Electronic Colloquium on Computational Complexity, Report No. 187 (2025)

On the Time Complexity of Feasible Proofs

Jiatu Li*

November 19, 2025

Abstract

Quantifying and understanding the complexity of mathematical proofs is a fundamental question in proof complexity. At the *qualitative* level, bounded arithmetic formalizes the notion of *feasible proofs*, where all functions implicit in proofs are from certain complexity classes. For instance, Cook's theory PV (STOC'75) captures proofs using only polynomial-time computable functions. Analogous to the relationship between algorithms and circuits, there is a tight connection between proofs in bounded arithmetic and propositional proofs via the *propositional translation*: If a statement is provable in a bounded theory (e.g. PV), then the family of propositional formulas formalizing the statement admits *polynomial-length* propositional proofs in its corresponding propositional proof system (e.g. Extended Frege).

The paper proposes a theory for the *quantitative* time complexity of arithmetic proofs. For any proof π in Cook's theory PV, its time complexity is a function of the input length of the variables in its conclusion, defined as the size of the propositional proofs obtained via a fixed propositional translation. In other words, the time complexity of arithmetic proofs is a uniform counterpart of proposition proof size, which is analogous to computational complexity theory, where the time complexity of algorithms is a uniform counterpart of circuit size.

Our main result is a *feasibility hierarchy theorem*, which shows that proofs with higher time complexity are strictly more powerful than those with lower time complexity. In other words, one cannot hope to generically reduce the time complexity of proving certain PV equations by writing longer and more complicated PV proofs. This theorem reveals the internal structure of the theory PV, suggesting the existence of a rich theory for the time complexity of proofs. Motivated by the feasibility hierarchy theorem, we propose a new proof complexity conjecture, which implies the unprovability of $\mathsf{NP} = \mathsf{coNP}$ in Cook's theory PV. This contributes to a recent line of work on the unprovability of complexity conjectures.

*Massachusetts Institute of Technology, Cambridge, MA, USA. jiatuli@mit.edu.

ISSN 1433-8092

Contents

1 Introduction

Understanding the complexity of mathematical proofs is a fundamental challenge in proof complexity. In the context of propositional logic, Cook and Reckhow [CR79] studied the *length of proofs* in propositional proof systems and its connection to the separation $\mathsf{NP} \neq \mathsf{coNP}$ in computational complexity theory. Since then, extensive research has established upper and lower bounds on proof length in various propositional proof systems (see, e.g., [Kra19, BN21] and references therein).

Similarly, researchers have also studied the complexity of *arithmetic proofs* (e.g. proofs in Peano Arithmetic and its fragments). A natural complexity measure is the *size* or *length* of the arithmetic proofs, and an early result is Gödel's speed-up theorem [Göd36] (see also [Bus94, Bus95a]).[1] The size or length of (first-order) arithmetic proofs is also discussed in several papers by Krajíček and Pudlák (see, e.g., [Pud86, KP88, Pud96]).[2]

Beyond proof length, another, and more widely used, approach to analyzing proof complexity originates from the tradition of *constructive mathematics* (see, e.g., [BPI22]). At a high level, the complexity of proofs is typically determined by the complexity of the *functions* they employ. This includes both explicitly defined function symbols and functions implicit in axioms and rules; for instance, using induction on a property φ implicitly requires a function to locate counterexamples.

Example 1.1 (functions implicit in arithmetic proofs)**.** Suppose that a proof uses the induction principle on a property φ, i.e., $\varphi(0) \to \forall n\ (\varphi(n) \to \varphi(n+1)) \to \forall n\ \varphi(n)$, which can be equivalently stated as

$$\forall n\ \exists n'\ (\varphi(0) \wedge \neg\varphi(n) \to (\varphi(n') \wedge \neg\varphi(n'+1))),$$

then the proof implicitly uses a function $n \mapsto n'$ satisfying that if $\varphi(0)$ and $\neg\varphi(n)$ are true, then $\varphi(n')$ is true and $\varphi(n'+1)$ is false.

In this regard, there are two main lines of work: the program of *reverse mathematics* (see, e.g., [Sim09, Sti20, DM22]) classifies proofs by the *computability* of functions, while the area of *bounded arithmetic* classifies them based on the *computational complexity* of functions. We focus on the latter approach.

Bounded arithmetic refers to a collection of fragments of Peano Arithmetic corresponding to functions in complexity classes (ranging from AC^0 to PSPACE, and beyond) based on the functions used in its proofs. Here, we highlight three representative theories (see [Kra95, CN10] for further examples).

- Parikh [Par71] introduced the first bounded theory known as $I\Delta_0$, which corresponds to the linear-time hierarchy (see, e.g., [Bus99]).
- Following Skolem's primitive recursive arithmetic [Sko23] and Cobham's machine-independent characterization of polynomial-time computation [Cob65], Cook [Coo75] introduced an equational theory called PV. This theory corresponds to FP, the class of polynomial-time computable functions.
- Independently, Buss [Bus86] defined theories S^i_2 and T^i_2 for $i \in \mathbb{N}$ corresponding to levels of the polynomial-time hierarchy, where the lowest level T^0_2 is equivalent to Cook's theory PV [Jeř06].

