$ *"Z is an accepting computation of $N^{\bar{X}}$ on $\bar{x}$" →*
"$G(Z, \bar{X}, \bar{x}, \cdot)$ is an accepting computation of $M^{H(Z,\bar{X},\bar{x},\cdot),\bar{X}}$ on $\bar{x}$"

(c) $\mathsf{S}_2^1(\alpha) \vdash r_N(\bar{x}) \leqslant p_N(r_M(\bar{x}), |\bar{x}|)$,

(d) the term $r_N(\bar{x})$ witnesses $N^{\bar{X}}$ as explicit NEXP*-machine.*

Proof. Set $r = r_M(\bar{x})$. By assumption, the triple of terms $r_M(\bar{x}), r_M(\bar{x}), r_M(\bar{x})$ witnesses that $M^{Y,\bar{X}}$ is explicit. In particular, every query *"$z \in Y$?"* made by $M^{Y,\bar{X}}$ on $\bar{x}$ satisfies $|z| \leqslant |r|$ and hence $z < 2^{|r|}$. The machine $N^{\bar{X}}$ on $\bar{x}$ guesses a binary string Y of length $2^{|r|}$ on a *guess tape* and then simulates $M^{Y,\bar{X}}$ on $\bar{x}$ as follows: an oracle query *"$z \in Y$?"* of $M^{Y,\bar{X}}$ is answered reading cell $z+1$ on the guess tape. As in the proof of Lemma 14, to prove (a)–(d) we need to design the details of N in a way so that the j-th step of the computation of M is simulated by N at a time easily computed from $\bar{x}, j$. To reduce notation, in the following we drop any reference to the oracles $\bar{X}$ as they will remain fixed throughout the proof.

Description of N. The machine N on $\bar{x}$ first computes r and two binary clocks initialized to $0^{|r|+1}$ and $0^{|r|}$, respectively. To write Y of length $2^{|r|}$ on the guess tape the machine checks whether the first clock equals $2^{|r|}$ and, if not, increases it by one and moves one cell to the right on the guess tape. This is done in exactly $2|r|+5$ steps. Once the clock equals $2^{|r|}$, the machine moves back to cell 1 on the guess tape and non-deterministically writes 0 or 1 in each step, except in the step that rebounds on cell 0 to end in cell 1. The terms are computed with explicit P-machines according to Lemma 13. The initial computation of terms, and initialization of clocks, takes time exactly $ini(\bar{x})$ for some PV-function $ini(\bar{x})$. Therefore, the guess of Y takes exactly $guess(\bar{x}) := ini(\bar{x}) + 2^{|r|} \cdot (2|r|+5) + 2^{|r|} + 1$ steps. Moreover, S_2^1 proves $guess(\bar{x}) \leqslant t_g(\bar{x})$, where

$$t_g(\bar{x}) := |t_i(\bar{x})| + 2^{|r_M(\bar{x})|} \cdot (2|r_M(\bar{x})| + 5) + 2^{|r_M(\bar{x})|} + 1,$$

for a suitable term $t_i(\bar{x})$ such that S^1_2 proves $ini(\bar{x}) \leqslant |t_i(\bar{x})|$.

The machine simulates r steps of M^Y using the second clock. Comparing this clock with r and updating it takes $2|r|+2$ steps. If the value of the clock is less than r, then a step of M^Y is simulated by reading the (z+1)-cell of the guess tape where z is the content of M^Y's oracle tape for Y. This is done as follows. The machine moves forward over the guess tape, and rewinds back to cell 1. With each step forward it increases the first clock by one and checks whether it equals z or $2^{|r|}$. If and when the clock equals z, it stores the *oracle bit* read on the guess tape in its state space. Otherwise, i.e., $z \geqslant 2^{|r|}$, the machine stores oracle bit 0. When the clock equals $2^{|r|}$, the scan of the guess tape ends, and the rewinding to cell 1 starts (in the next step). Doing this takes time exactly $2^{|r|} \cdot (2|r|+4) + 2^{|r|} + 1$ and the oracle bit is stored at time $\min\{z, 2^{|r|}\} \cdot (2|r|+4)$. Thus, when the value of the second clock is less than r, one step of M^Y is simulated in exactly

$$t_s(\bar{x}) := (2|r_M(\bar{x})|+2) + 2^{|r_M(\bar{x})|} \cdot (2|r_M(\bar{x})|+4) + 2^{|r_M(\bar{x})|} + 2$$

steps. Otherwise, the simulation halts in an accepting or rejecting state according to M^Y's state. In total, the machine runs for exactly $guess(\bar{x}) + r \cdot t_s(\bar{x}) + (2|r|+2)$ steps. The steps of M^Y on $\bar{x}$ are simulated at times

$$t(\bar{x}, j) := guess(\bar{x}) + (j+1) \cdot t_s(\bar{x})$$

for $j < r_M(x)$. The runtime is bounded by the term

$$r_N(\bar{x}) := t_g(\bar{x}) + r_M(\bar{x}) \cdot t_s(\bar{x}) + (2|r_M(\bar{x})|+2)$$

Explicitness. We argue that this bound on the runtime of N can be verified in $\mathsf{S}^1_2(\alpha)$, given a halting computation Z of N on $\bar{x}$. Note that, unlike the simulation in Lemma 14, a single step is simulated in possibly exponential time $t_s(\bar{x})$. However, this possibly exponential time computation is simply described: Since M^Y is an explicit PSPACE-machine, its configurations can be coded by numbers. Now, given a number coding the configuration of M^Y within Z at time $t(j) := t(\bar{x}, j)$, say with Y-oracle query z, and given a time $i < t_s(\bar{x})$, we can compute the configuration of the clocks and the state of the (to-be-)stored oracle-bit at time $t(j)+i$. Now, quantifier-free induction suffices to prove that the oracle bit is stored at the desired time and equals the content of the (z+1)-cell of the guess tape (or 0 if $z \geqslant 2^{|r|}$). Quantifier-free induction proves that the configurations of M^Y within Z at times $t(j)$ for $j < r$ are successors of those preceding them. In particular, $\mathsf{S}^1_2(\alpha)$ proves that the configuration at time $r_N(\bar{x})$ is halting. Space and query bounds can be similarly verified, so N is explicit and witnessed by $r_N(\bar{x})$.

Proof of (a)–(d). For (a), the quantifier-free formula F concatenates an initial polynomial-time computation of the terms and clocks, a guess of Y, and a simulation of Z. Each configuration of the guess of Y is computable in polynomial time. The simulation of Z stretches each step of M^Y to a time $t_s(\bar{x})$ computation, each configuration of which is easily computed from Y and Z in polynomial time. Quantifier-free induction proves that a Y-query z in Z is answered according to the bit in the (z+1)-cell on the guess tape.

For (b), the quantifier-free formula H extracts the guess Y from Z and the quantifier-free formula G extracts the simulated computation at the times $t(\bar{x}, j)$ for $j < r_M(\bar{x})$.

For (c) and (d), we already argued that the term $r_N(\bar{x})$ witnesses N as an explicit NEXP-machine. The claim that $r_N(\bar{x}) \leqslant p_N(r_M(\bar{x}), |\bar{x}|)$ holds for a suitable polynomial p_N follows by inspection, and $\mathsf{S}^1_2(\alpha)$ proves it. $\square$

Now we can state the lemma that proves that every $\hat{\Sigma}^{1,b}_1$-formula has a formally verified model-checker. In its statement, the bounding term $bt_\psi(\bar{x})$ of a $\hat{\Sigma}^{1,b}_1$-formula $\psi = \psi(\bar{X}, \bar{x})$ as in Equation (4) is defined to be the bounding term $bt_\varphi(\bar{x})$ of its maximal $\Sigma^{1,b}_0$ subformula $\varphi = \varphi(Y, \bar{X}, \bar{x})$.

Lemma 20. *For every $\hat{\Sigma}^{1,b}_1$-formula $\psi = \psi(\bar{X}, \bar{x})$, there exists an explicit NEXP-machine $N^{\bar{X}}_\psi$, a term $r_\psi(\bar{x})$, and a polynomial $p_\psi(m, \bar{n})$, such that*

- *(a)* $\mathsf{V}^0_2 \vdash\ \psi(\bar{X}, \bar{x}) \rightarrow \exists_2 Y$ *"Y is an accepting computation of $N^{\bar{X}}_\psi$ on $\bar{x}$".*
- *(b)* $\mathsf{S}^1_2(\alpha) \vdash\ \neg\psi(\bar{X}, \bar{x}) \rightarrow \neg\exists_2 Y$ *"Y is an accepting computation of $N^{\bar{X}}_\psi$ on $\bar{x}$".*
- *(c)* $\mathsf{S}^1_2(\alpha) \vdash\ r_\psi(\bar{x}) \leqslant p_\psi(bt_\psi(\bar{x}), |\bar{x}|)$*,*
- *(d) the term $r_\psi(\bar{x})$ witnesses $N^{\bar{X}}_\psi$ as explicit* NEXP*-machine.*

Furthermore, if the maximal $\Sigma^{1,b}_0$-subformula of ψ is a $\Pi^b_1(\alpha)$-formula, then

- *(e)* $\mathsf{S}^1_2(\alpha) \vdash \psi(\bar{X}, \bar{x}) \leftrightarrow \exists_2 Y$ *"Y is an accepting computation of $N^{\bar{X}}_\psi$ on $\bar{x}$".*

Proof. Let $\psi(\bar{X}, \bar{x}) = \exists_2 Y\ \varphi(Y, \bar{X}, \bar{x})$ where $\varphi = \varphi(Y, \bar{X}, \bar{x})$ is a $\Sigma^{1,b}_0$-formula. Recall that the bounding term of ψ is $bt_\psi(\bar{x}) = bt_\varphi(\bar{x})$. In what follows, to lighten the notation, we drop any reference to the set parameters $\bar{X}$ in formulas, and to the oracles $\bar{X}$ in machines, since they remain fixed throughout the proof.

Let M^Y_φ be the explicit PSPACE-machine given by Lemma 14 applied to φ. Let r_φ and p_φ be the term and the polynomial also given by that lemma. By Lemma 14.e, the term r_φ witnesses M^Y_φ as explicit EXP-machine. Therefore, Lemma 19 applies to M^Y_φ and r_φ and we get an explicit NEXP-machine N_ψ, a term r_ψ, and a polynomial p_ψ. We prove (a)–(e) using the quantifier-free $\mathsf{PV}(\alpha)$-formulas F, G, H also given by Lemma 19, and the $\Sigma^{1,b}_0$-formula C_φ given by Lemma 14.

For (a), argue in V^0_2 and assume $\psi(\bar{x})$ holds. Choose Y such that $\varphi(Y, \bar{x})$ holds. By Lemma 14.c, the set $Z := C_\varphi(Y, \bar{x}, \cdot)$ is a halting computation of M^Y_φ on $\bar{x}$. Note that Z exists by $\Sigma^{1,b}_0$-comprehension, which defines the theory V^0_2. By Lemma 14.b, the computation Z cannot be rejecting, so it is accepting. By Lemma 19.a, the set $F := F(Z, Y, \bar{x}, \cdot)$ is an accepting computation of N_ψ on $\bar{x}$. Note that F exists by $\Delta^b_1(\alpha)$-comprehension.

For (b), argue in $\mathsf{S}^1_2(\alpha)$ and assume Y is an accepting computation of N_ψ on $\bar{x}$. By Lemma 19.b we have that $G(Y, \bar{x}, \cdot)$ is an accepting computation of M^Z_φ on $\bar{x}$, for $Z := H(Y, \bar{x}, \cdot)$. Note that Z exists by $\Delta^b_1(\alpha)$-comprehension. By Lemma 14.a we get that $\varphi(Z, \bar{x}, \cdot)$ holds. Thus $\psi(\bar{x})$ follows.

For (c) and (d), refer to Lemma 19.c, the choices of r_ψ and p_ψ, and the fact that $bt_\psi(\bar{x}) = bt_\varphi(\bar{x})$. This also gives the claim that $r_\psi(\bar{x})$ witnesses N_ψ as explicit NEXP-machine.

For (e), argue in $\mathsf{S}^1_2(\alpha)$. If $\neg\psi(\bar{x})$ holds, use (b). If $\psi(\bar{x})$ holds, choose Y such that $\varphi(Y, \bar{x})$ holds. Then Lemma 14.g and $\Delta^b_1(\alpha)$-comprehension imply that there exists an accepting computation Z of M^Y_φ on $\bar{x}$. Now argue as in (a). $\square$

4 Consistency for NEXP

In this section we use the results of Section 3 to define two formalizations of $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$. The so-called *A-formalization* will be a variant of the one based on the α-formulas of Section 2. This new variant will be based on NEXP-machines instead of $\hat{\Sigma}^{1,b}_1$-formulas. Then we suggest an even better formalization, the *B-formalization*, that will be based on the Easy Witness Lemma (EWL) for NEXP-machines, and an associated collection of β-formulas defined here. The B-formalization is better because its negation is expressed by a $\Pi^{1,b}_1$-formula, i.e., a two-sorted formula without existential set quantifiers. In contrast, the corresponding formula for the A-formalization is the conjunction of a $\Pi^{1,b}_1$-formula and a $\Sigma^{1,b}_1$-formula. Finally, we will prove that the consistency of both formalizations with the theory V^0_2 follows from Proposition 1 and our work on formally-verified model-checkers.

4.1 A universal machine

A canonical NEXP-complete problem called Q_0 is:

> *Given $\langle N, x, t\rangle$ as input, where N is a (number coding a) non-deterministic machine, and x and t are numbers written in binary, does N accept x in at most t steps?*

A non-deterministic exponential-time machine M_0 for Q_0, on input $\langle N, x, t\rangle$, guesses and verifies a time-t computation of N on x. We ask for an implementation of this so that a weak theory can verify its correctness. This is a quite direct consequence of Lemmas 14 and 20.

Lemma 21. *There exists an explicit NEXP-machine M_0 with one input-tape and without oracles, such that for every explicit NEXP-machine M with one input-tape and without oracles, say witnessed by the term $t_M(x)$, there are quantifier-free $\mathsf{PV}(\alpha)$-formulas $F(Z, x, u)$ and $G(Z, x, u)$ such that*

(a) $\mathsf{S}^1_2(\alpha) \vdash$ *"Z is an accepting computation of M on x" $\rightarrow$ "$F(Z, x, \cdot)$ is an accepting computation of M_0 on $\langle M, x, t_M(x)\rangle$",*

(b) $\mathsf{S}^1_2(\alpha) \vdash$ *"Z is an accepting computation of M_0 on $\langle M, x, t_M(x)\rangle$" $\rightarrow$ "$G(Z, x, \cdot)$ is an accepting computation of M on x".*

In particular,

(c) $\mathsf{S}^1_2(\alpha) \vdash$ $\exists_2 Z$ *"Z is an accepting computation of M_0 on $\langle M, x, t_M(x)\rangle$"* $\leftrightarrow$
$\exists_2 Z$ *"Z is an accepting computation of M on x".*

Proof. Let π_1, π_2, π_3 be PV-functions that extract x_1, x_2, x_3 from $z = \langle x_1, x_2, x_3\rangle$. Define Π^b_1-formulas as follows:

$$\begin{aligned}\varphi_1(Z,z) &:= \varphi_2(Z, \pi_1(z), \pi_2(z), \pi_3(z)),\\ \varphi_2(Z,N,x,t) &:= \text{“}Z \textit{ is an accepting time-}t \textit{ computation of } N \textit{ on } x\text{”}.\end{aligned}$$

Let M_1^Z be the machine given by Lemma 14 applied to $\varphi_1 = \varphi_1(Z,z)$, and let $r_1(z)$ be the corresponding term. Since φ_1 is a $\Pi^b_1(\alpha)$-formula, let $t_1(z)$ and $C_1(Z,z,w,u)$ be the term and the quantifier-free $\mathsf{PV}(\alpha)$-formula given by Lemma 14.g. We set M_0 to the explicit NEXP-machine given by Lemma 19 applied to M_1^Z with term $r_1(z)$ witnessing it as EXP-machine by Lemma 14.e. In the proof of (a)–(b) we use the quantifier-free $\mathsf{PV}(\alpha)$-formulas F_1, G_1, H_1 given by Lemma 19 on M_1^Z.

For (a) we set $F(Z,x,u) := F_1(C,Z,z,u)$ where C abbreviates $C_1(Z,z,t_1(z),\cdot)$ and in both cases z abbreviates $\langle M, x, t_M(x)\rangle$. Argue in $\mathsf{S}^1_2(\alpha)$ and assume Z is an accepting computation of M on x. Since M is explicit and $t_M(x)$ is a term witnessing it, we have that Z is an accepting time-t computation of M on x, for $t := t_M(x)$. It follows that $\varphi_2(Z,M,x,t_M(x))$ holds, and hence $\varphi_1(Z,z)$ holds. Since φ_1 is a $\Pi^b_1(\alpha)$-formula, by Lemma 14.g we have that the set $C := C_1(Z,z,t_1(z),\cdot)$ is an accepting computation of M_1^Z on z. Such a C exists by $\Delta^b_1(\alpha)$-comprehension because C_1 is a quantifier-free $\mathsf{PV}(\alpha)$-formula. By Lemma 19.a we get that the set $F := F(Z,x,\cdot) = F_1(C,Z,z,\cdot)$ is an accepting computation of M_0 on z; i.e., the right-hand side of the implication in (a) holds. Again, F exists by $\Delta^b_1(\alpha)$-comprehension.

For (b) we set $G(Z,x,u) := G_1(Z,z,u)$ where, again, z abbreviates $\langle M, x, t_M(x)\rangle$. Argue in $\mathsf{S}^1_2(\alpha)$ and assume Z is an accepting computation of M_0 on z. Then, by Lemma 19.b we have that the set $G := G(Z,x,\cdot) = G_1(Z,z,\cdot)$ is an accepting computation of M_1^H on z for $H := H_1(Z,z,\cdot)$. The two sets G and H exist by Δ^b_1-comprehension. Now, Lemma 14.a implies that $\varphi_1(H,z)$ holds; i.e., H is an accepting time-t computation of M on x, for $t := t_M(x)$, and hence also an accepting computation of M on x. This shows that the right-hand side in the implication in (b) holds.