Notably, bounded theories are closely related to propositional proof systems through *propositional translations*. Cook [Coo75] proved that if a statement $\phi(x)$ is provable in PV, the formula formalizing $\forall x \in \{0,1\}^n\ \phi(x)$ admits a *polynomial-size* proof in Extended Frege EF, which can be efficiently generated from the PV proof π (see, e.g., [Kra95, Kra19]). Similar connections are known for Parikh's $I\Delta_0$ [PW85, Ajt83] and Buss's theories [KP90, KT90].

The missing corner: quantitative complexity analysis. There is an extensive line of work on super-polynomial separations between different propositional proof systems or their uniform counterpart, separations of different bounded theories (see Section 1.6). A central theme of this research is the classification

[1] In particular, Gödel proved that for any recursive function $h : \mathbb{N} \to \mathbb{N}$, there exists an infinite family of statements ϕ_k such that ϕ_k can be proved in k steps in second-order arithmetic but requires $h(k)$ steps to prove within first-order arithmetic.

[2] Friedman (unpublished) and Pudlák [Pud86] proved that the "finitistic consistency sentence", namely "there is no proof of inconsistency within size n", requires $n^{\Omega(1)}$ size. This can be viewed as an exponential size lower bound, as it requires $O(\log n)$ bits to describe the finitistic consistency sentence when n is encoded in binary.

of proof systems — both propositional proof systems and bounded theories — based on their *qualitative* strength. The significance of such a classification is evident: it aligns with the Cook-Reckhow program toward proving NP $\neq$ coNP [CR79] and guides the design of automated SAT solvers (see, e.g., [BN21]), which are widely used in practice.

However, despite extensive research in proof complexity, the *quantitative complexity* of proofs remains largely unexplored. For propositional proof systems, there has been little interest in improving the concrete exponent of polynomial-size proof complexity upper bounds (see, e.g., [CR79, Bus87, Bus15]). Moreover, no formal attempt has been made, to our knowledge, to define the notion of "time complexity" of proofs in bounded theories, such as Cook's theory PV, as a uniform counterpart of propositional proof size. This stands in sharp contrast to the field of algorithms and computational complexity, where studying the quantitative time complexity of classical problems across various computational models has been a central research focus for decades (see, e.g., [Knu97, CLRS22]), alongside the development of computational complexity theory (see, e.g., [Gol08, AB09, Wil18]).

1.1 Intuition of Time Complexity of Proofs

The main contribution of this paper is the formal definition of the *time complexity* of proofs in bounded theories, along with an investigation of its fundamental properties. We focus on Cook's equational theory PV [Coo75], which corresponds to polynomial-time computation, though our results are likely to extend to other bounded theories.

- (*Time Complexity of Proofs*). We define the time complexity of a PV proof π as **the length of the corresponding Extended Frege proof generated by Cook's propositional translation** [Coo75]. The time complexity measure satisfies desired closure properties, and it is *nontrivial* in that it does not directly correspond to proof length — PV proofs of the same length can vary significantly in time complexity.
- (*A Hierarchy Theorem*). The main technical result of the paper is the *feasibility hierarchy theorem*, which asserts that PV proofs with higher time complexity prove strictly more equations than those with lower time complexity.
- (*A New Proof Complexity Conjecture*). Inspired by the hierarchy theorem, we introduce a new proof complexity conjecture. The conjecture is an analogue of a circuit lower bound proved **using the time hierarchy theorem**. We show that the proof complexity conjecture implies that NP $=$ coNP is unprovable in PV_1, a major open problem in meta-mathematics of complexity theory.

1.1.1 What is the Time Complexity of Proofs?

In a nutshell, the **time complexity** of a PV proof π is defined as the **size of the Extended Frege proof** obtained from π via a **fixed and standard** propositional translation [Coo75]. For readers familiar with computational complexity theory, this is analogous to defining the time complexity of an algorithm by the size of the circuits obtained via a fixed and standard transformation from algorithm to circuits.

In (slightly) more detail, let π be a PV proof concluding an equation e and $\vec{x}$ denotes the variables in e, the time complexity of π is defined as a **function** $\mu_\pi^{\mathsf{TC}}(\vec{n})$ such that for any input length $\vec{n}$ for the variables $\vec{x}$, $\mu_\pi^{\mathsf{TC}}(\vec{n})$ is the size of the Extended Frege proof obtained from π via a fixed and standard propositional translation on the input length $\vec{n}$. Throughout this paper, we denote the EF proof by $[\pi]^{\vec{n}}_{\mathsf{Cook}}$, and the conclusion of the Extended Frege proof, denoted by $[e]^{\vec{n}}_{\mathsf{Cook}}$, formalizes the statement $\forall \vec{x} \in \{0,1\}^{\vec{n}}\ e(\vec{x})$.

We make three remarks about the definition of time complexity of proofs.