The final statement follows from (a) and (b) by $\Delta^b_1(\alpha)$-comprehension. $\square$

4.2 Formalization

Before we define the B-formalization of $\mathsf{NEXP} \notin \mathsf{P/poly}$, let us revisit the so-called direct formalization of Section 2 and adapt it to explicit machines. This adaptation will be referred to as the *A-formalization*.

Recall the sentence α^c_ψ from Definition 10. If M is an explicit NEXP with one input-tape and without oracles, then we write $\alpha^c_M := \alpha^c_\psi$ where $\psi(x)$ is the formula

$$\exists_2 Y \text{ “}Y \textit{ is an accepting computation of } M \textit{ on } x\text{”}. \tag{14}$$

Note that this is a $\hat{\Sigma}^{1,b}_1$-formula as required by Definition 10. As we argue below and is easy to see, the set of sentences $\{\neg\alpha^c_{M_0} \mid c \in \mathbb{N}\}$ is a formalization of $\mathsf{NEXP} \notin \mathsf{P/poly}$, this time in the language of machines.

We define now the *B-formalization*, which has lower quantifier complexity. This is based on the β^c_M-formulas that were defined in the introduction, and that we recall next. As we did before, we use number variables in capital letters C, D when we think of them as denoting circuits. Recall also the notations $C(x)$ and $D_x(\cdot)$ from Section 2.1.

Definition 22. Let $c \in N$ and let M be an explicit NEXP-machine with one input-tape and without oracles. Define

$$\begin{aligned}\beta^c_M \;:=\; & \forall n{\in}Log_{>1}\ \exists C{<}2^{n^c}\ \exists D{<}2^{n^c}\ \forall x{<}2^n\ \forall_2 Y \\ & (C(x){=}0 \ \rightarrow\ \neg\,\text{“}Y \textit{ is an accepting computation of } M \textit{ on } x\text{”}) \wedge \\ & (C(x){=}1 \ \rightarrow\ \text{“}D_x(\cdot) \textit{ is an accepting computation of } M \textit{ on } x\text{”}).\end{aligned}$$

We set

$$\text{“}\mathsf{NEXP} \notin \mathsf{P/poly}\text{”} \;:=\; \{\neg\beta^c_{M_0} \mid c \in \mathbb{N}\}.$$

In words, the formula $\beta^c_{M_0}$ says that the NEXP-machine M_0 for the canonical NEXP-complete problem Q_0 has witness circuits of encoding-size n^c.

Lemma 23. *For every $c \in \mathbb{N}$ and every explicit NEXP-machine M with one input-tape and without oracles, $\mathsf{S}^1_2(\alpha)$ proves $(\beta^c_M \rightarrow \alpha^c_M)$.*

Proof. The formula β^c_M states that the (single) existential set-quantifier in α^c_M is witnessed by $D_x(\cdot)$, and this set exists by $\Delta^b_1(\alpha)$-comprehension. $\square$

The next easy Proposition states that the formalizations introduced so far are indeed formalizations of $\mathsf{NEXP} \notin \mathsf{P/poly}$.

Proposition 24. *The following are equivalent.*

(a) $\mathsf{NEXP} \notin \mathsf{P/poly}$.

(b) $\{\neg\alpha^c_{M_0} \mid c \in \mathbb{N}\}$ *is true.*

(c) $\{\neg\alpha^c_{M} \mid c \in \mathbb{N}\}$ *is true for some explicit* NEXP*-machine* M.

(d) $\{\neg\beta^c_{M_0} \mid c \in \mathbb{N}\}$ *is true.*

(e) $\{\neg\beta^c_{M} \mid c \in \mathbb{N}\}$ *is true for some explicit* NEXP*-machine* M.

Proof. We show that (a)-(b)-(c) are equivalent, and that (a)-(d)-(e) are equivalent. To see that (a) implies (b), assume (b) fails; i.e., $\alpha^c_{M_0}$ is true for some $c \in \mathbb{N}$. Then $Q_0 \in \mathsf{SIZE}[n^c]$. As Q_0 is NEXP-complete, (a) fails. That (b) implies (c) is trivial since M_0 is an explicit NEXP-machine. That (c) implies (a) is obvious since every explicit NEXP-machine defines a language in NEXP. To see that (a) implies (d) argue as in the proof that (a) implies (b) swapping β for α. That (d) implies (e) is trivial since M_0 is an explicit NEXP-machine. Finally, that (e) implies (a) follows from the Easy Witness Lemma 2. $\square$

It is straightforward to see that the equivalences (b)-(c) and (d)-(e) in Proposition 24 have direct proofs (i.e., proofs that do not rely on the EWL). We use Lemma 21 to prove this on the formal level, for both formalizations.

Lemma 25. *For every $c \in \mathbb{N}$ and every 1-input explicit* NEXP*-machine M without oracles there is $d \in \mathbb{N}$ such that $\mathsf{S}^1_2(\alpha)$ proves $(\alpha^c_{M_0} \to \alpha^d_M)$ and $(\beta^c_{M_0} \to \beta^d_M)$.*

Proof. We refer to the implication between α's as the *α-case*, and to the implication between β's as the *β-case*. Both have similar proofs, so we prove them at the same time. Let M be witnessed by the term $t_M(x)$. Let $F(Z,x,u)$ and $G(Z,x,u)$ be the formulas given by Lemma 21 on M. Argue in $\mathsf{S}^1_2(\alpha)$ and assume $\alpha^c_{M_0}$ or $\beta^c_{M_0}$, as appropriate. Let $n \in Log_{>1}$ be given. We aim to find a circuit C in the α-case, and two circuits C, D in the β-case, witnessing α^e_M or β^e_M, respectively, for the given n, and for suitable $e \in \mathbb{N}$. Choose $d \in \mathbb{N}$ such that $|\langle M, x, t_M(x)\rangle| < n^d$ for all $x < 2^n$. In the α-case, let C_0 be a circuit with $|C_0| < m^c$ that witnesses $\alpha^c_{M_0}$ for $m := n^d$. In the β-case let C_0, D_0 be circuits with $|C_0|, |D_0| < m^c$ that witness $\beta^c_{M_0}$ for $m := n^d$.

Choose C such that $C(x) = C_0(\langle M, x, t_M(x)\rangle)$ and $e \in \mathbb{N}$ such that $C < 2^{n^e}$. This C will be the witness-circuit in the α-case, and the first of the two witness-circuits in the β-case. For the latter, we choose the second circuit D as follows. By Lemma 21.a, the set $F(Z,x,\cdot)$ is an accepting computation of M on x whenever Z is an accepting computation of M_0 on $\langle M, x, t_M(x)\rangle$. By Lemma 9 there is a circuit D such that

$$D(x,u) \leftrightarrow G(D_0(\langle M, x, t_M(x)\rangle, \cdot), x, u)$$

for all x, u with $x < 2^n$. Then $C, D < 2^{n^e}$ for suitable $e \in \mathbb{N}$. This is the $e \in \mathbb{N}$ we choose in the β-case.

We claim that C witnesses α^e_M for the given n in the α-case, and C, D witness β^e_M for the given n in the β-case. Let $x < 2^n$ and choose $z := \langle x, M, t_M(x)\rangle$. Let Z be any set and let $Y := F(Z,x,\cdot)$, which exists by $\Delta^b_1(\alpha)$-comprehension. If $C(x) = 0$, then $C_0(z) = 0$ and both $\alpha^c_{M_0}$ and $\beta^c_{M_0}$ imply that Y is not an accepting computation of M_0 on z. By Lemma 21.a this means that Z is not an accepting computation of M on x. In both cases, this completes one half of the verification of the witnesses. If $C(x) = 1$, then $C_0(z) = 1$ and $\alpha^c_{M_0}$ implies that there exists an accepting computation Y of M_0 on z, and $\beta^c_{M_0}$ implies that $Y := D_0(z,\cdot)$ is such an accepting computation of M_0 on z. But then Lemma 21.b implies that $Z := G(Y,x,\cdot)$, which exists by $\Delta^b_1(\alpha)$-comprehension, is an accepting computation of M on x. In both cases, this completes the other half of the verification of the witness: in the β-case, because $Z = D(x,\cdot)$. ☐

4.3 Consistency

M: added subsection heading to parallel Sec 5

For every explicit NEXP-machine M, which by default has one input-tape and no oracles, recall that $\alpha^c_M := \alpha^c_\psi$ for ψ taken as in (14). For a theory T that extends $\mathsf{S}^1_2(\alpha)$, consider the following *A-statements* for T:

A: $\mathsf{T} + \{\neg\alpha^c_M \mid c \in \mathbb{N}\}$ is consistent for some explicit NEXP-machine M,
A0: $\mathsf{T} + \{\neg\alpha^c_{M_0} \mid c \in \mathbb{N}\}$ is consistent.

Consider also the corresponding *B-statements* for T:

B: $\mathsf{T} + \{\neg\beta^c_M \mid c \in \mathbb{N}\}$ is consistent for some explicit NEXP-machine M,
B0: $\mathsf{T} + \{\neg\beta^c_{M_0} \mid c \in \mathbb{N}\}$ is consistent.

Next, recall the statement of Proposition 1, which we now state for an arbitrary theory T that extends $\mathsf{S}^1_2(\alpha)$. We refer to it as the *C-statement*, or the *direct consistency statement* for T:

C: $\mathsf{T} + \{\neg\alpha^c_\psi \mid c \in \mathbb{N}\}$ is consistent for some $\hat{\Sigma}^{1,b}_1$-formula $\psi(x)$.

Let us explicitly point out that the formula $\psi(x)$ of the C-statement has only one free variable of the number sort, and no free variables of the set sort.

We view the following proposition as justification that our formalization is faithful. It takes record of which implications in Proposition 24 hold over weak theories.

Proposition 26. *Let T be a theory extending $\mathsf{S}^1_2(\alpha)$ and consider the A,B,C-statements for T. Then, the following hold: the A-statements are equivalent, the B-statements are equivalent, and both A-statements imply both B-statements as well as the C-statement.*

Proof. Each A-statement implies the corresponding B-statement by Lemma 23. Further, Lemma 25 proves that the A-statements are equivalent, and that the B-statements are equivalent; for the back implications note that M_0 is certainly an explicit NEXP-machine. Further, it is obvious from the definition of α^c_M that A implies C and hence both A-statements imply C. □

When $\mathsf{T} = \mathsf{V}^0_2$, we argue below that the model-checker lemmas can be used to show that the implication A-to-C in Proposition 26 can be reversed. It will follow that all A,B,C-statements for V^0_2 are equivalent. Composing with Proposition 1 we get the following corollary, which entails Theorem 3.

Theorem 27. *For $\mathsf{T} = \mathsf{V}^0_2$ all statements C, A, A0, B, B0 are true.*

Proof. Proposition 1 states that C is true for $\mathsf{T} = \mathsf{V}^0_2$. Hence, by Proposition 26, it suffices to show that C implies A for $\mathsf{T} = \mathsf{V}^0_2$. But this follows from Lemma 20.a and 20.b. Indeed, this lemma implies that every $\hat{\Sigma}^{1,b}_1$-formula $\psi(x)$ is V^0_2-provably equivalent to a formula of the form (14) for suitable M. □

5 Consistency for barely superpolynomial time

In this section we fix $r \in \mathsf{PV}$ such that

(r0) the function $x \mapsto r(x)$ is computable in time $O(r(x))$;

(r1) $\mathsf{S}^1_2 \vdash (|x|=|y| \to r(x)=r(y))$;

(r2) $\mathsf{S}^1_2 \vdash (|x|<|y| \to r(x)<r(y))$;

(r3) for every polynomial p there is $f \in \mathsf{PV}$ such that $\mathsf{S}^1_2 \vdash p(r(x)) \leqslant r(f(x))$;

(r4) for every $c \in \mathbb{N}$ there is $n_c \in \mathbb{N}$ such that $\mathbb{N} \models \forall x\ (|x|{>}n_c \to r(x){>}|x|^c)$.

A function r that satisfies (r4) is called *length-superpolynomial*.

An *explicit* $\mathsf{NTIME}(\mathsf{poly}(r(x)))$*-machine* is an explicit NEXP-machine M that is witnessed by $p(r(x))$ for some polynomial p. Here, we deviate from our convention that explicit machines are witnessed by terms and allow PV-symbols. In the notation $\mathsf{NTIME}(\mathsf{poly}(r(x)))$, the x is there to emphasize that the runtime is measured as a function of the input x and not its length. If we want to measure runtime as a function of the length of the input, then we use n instead of x. For example, $\mathsf{NP} = \mathsf{NTIME}(n^{O(1)})$ is given by the collection of explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machines with $r(x) = |x|$, and the classes $\mathsf{NE} = \mathsf{NTIME}(2^{O(n)})$ and $\mathsf{NTIME}(n^{O(\log^{(k)} n)})$ are given by the collections of explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machines for $r(x) = 2^{|x|}$ and $r(x) = |x|^{\log^{(k)}|x|}$, respectively; the latter two satisfy (r0)-(r4), if $k \geqslant 1$ in the second.

Remark 28. (r3) is not implied by the other conditions.

Proof. We shall define a function $r(x)$ which consists of slow growing segments interspersed with fast growing segments. First, choose a fast growing function $R \in \mathsf{PV}$ so that $R(x)$ depends only on $|x|$ and so that $R(x)^2 \geqslant R(x) + |x|^{\omega(1)}$. For instance $R(x) = 2^{|x|}$ works. Second, define $\ell : \mathbb{N} \to \mathbb{N}$ be increasing with $\ell(c+1) > \ell(c)^c + 1$ and with $R(x)^2 \geqslant R(x) + |x|^c$ for all $x \geqslant 2^{\ell(c)-1}$. Let $x_c := 2^{\ell(c)-1}$ and $y_c := 2^{\ell(c)^c} - 1$ be the first and last numbers of length $\ell(c)$ and $\ell(c)^c$, respectively. Finally, let $r(x) := R(x_c) + |x| - |x_c|$ for $x_c \leqslant x \leqslant y_c$, and let $r(x) := R(x)$ for $y_c < x < x_{c+1}$. The slow growing segments of $r(x)$ are where $x_c \leqslant x \leqslant y_c$, and here $r(x)$ is chosen to be as slow growing as possible while satisfying (r1) and (r2).

Clearly, ℓ and R can be chosen so that $r(x)$ is in PV and properties (r0), (r1), (r2), and (r4) hold for r. We claim (r3) fails for $p(x) = x^2$.

Indeed, let $f \in \mathsf{PV}$ be given and choose c such that $|f(x_c)| < |x_c|^c = |y_c|$. Then

$$p(r(x_c)) = r(x_c)^2 = R(x_c)^2 \geqslant R(x_c) + |x_c|^c = R(x_c) + |y_c| > r(y_c) > r(f(x_c))$$

where the last inequality follows from (r2). ☐

5.1 A more general universal machine

We start with the analogue of Lemma 21.

Lemma 29. *There is an explicit* $\mathsf{NTIME}(\mathsf{poly}(r(x)))$*-machine* M_r *with one input-tape and without oracles such that for every explicit* $\mathsf{NTIME}(\mathsf{poly}(r(x)))$*-machine* M *with one input-tape and without oracles there are* $f_M(x) \in \mathsf{PV}$ *and quantifier-free* $\mathsf{PV}(\alpha)$*-formulas* F_M *and* G_M *such that*

(a) $\mathsf{S}^1_2(\alpha) \vdash$ *"Z is an accepting computation of M on x" →*
"$F_M(Z,x,\cdot)$ is an accepting computation of M_r on $\langle M, x, f_M(x)\rangle$".

(b) $\mathsf{S}^1_2(\alpha) \vdash$ *"Z is an accepting computation of M_r on $\langle M, x, f_M(x)\rangle$" →*
"$G_M(Z,x,\cdot)$ is an accepting computation of M on x",

In particular,

$$\text{(c) } \mathsf{S}^1_2(\alpha) \vdash \begin{array}{l} \exists_2 Z \text{ “Z is an accepting computation of M_r on $\langle M, x, f_M(x)\rangle$” } \leftrightarrow \\ \exists_2 Z \text{ “Z is an accepting computation of M on x”} \end{array}$$

Proof. Choose according to Lemma 14 a machine M^Z_φ and a term $r_\varphi(N,x,t)$ for

$$\varphi(Z,N,x,t) := \text{“Z is an accepting time-t computation of N on x”.}$$

By the comment after Equation (6), there is a polynomial p_1 so that $bt_\varphi(N,x,t) \leqslant p_1(t,|N|,|x|)$ provably in S^1_2. By Lemma 14.d, there is a polynomial p_2 so that $r_\varphi(N,x,t) \leqslant p_2(t,|N|,|x|)$ provably in S^1_2. For M^Z_φ choose a machine M_1 and a term $r_1(N,x,t)$ according to Lemma 19. By Lemma 19.c, there is a polynomial p_3 so that $r_1(N,x,t) \leqslant p_3(t,|N|,|x|)$.

Define M_r to compute on z as follows. It first checks that $z = \langle N,x,t\rangle$ for certain N,x,t and computes $\langle N,x,r(t)\rangle$; if the check fails, the machine stops. After this *initial* computation M_r runs M_1 on $\langle N,x,r(t)\rangle$. The initial computation can be implemented with explicit P-machines (Lemma 13), say with time bound $p_4(|z|)$ for a polynomial p_4. Then M_r is an explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machine. Indeed, it is witnessed by $p_4(|z|)+p_3(r(z),|z|,|z|) \leqslant p_5(r(z))$ for a polynomial p_5. Here we use that S^1_2-provably t,N,x are bounded by z, and r is non-decreasing with $r(x) \geqslant |x|$ by (r1) and (r2).

Let M be an explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machine, say witnessed by $p_M(r(x))$ for a polynomial p_M. Choose f_M for p_M according to (r3).

For (a), argue in S^1_2 and assume Z is an accepting computation of M on x. Then Z is time-$p_M(r(x))$, so by (r3) we can repeat the halting configuration to get an accepting time-$r(f_M(x))$ computation Z_0 of M on x, i.e., $\varphi(Z_0,M,x,r(f_M(x)))$ holds. By Lemma 14.g, the set $Z_1 := C_\varphi(Z_0,M,x,r(f_M(x)),t_\varphi(M,x,r(f_M(x))),\cdot)$ is an accepting computation of $M^{Z_0}_\varphi$ on the triple $M,x,r(f_M(x))$. By Lemma 19.a, the set $Z_2 := F(Z_1,Z_0,M,x,r(f_M(x)),\cdot)$ is an accepting computation of M_1 on the triple $M,x,r(f_M(x))$. Compose Z_2 with an initial computation of M_r on $z := \langle M,x,f_M(x)\rangle$ to get an accepting computation Z_3 of M_r on z. It is clear that $Z_3 = F_M(Z,x,\cdot)$ for some quantifier-free $\mathsf{PV}(\alpha)$-formula F_M.

For (b), argue in S^1_2 and let Z be an accepting computation of M_r on $\langle M,x,f_M(x)\rangle$. From Z extract an accepting computation Z_0 of M_1 on the triple $M,x,r(f_M(x))$. By Lemma 19.b, $Z_1 := G(Z_0,M,x,r(f_M(x)),\cdot)$ is an accepting computation of $M^{Z_2}_\varphi$ on the triple $M,x,r(f_M(x))$ where $Z_2 := H(Z_0,M,x,r(f_M(x)),\cdot)$. Clearly, Z_0 can be described by a quantifier-free $\mathsf{PV}(\alpha)$-formula, so Z_1 and Z_2 exist by $\Delta^b_1(\alpha)$-comprehension. Hence, by Lemma 14.a, $\varphi(Z_2,M,x,r(f_M(x)))$ holds, i.e., Z_2 is an accepting time-$r(f_M(x))$ computation of M on x. By (r3) we can shrink Z_2 to time $p_M(r(x))$ and get an accepting computation Z_3 of M on x. Clearly, $Z_3 = G_M(Z,x,\cdot)$ for some quantifier-free $\mathsf{PV}(\alpha)$-formula G_M.

Finally, (c) follows from (a) and (b) by $\Delta^b_1(\alpha)$-comprehension. ☐

5.2 Formalization

To faithfully formalize $\mathsf{NTIME}(\mathsf{poly}(r(x))) \notin \mathsf{P/poly}$ we intend to follow the path paved in Section 4. Some modification are, however, required. First, we need an analogue of the Easy Witness Lemma. This has been achieved by Murray and Williams [27]:

Lemma 30. *Let $t(n)$ be a function that is increasing, time-constructible, and superpolynomial. If* $\mathsf{NTIME}(\mathsf{poly}(t(n))) \subseteq \mathsf{P/poly}$*, then every* $\mathsf{NTIME}(\mathsf{poly}(t(n)))$*-machine M has polynomial-size witness circuits.*

That $t(n)$ is *superpolynomial* means that for every $c \in \mathbb{N}$ there is $n_c \in \mathbb{N}$ such that $t(n) > n^c$ for all $n > n_c$. That M has *witness circuits of size* $s(n)$, where $s : \mathbb{N} \to \mathbb{N}$ is a function, means that for every $x \in \{0,1\}^*$ that is accepted by M, there exists a circuit D of size at most $s(|x|)$ such that $tt(D)$ encodes an accepting computation of M on x. Note that, in contrast to Lemma 2, the circuit D can depend on x. We do not know whether Lemma 30 holds true for *oblivious* witness circuits as in Lemma 2.

Lemma 30 follows from the central result of [27]:

Lemma 31 (Lemma 4.1 in [27])**.** *There are $e, g \in \mathbb{N}$ with $e, g \geqslant 1$ such that for all increasing time-constructible functions $s(n)$ and $t(n)$, and for $s_2(n) := s(en)^e$, if* $\mathsf{NTIME}(O(t(n)^e)) \subseteq \mathsf{SIZE}(s(n))$*, then every* $\mathsf{NTIME}(t(n))$*-machine has witness circuits of size $s_2(s_2(s_2(n)))^{2g}$, provided that $s(n) < 2^{n/e}/n$ and $t(n) \geqslant s_2(s_2(s_2(n)))^d$ for a sufficiently large $d \in \mathbb{N}$.*

Proof of Lemma 30 from Lemma 31. We start noting that there is a non-deterministic machine U that decides the problem Q_0 defined in Section 4.1 in time $O(|x| + |M| \cdot t^2)$ on input $\langle M, x, t\rangle$: after reading the input, guess the non-deterministic choices of M and deterministically in time $c_M \cdot t^2$ simulate the computation path of M on input x as determined by those choices, where c_M is a simulation overhead constant that depends only on M and that we may assume is at most $|M|$.

Assume $\mathsf{NTIME}(\mathsf{poly}(t(n))) \subseteq \mathsf{P/poly}$. Fix $c \in \mathbb{N}$ with $c \geqslant 1$ and an $\mathsf{NTIME}(t(n)^c)$-machine M. We intend to apply Lemma 31 to M for a suitably chosen $s(n)$, with $t(n)^c$ in the role of $t(n)$. For that, we will need to show that $\mathsf{NTIME}(O(t(n)^{ce})) \subseteq \mathsf{SIZE}(s(n))$ for the chosen $s(n)$, where $e \geqslant 1$ is the first of the two constants in Lemma 31.

The restriction of U to inputs of the form $\langle M, x, t(|x|)^{ce+1}\rangle$ runs in time $O(|x| + |M| \cdot t(|x|)^{2ce+2})$. Therefore, the set of pairs $\langle M, x\rangle$ such that U accepts on input $\langle M, x, t(|x|)^{ce+1}\rangle$ is in $\mathsf{NTIME}(\mathsf{poly}(t(n)))$, so by the assumption, it is decided by circuits of size $p(|\langle M, x\rangle|)$ for a suitable polynomial $p(n)$.

Now, choose $s(n)$ as a polynomial such that for every non-deterministic Turing machine M and every x that is sufficiently long with respect to M it holds that $p(|\langle M, x\rangle|) < s(|x|)$. We verify that $\mathsf{NTIME}(O(t(n)^{ce})) \subseteq \mathsf{SIZE}(s(n))$: If B is a set in $\mathsf{NTIME}(O(t(n)^{ce}))$ and M is a non-deterministic Turing machine that witnesses this, then, for sufficiently long x, we have that x is in B if and only if U accepts on $\langle M, x, t(|x|)^{ce+1}\rangle$. Hence, by the choice of $s(n)$, the set B is in $\mathsf{SIZE}(s(n))$.

The requirements of Lemma 31 that $s(n) < 2^{n/e}/n$ and $t(n)^c \geqslant s_2(s_2(s_2(n)))^d$ for a sufficiently large constant $d \in \mathbb{N}$ are obviously met because $s(n)$ is polynomially bounded and $t(n)$ is superpolynomial. Lemma 31 applied to $s(n)$ and $t(n)^c$ then gives that M has witness circuits of size $s_2(s_2(s_2(n)))^{2g}$, where $g \geqslant 1$ is the second of the two constants in Lemma 31. Since $s(n)$ is polynomially bounded, also this function is polynomially bounded. Thus, M has polynomial-size witness circuits. $\square$

Lemma 30 enables the following $\forall\Pi_1^{1,b}$-formalization of $\mathsf{NTIME}(\mathsf{poly}(r(x))) \notin \mathsf{P/poly}$:

Definition 32. For an explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machine M with one input-tape and without oracles define

$$\begin{aligned}\gamma_M^c \;:=\; & \forall n{\in}Log_{>1}\ \exists C{<}2^{n^c}\ \forall x{<}2^n\ \exists D{<}2^{n^c}\ \forall_2 Y \\ & (C(x){=}0 \ \to\ \neg\text{“}Y \textit{ is an accepting computation of } M \textit{ on } x\text{”}) \wedge \\ & (C(x){=}1 \ \to\ \text{“}D(\cdot) \textit{ is an accepting computation of } M \textit{ on } x\text{”}).\end{aligned}$$

Let M_r be the explicit $\mathsf{NTIME}(\mathsf{poly}(r(x))))$-machine of Lemma 29. Define

$$\text{“}\mathsf{NTIME}(\mathsf{poly}(r(x))) \notin \mathsf{P/poly}\text{”} \;:=\; \{\neg\gamma_{M_r}^c \mid c \in \mathbb{N}\}.$$

The following is the analogue of Lemma 23 and is similarly proved.

Lemma 33. *For every $c \in \mathbb{N}$ and every explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machine M with one input-tape and without oracles, $\mathsf{S}_2^1(\alpha)$ proves $(\gamma_M^c \to \alpha_M^c)$.*

Lemma 34. *For every $c \in \mathbb{N}$ and every explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machine M with one input-tape and without oracles there is $d \in \mathbb{N}$ such that $\mathsf{S}_2^1(\alpha)$ proves $(\alpha_{M_r}^c \to \alpha_M^d)$ and $(\gamma_{M_r}^c \to \gamma_M^d)$.*

Proof. This is proved similarly as Lemma 25. We only treat the γ-case. Choose $f_M(x) \in \mathsf{PV}$ according to Lemma 29. Argue in $\mathsf{S}_2^1(\alpha) + \gamma_{M_r}^c$. Let $n \in Log_{>1}$ be given. Choose $e \in \mathbb{N}$ such that $|\langle M, x, f_M(x)\rangle| < n^e$ for all $x < 2^n$. Choose C_0 witnessing $\gamma_{M_r}^c$ for $m := n^e$. Choose a circuit C such that $C(x) = C_0(\langle M, x, f_M(x)\rangle)$ for all $x < 2^n$. We shall choose d large enough such that $C \leqslant 2^{n^d}$ and choose C to witness the first existential quantifier in γ_M^d for n. To verify this choice, let $x < 2^n$ be given.

If $C(x) = 0$, then there are no accepting computations of M_r on $\langle M, x, f_M(x)\rangle$. By Lemma 29.a and $\Delta_1^b(\alpha)$-comprehension, there are no accepting computations of M on x. If $C(x) = 1$, then there is a circuit $D_0 < 2^{m^c}$ such that $D_0(\cdot)$ is an accepting computation of M_r on $\langle M, x, f_M(x)\rangle$. By Lemma 29.b, $G_M(D_0(\cdot), x, \cdot)$ is an accepting computation of M on x. By Lemma 9 there is a circuit D such that $\big(D(u) \leftrightarrow G_M(D_0(\cdot), x, u)\big)$ for all $u \leqslant \langle p_M(r(x)), p_M(r(x), |M|)\rangle$ where p_M is a polynomial such that $p_M(r(x))$ witnesses M. Choose $d \in \mathbb{N}$ large enough such that $D < 2^{n^d}$. ☐

Finally, we are in the position to verify that the formulas considered formalize the intended circuit lower bound.

Proposition 35. *The following are equivalent.*

(a) $\mathsf{NTIME}(\mathsf{poly}(r(x))) \notin \mathsf{P/poly}$.

(b) $\{\neg\alpha_{M_r}^c \mid c \in \mathbb{N}\}$ *is true.*

(c) $\{\neg\alpha_M^c \mid c \in \mathbb{N}\}$ *is true for some explicit* $\mathsf{NTIME}(\mathsf{poly}(r(x)))$*-machine* M.

(d) $\{\neg\gamma_M^c \mid c \in \mathbb{N}\}$ *is true for some explicit* $\mathsf{NTIME}(\mathsf{poly}(r(x)))$*-machine* M.

(e) $\{\neg\gamma_{M_r}^c \mid c \in \mathbb{N}\}$ *is true.*

Proof. To see that (a) implies (b), assume (b) fails, so $\alpha^c_{M_r}$ is true for some $c \in \mathbb{N}$. Then the problem accepted by M_r is in $\mathsf{SIZE}[n^c]$. By Lemma 29 this problem is $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-hard under polynomial time reductions. Since $\mathsf{P/poly}$ is downward-closed under polynomial-time reductions, (a) fails. The claim that (b) implies (c) is trivial since M_r is an explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machine. That (c) implies (d) follows from Lemma 33. That (d) implies (e) follows from Lemma 34. That (e) implies (a) follows from Lemma 30: by (r1) there is a function $t(n)$ such that $t(|x|) = r(x)$ for every x; then $\mathsf{NTIME}(\mathsf{poly}(r(x))) = \mathsf{NTIME}(\mathsf{poly}(t(n)))$ where the time-bound on the left is written as a function of the input x and on the right as a function of its length $n = |x|$; further, $t(n)$ is time-constructible by (r0) and (r1), increasing by (r2) and superpolynomial by (r4). ◻

5.3 Consistency

For a theory T that extends $\mathsf{S}^1_2(\alpha)$, the new A,B-statements are the following:

A$_r$: $\mathsf{T} + \{\neg\alpha^c_M \mid c \in \mathbb{N}\}$ is consistent for some explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machine M,
B$_r$: $\mathsf{T} + \{\neg\gamma^c_M \mid c \in \mathbb{N}\}$ is consistent for some explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machine M,
A0$_r$: $\mathsf{T} + \{\neg\alpha^c_{M_r} \mid c \in \mathbb{N}\}$ is consistent.
B0$_r$: $\mathsf{T} + \{\neg\gamma^c_{M_r} \mid c \in \mathbb{N}\}$ is consistent.

To define the corresponding C-statement, we say that the bounding term of a $\hat{\Sigma}^{1,b}_1$-formula $\psi = \psi(x)$ is *polynomial in* $r(x)$ if S^1_2 proves $bt(\psi) \leqslant p(r(x))$ for some polynomial $p(n)$. Then:

C$_r$: $\mathsf{T} + \{\neg\alpha^c_\psi \mid c \in \mathbb{N}\}$ is consistent for some $\hat{\Sigma}^{1,b}_1$-formula $\psi = \psi(x)$ whose bounding term is polynomial in $r(x)$.

Before we prove the analogue of Theorem 27 we state the proof complexity lower bound on which it is based. Recall the Pigeonhole Principle formula $PHP(x)$ from the proof of Proposition 1. The first strong lower bounds on the provability of $PHP(x)$ were due to Ajtai [1]; here we need the later quantitative improvements from [3]. This can be called the *gem* of proof complexity. We use it in the following form. Recall that a function is called length-superpolynomial when it satisfies (r4).

Theorem 36 (Gem Theorem)**.** *For every length-superpolynomial* PV*-function* $s(x)$*, the theory* V^0_2 *does not prove* $PHP(s(x))$*.*

Proof. Consider the Paris-Wilkie propositional translations $F_n := \langle PHP(s(n))\rangle_n$ for $n \in \mathbb{N}$; see [22, Definition 9.1.1] in the form used in [22, Corollary 9.1.4]. Assume for contradiction that $PHP(s(x))$ is provable in V^0_2. Then, there exist constants $c, d \in \mathbb{N}$ such that for every sufficiently large $n \in \mathbb{N}$, the propositional formulas F_n have Frege proofs of depth d and size $2^{|n|^c}$: apply [22, Corollary 9.1.4] with the function $f(x) = x\#x$ and note that V^0_2 is conservative over the theory considered there: from a model of that theory, get a model of V^0_2 by just adding all bounded sets that are definable by bounded formulas.

Now, let $n \in \mathbb{N}$ be large enough to ensure this upper bound and at the same time such that $s(n) > |n|^{6^d c}$, which exists because $s(x)$ is length-superpolynomial. Setting $m := s(n)$, this means that the propositional formula $PHP^{m+1}_m := F_n$ has Frege proofs of depth d and size bounded by an exponential in $m^{1/6^d}$. It is well-known that if m is sufficiently large, then this is false; see [22, Theorem 12.5.3]. $\square$

Finally we can prove the analogue of Theorem 27, which entails Theorem 4.

Theorem 37. *For* $\mathsf{T} = \mathsf{V}^0_2$*, all statements* C_r*,* A_r*,* $A0_r$*,* B_r*,* $B0_r$ *are true.*

Proof. The analogue of Proposition 26 for the A_r,B_r,C_r-statements has the same proof using Lemmas 33, 34 in place of Lemmas 23, 25. Note that the claim that A_r implies C_r follows from the remark after Equation (8). As in the proof Theorem 27, that C_r implies A_r for $\mathsf{T} = \mathsf{V}^0_2$ follows from Lemma 20.a and 20.b. We also need 20.c along with $r(x) \geqslant |x|$ by (r1) and (r2) to guarantee that the explicit NEXP-machine is an explicit $\mathsf{NTIME}(\mathsf{poly}(r(x)))$-machine.

We are left to show that C_r holds for $\mathsf{T} = \mathsf{V}^0_2$. This is proved by tightening the choice of parameters in the argument that proved Proposition 1.

Consider the formula

$$y \leqslant r(x) \wedge \neg PHP(y) \tag{15}$$

and write this as $\psi = \psi(z)$, where $z = \langle x, y\rangle$; i.e., $x = \pi_1(z)$ and $y = \pi_2(z)$ with π_1 and π_2 as PV-functions. The formula $\psi(z)$ is logically equivalent to a $\hat{\Sigma}^{1,b}_1$-formula whose bounding term is polynomial in $r(z)$ by (r1) and (r2). We claim that $\mathsf{V}^0_2 + \{\neg\alpha^c_\psi \mid c \in \mathbb{N}\}$ is consistent, which will give C_r.

For the sake of contradiction, assume otherwise. By compactness, there exists $c \in \mathbb{N}$ such that V^0_2 proves α^c_ψ. As in the proof of Proposition 1, we show that this implies that V^0_2 proves $PHP(r(x))$, which contradicts the Gem Theorem by (r4).

Argue in V^0_2 and set $n := \max\{|z|, 2\}$, where $z = \langle x, r(x)\rangle$. Then α^c_ψ on n gives a circuit C such that, for all $u \leqslant z$ and $v \leqslant z$ with $\langle u, v\rangle \leqslant z$, we have

$$\neg C(\langle u, v\rangle) \leftrightarrow (v \leqslant r(u) \to PHP(v)).$$

Noting that $\langle x, v\rangle \leqslant z$ for all $v \leqslant r(x)$, fix u to x in the circuit $C(\langle u, v\rangle)$ and get a circuit $D(v)$ such that

$$\forall v \leqslant r(x)\ (\neg D(v) \leftrightarrow PHP(v)).$$

Recall that V^0_2 proves that $PHP(x)$ is inductive. Hence, plugging $\neg D(v)$ for $PHP(v)$ gives $PHP(r(x))$ by quantifier-free $\mathsf{PV}(\alpha)$-induction. $\square$

6 Magnification

For this section, a $\exists_2\Pi^b_1(\alpha)$-formula is a $\hat{\Sigma}^{1,b}_1$-formula as in (4) in which its maximal $\Sigma^{1,b}_0$-subformula $\varphi(\bar{X}, Y, \bar{x})$ is a $\Pi^b_1(\alpha)$-formula.

Lemma 38. *For every* $c \in \mathbb{N}$ *and every* $\exists_2\Pi^b_1(\alpha)$*-formula* $\psi(\bar{x}, y)$ *without free set variables, the theory* $\mathsf{S}^1_2(\alpha) + \beta^c_{M_0}$ *proves*

$$\exists C\ \forall y \leqslant z\ \big(C(y){=}1 \leftrightarrow \psi(\bar{x}, y)\big). \tag{16}$$

Proof. Argue in $\mathsf{S}^1_2(\alpha) + \beta^c_{M_0}$. For simplicity assume $\bar{x}$ is empty; the general case can be handled by taking tuples through Cantor pairing. For $\psi = \psi(y)$ choose $M := N_\psi$ according to Lemma 20. Note that since ψ does not have free set variables, M is without oracles. By Lemma 20.e, the formula $\psi(y)$ is equivalent to

$$\exists_2 Y \text{ “}Y \textit{ is an accepting computation of } M \textit{ on } y\text{”}.$$

By Lemmas 23 and 25 we have α^d_M for some $d \in \mathbb{N}$. Let z be given and choose $n \in \mathit{Log}_{>1}$ with $|z| \leqslant n$. Let C witness α^d_M for n. This C witnesses (16). □

It follows that over $\mathsf{S}^1_2(\alpha)$ the circuit upper bound statement $\beta^c_{M_0}$ implies comprehension for $\exists_2\Pi^b_1(\alpha)$-formulas *without free set variables*. For later reference, we note that allowing free set variables entails full $\hat{\Sigma}^{1,b}_1$-comprehension:

Lemma 39. $\mathsf{S}^1_2(\alpha) + \exists_2\Pi^b_1(\alpha)$*-comprehension proves* V^1_2.

Proof. Let T denote $\mathsf{S}^1_2(\alpha) + \exists_2\Pi^b_1(\alpha)$-comprehension. Since $\mathsf{S}^1_2(\alpha) + \Sigma^{1,b}_1$-comprehension proves V^1_2, it suffices to show that the set of formulas that are T-provably equivalent to a $\exists_2\Pi^b_1(\alpha)$-formula is closed under $\vee, \wedge, \exists_2 Y, \exists y{\leqslant}t(\bar{x})$ and $\forall y{\leqslant}t(\bar{x})$. We verify the latter: the formula

$$\forall y{\leqslant}u\ \exists_2 Y\ \varphi(\bar{X}, Y, \bar{x}, u, y)$$

with $\varphi(\bar{X}, Y, \bar{x}, u, y)$ a $\Pi^b_1(\alpha)$-formula is T-provably equivalent to

$$\exists_2 Z\ \forall y{\leqslant}u\ \varphi(\bar{X}, Z(y,\cdot), \bar{x}, u, y),$$

where $Z(y,v)$ abbreviates the atomic formula $\langle y, v\rangle \in Z$. Indeed, assuming the former formula, the latter is proved by induction on u. As the latter is a $\exists_2\Pi^b_1(\alpha)$-formula, induction for it follows from comprehension. □

The following lemma makes precise the idea sketched in Section 1.3.

Lemma 40. *For every $c \in \mathbb{N}$ and every model $(M, \mathcal{X})$ of $S^1_2(\alpha) + \beta^c_{M_0}$, there exists $\mathcal{Y} \subseteq \mathcal{X}$ such that $(M, \mathcal{Y})$ is a model of* V^1_2.

Proof. By $\Delta^b_1(\alpha)$-comprehension, for every $C \in M$ that is a circuit in the sense of M there is a set $A \in \mathcal{X}$ such that

$$(M, \mathcal{X}) \vDash \forall y\ (C(y){=}1 \leftrightarrow y{\in}A).$$

By extensionality such a set A is uniquely determined by C and we write $\hat{C}$ for it. For these two claims we used the fact that $C(y){=}1 \to y{<}2^{|C|}$ holds in every model of S^1_2.

Let

$$\mathcal{Y} := \left\{\hat{C} \in \mathcal{X} \mid C \in M \textit{ is a circuit in the sense of } M\right\}.$$

Since $\mathcal{Y} \subseteq \mathcal{X}$, the model $(M, \mathcal{Y})$ satisfies all $\Pi^{1,b}_1$-sentences which are true in $(M, \mathcal{X})$, so in particular extensionality, set boundedness, $\Sigma^b_1(\alpha)$-induction, and $\beta^c_{M_0}$.

The point of the model $(M, \mathcal{Y})$ is that it eliminates set parameters. More precisely, let $\varphi(\bar{x})$ be a $\Sigma^{1,b}_\infty$-formula with parameters from $(M, \mathcal{Y})$, and define $\varphi^*(\bar{x})$ as follows: replace every subformula of the form $t{\in}\hat{C}$ where t is a term (possibly with number parameters from M) and $\hat{C}$ is a set parameter from $\mathcal{Y}$ by $C(t){=}1$ (i.e., by $eval(C,t){=}1$). Note every set parameter in $\varphi(\bar{x})$ becomes a number parameter in $\varphi^*(\bar{x})$, and

$$(M,\mathcal{Y}) \vDash \forall\bar{x}\ (\varphi(\bar{x}) \leftrightarrow \varphi^*(\bar{x})). \tag{17}$$

Claim: $(M,\mathcal{Y}) \vDash \mathsf{S}^1_2(\alpha)$.

Proof of the Claim. It suffices to show that $(M,\mathcal{Y})$ models $\Delta^b_1(\alpha)$-comprehension. So let $\varphi(x)$ be a $\Delta^b_1(\alpha)$-formula with parameters from $(M,\mathcal{Y})$ and $a\in M$. Then $\varphi^*(x)$ is a number-sort formula, namely a Δ^b_1-formula with (number) parameters from M. Since $M\vDash\mathsf{S}^1_2$, Buss' witnessing theorem implies that $\varphi^*(x)$ is equivalent in M to a quantifier-free PV-formula with the same parameters. Lemma 9 applied to $n := \max\{|a|,2\}$ gives a circuit C in the sense of M such that

$$M \vDash \forall x{<}2^n(C(x)=1 \leftrightarrow \varphi^*(x)).$$

Then $\hat{C}\in\mathcal{Y}$ and $(M,\mathcal{Y})$ satisfies $\forall y{\leqslant}a(y\in\hat{C} \leftrightarrow \varphi(y))$ by (17). ⊣

By the Claim and Lemma 39, it suffices to show that $(M,\mathcal{Y})$ has $\exists_2\Pi^b_1(\alpha)$-comprehension. Let $\psi(x)$ be a $\exists_2\Pi^b_1(\alpha)$-formula with parameters from $(M,\mathcal{Y})$, and let $a\in M$. Then $\psi^*(x)$ is a $\exists_2\Pi^b_1(\alpha)$-formula without set parameters. We already noted that $(M,\mathcal{Y})\vDash\beta^c_{M_0}$. Hence, by the Claim, Lemma 38 applies and gives $C\in M$ such that

$$(M,\mathcal{Y}) \vDash \forall x{\leqslant}a\ (C(x){=}1 \leftrightarrow \psi^*(x)).$$

Then $\hat{C}\in\mathcal{Y}$ and $(M,\mathcal{Y})$ satisfies $\forall x{\leqslant}a\ (x{\in}\hat{C} \leftrightarrow \psi(x))$ by (17). □

As announced in Section 1.3 this lemma implies Corollaries 5 and 6.

Proof of Corollary 5. Assume that T is inconsistent with "$\mathsf{NEXP}\notin\mathsf{P/poly}$". By compactness, the theory T proves $\beta^c_{M_0}$ for some $c\in\mathbb{N}$. Let ψ be a number sort consequence of V^1_2 and $(M,\mathcal{X})$ a model of T. We have to show that $M\vDash\psi$. But by Lemma 40 there exists $\mathcal{Y}\subseteq\mathcal{X}$ such that $(M,\mathcal{Y})\vDash\mathsf{V}^1_2$, so $(M,\mathcal{Y})\vDash\psi$, and $M\vDash\psi$. □

Proof of Corollary 6. Assume $\mathsf{S}^1_2(\alpha)$ does not prove "$\mathsf{NEXP}\notin\mathsf{P/poly}$", say, it does not prove $\neg\beta^c_{M_0}$. Then there is a model $(M,\mathcal{X})$ of $\mathsf{S}^1_2(\alpha)+\beta^c_{M_0}$. By Lemma 40 there exists $\mathcal{Y}\subseteq\mathcal{X}$ such that $(M,\mathcal{Y})\vDash\mathsf{V}^1_2$. Since $\beta^c_{M_0}$ is a $\Pi^{1,b}_1$-formula, we have $(M,\mathcal{Y})\vDash\beta^c_{M_0}$. Thus, V^1_2 does not prove "$\mathsf{NEXP}\notin\mathsf{P/poly}$". □

Remark 41. The introduction mentioned that Corollary 6 might raise hopes to complete Razborov's program by construcing a model of $\mathsf{S}^1_2(\alpha)$ satisfying some $\beta^c_{M_0}$. There are good general methods to construct models even of certain extensions of $\mathsf{T}^1_2(\alpha)$ based on forcing (see [35] and [25] for an extension). However, these methods are tailored for $\hat{\Sigma}^{1,b}_1(\alpha)$-statements, not $\Pi^{1,b}_1$ like $\beta^c_{M_0}$. By the method of feasible interpolation and assuming the existence of suitable pseudorandom generators, Razborov [33] proved that for every Σ^b_∞-definable $t(n)=n^{\omega(1)}$ and every Σ^b_∞-formula $\varphi(x)$ there exists a model $(M,\mathcal{X})$ of $\mathsf{S}^2_2(\alpha)$ that for some $n\in M$ contains a set $C\in\mathcal{X}$ coding a size-$t(n)$ circuit that computes $\varphi(x)$; i.e., for every $a < 2^n$ there is $X_a \in \mathcal{X}$ coding a computation of C on a of the truth value of $\varphi(a)$. Getting a circuit (and computations) coded by a number seems to require new ideas.

The best currently known unprovability result is due to Pich [29, Corollary 6.2] and is conditional: a theory formalizing NC^1-reasoning does not prove almost everywhere superpolynomial lower bounds for SAT unless subexponential size formulas can approximate polynomial size circuits. Reaching S^1_2 seems to require new ideas.

References

[1] M. Ajtai. The complexity of the pigeonhole principle. Proceedings of the 29th Annual Symposium on the Foundations of Computer Science (FOCS'88), pp. 346-355, 1988. 1.3, 5.3

[2] A. Atserias, M. Müller. Partially definable forcing and bounded arithmetic. Archive for Mathematical Logic 54 (1): 1-33, 2015. 1.3

[3] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, P. Pudlák, A. Woods. Exponential lower bound for the pigeonhole principle (extended abstract). Proceedings of the ACM Symposium on Theory of Computing (STOC'92), ACM Press, pp. 200-220, 1992. 1.3, 5.3

[4] A. Beckmann, S. R. Buss. Improved witnessing and local improvement principles for second-order bounded arithmetic. ACM Transactions on Computational Logic 15 (1): Article 2, 2014. 3

[5] S. R. Buss. Bounded Arithmetic. Bibliopolis, Naples, 1986. 1, 1.1, 2.1

[6] J. Bydzovsky, M. Müller. Polynomial time ultrapowers and the consistency of circuit lower bounds. Archive for Mathematical Logic 59 (1): 127-147, 2020. 1.1

[7] J. Bydzovsky, J. Krajíček, I. C. Oliveira. Consistency of circuit lower bounds with bounded theories. Logical Methods in Computer Science 16 (2), 2020. 1.1

[8] M. Carmosino, V. Kabanets, A. Kolokolova, I. C. Oliveira. LEARN-uniform circuit lower bounds and provability in bounded arithmetic. Proceedings of the 62nd Annual Symposium on Foundations of Computer Science (FOCS'21), pp. 770-780, 2021. 1.1

[9] L. Chen, S. Hirahara, I. C. Oliveira, J. Pich, N. Rajgopal, R. Santhanam. Beyond natural proofs: hardness magnification and locality. Proceedings of the 11th Innovations in Theoretical Computer Science (ITCS'20), LIPIcs 151, pp. 70:1-70:48, 2020. 1.3

[10] S. A. Cook, J. Krajíček. Consequences of the Provability of $\mathsf{NP} \subseteq \mathsf{P/poly}$. Journal of Symbolic Logic 72 (4): 1353-1371, 2007. 1.1, 2

[11] S. A. Cook, P. Nguyen. Logical Foundations of Proof Complexity. ASL Perspectives in Logic, Cambridge University Press, 2010.

[12] M. L. Furst, J. B. Saxe, M. Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. Math. Syst. Theory 17(1): 13-27, 1984. Preliminary version in FOCS'81. 1

[13] P. Hájek, P. Pudlák. Metamathematics of First-Order Arithmetic. Perspectives in Mathematical Logic, Springer, 1998.

[14] R. Impagliazzo, V. Kabanets, A. Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. Journal of Computer and System Sciences 65 (4): 672-694, 2002. 1.2

[15] E. Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. Annals of Pure and Applied Logic 129: 1-37, 2004. 1

[16] E. Jeřábek. Weak pigeonhole principle, and randomized computation. PhD thesis, Faculty of Mathematics and Physics, Charles University, Prague, 2005. 1

[17] E. Jeřábek. Approximate counting in bounded arithmetic. Journal of Symbolic Logic 72 (3): 959-993, 2007. 1

[18] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. Information and Control 55 (1-3): 40-56, 1982. 1.1

[19] R. M. Karp, R. J. Lipton. Some connections between nonuniform and uniform complexity classes. Proceedings of the ACM Symposium on Theory of Computing (STOC'80), pp. 302-309, 1980. 1.1

[20] J. Krajíček. Exponentiation and second order bounded arithmetic. Annals of Pure and Applied Logic 48 (3): 261-276, 1990. 1.3

[21] J. Krajíček. No counter-example interpretation and interactive computation. In: Logic from Computer Science, ed. Y. N. Moschovakis, Mathematical Sciences Research Institute Publ. 21, Springer, pp. 287-293, 1992. 1.1

[22] J. Krajíček. Bounded Arithmetic, Propositional Logic, and Complexity Theory. Encyclopedia of Mathematics and Its Applications 60, Cambridge University Press, 1995. 1, 1.2, 1.3, 2, 2.1, 2.2, 3.1, 5.3

[23] J. Krajíček. Forcing with random variables and proof complexity, London Mathematical Society Lecture Note Series, No.382, Cambridge University Press, 2011. 1

[24] J. Krajíček, I. C. Oliveira. Unprovability of circuit upper bounds in Cook's theory PV. Logical Methods in Computer Science 13 (1), 2017. 1.1

[25] M. Müller. Typical forcing, NP search problems and an extension of a theorem of Riis. Annals of Pure and Applied Logic 172 (4): 102930, 2021. 41

[26] M. Müller, J. Pich. Feasibly constructive proofs of succinct weak circuit lower bounds. Annals of Pure and Applied Logic 172 (2): Article 102735, 2020. 1, 1.3

[27] C. Murray, R. Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'18), pp. 890–901, 2018. 1.2, 5.2, 5.2, 31

[28] I. C. Oliveira, R. Santhanam. Hardness magnification for natural problems. Proceedings of the 59th Symposium on Foundations of Computer Science (FOCS'18), pp. 65-76, 2018. 1.3

[29] J. Pich. Circuit lower bounds in bounded arithmetics. Annals of Pure and Applied Logic 166 (1): 29-45, 2015. 41

[30] J. Pich. Logical strength of complexity theory and a formalization of the PCP theorem in bounded arithmetic. Logical Methods in Computer Science 11(2), 2015. 1

[31] J. Pich, R. Santhanam. Strong co-nondeterministic lower bounds for NP cannot be proved feasibly. Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC'21). Association for Computing Machinery, pp. 223–233, 2021. 1

[32] A. A. Razborov. Bounded arithmetic and lower bounds in Boolean complexity. Feasible Mathematics II, pp. 344-386, 1995. 1

[33] A. A. Razborov. Unprovability of lower bounds on the circuit size in certain fragments of bounded arithmetic. Izvestiya of the Russian Academy of Science 59: 201-224, 1995. 41

[34] A. A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus. Annals of Mathematics 181 (2): 415-472, 2015. 1

[35] S. Riis. Finitization in bounded arithmetic. Basic Research in Computer Science, BRICS Report Series, RS-94-23, 1994. 41

[36] R. Santhanam, R. Williams. On uniformity and circuit lower bounds. Computational Complexity 23 (2): 177–205, 2014. 1.1

[37] G. Takeuti. Bounded arithmetic and truth definition. Annals of Pure and Applied Logic 39: 75-104, 1988. 1.3

[38] R. Williams. Improving exhaustive search implies superpolynomial lower bounds. SIAM Journal on Computing 42 (3): 1218-1244, 2013. 1.2

[39] R. Williams. Natural proofs versus derandomization. SIAM Journal on Computing 45 (2): 497-529, 2016. 1.2