First, the time complexity of PV proofs is a different measure from the size of the proof. The former measure is similar to the time complexity of algorithms, while the latter one is similar to the size of programs. The definition of time complexity emphasizes the view that arithmetic proofs are **uniform propositional proofs**, and measures the complexity of arithmetic proofs via the size of corresponding

propositional proofs. We find this approach natural and fundamental, as it provides a uniform counterpart of the established theory of propositional proof complexity.

Second, the notion of time complexity is meaningful only with respect to some fixed standard propositional translation. One could define a strange propositional translation that trivializes the notion; we will give an example below. Fix a standard propositional translation $[\cdot]^{\vec{n}}_{\mathsf{Cook}'}$ and let

$$\ell(\vec{n}, k) := \max_{|\pi|}\{|[\pi]^{\vec{n}}_{\mathsf{Cook}'}| \mid \pi \text{ is a PV proof of size } k\}, \tag{1.1}$$

that is, the size of the largest possible propositional translation of size-k PV proofs. We can define a new propositional translation $\langle\cdot\rangle$ as follows: Given any PV proof π and input lengths $\vec{n}$, the corresponding propositional proof is obtained by obtaining $[\pi]^{\vec{n}}_{\mathsf{Cook}'}$ and then padding meaningless equations so that the proof is of size $\ell(\vec{n}, k)$. The time complexity of PV proofs with respect to this new propositional translation is essentially a more complex way of expressing the size of proofs.

Nevertheless, we argue that this is also the case for computational complexity: The notion of time complexity only makes sense if we choose a fixed and standard way to simulate the machines. If one were to define the time complexity of an algorithm as its running time on a machine that stalls unpredictably, there would be no meaningful complexity theory.

Finally, the time complexity of PV proofs is *not* intended to capture the *minimum* size of the Extended Frege proofs — the fixed and standard propositional translation is not necessarily an optimal one. This is, to some extent, a concession to our current lack of knowledge in proof complexity, as it is consistent with our knowledge that every sequence of tautology $\{\phi_n\}_{n\in\mathbb{N}}$ can be proved by $O(|\phi_n|^3)$-size Extended Frege proofs [Kra95, Chapter 13]. Similarly, the time complexity of algorithms is not intended to capture the minimum circuit size to implement the algorithm — it is consistent with our knowledge that $\mathsf{DTIME}[2^n] \subseteq \mathsf{SIZE}[O(n)]$ [LY22, FGHK23].

1.1.2 Which Propositional Translation to Choose?

As noted, the *choice of propositional translation* is crucial in the definition of time complexity of proofs. For instance, the definition would be meaningless if the time complexity became a mere alias for the proof's length (see Equation (1.1)).

In this paper, we define a particular propositional translation that we argue is best suited for the study of time complexity of proofs. At a high level, it satisfies the following good properties:

- (*Computational Efficiency*). Given a proof π and input lengths $n_1, \ldots, n_k$ in binary, the size of its propositional translation $[\pi]^{n_1,\ldots,n_k}_{\mathsf{Cook}'}$ can be computed in time $f(|\pi|) \cdot \mathsf{polylog}(n_1, \ldots, n_k)$ for some function f — there exists an FPT algorithm for computing the time complexity of proofs. When $n_1, \ldots, n_k \gg |\pi|$, this is much more efficient than generating $[\pi]^{n_1,\ldots,n_k}_{\mathsf{Cook}'}$. This suggests that we can *analyze* time complexity without generating the propositional proofs.

- (*No Obvious Asymptotic Redundancy*). It is not hard to imagine that appropriate padding is necessary for the computational efficiency of the time complexity. Nevertheless, we argue that the padding does not seem to incur an obvious asymptotic overhead.[3]

- (*Nontrivial*). The propositional translation does not make the time complexity an alias of PV proof length: For every large $k \in \mathbb{N}$, PV proofs of length k could have time complexity as large as $n^{2^{\Omega(k)}}$, or as small as $O(k \cdot n)$. In other words, we are not padding too much (as in Equation (1.1)) that trivializes the time complexity of proofs.

- (*Closure Properties*). It satisfies desired closure properties. For instance, if $\alpha \to \beta$ can be proved in time t_1 and α can be proved in time t_2, β can be proved in time $O(t_1 + t_2)$.

[3] Again, it is impossible to argue that there is no overhead at all, as it is consistent with our knowledge that every tautology ϕ can be proved by $O(|\phi|^3)$-size Extended Frege proofs.

The last three bullets are arguably desired properties, but at this point, it is not clear why the computational efficiency of time complexity is important. Indeed, the computational efficiency, along with a few other properties, is crucial for the proof of the *hierarchy theorem*. That is a property that we expect from any complexity measure.

1.1.3 How to Fairly Compare the Time Complexity of Statements?

In addition to the complexity of a proof (an analogy of runtime of algorithms), it is important to define and study the complexity of *proving a statement* (an analogy of time complexity of a *language*). Naturally, we can define that a PV equation $e(\vec{x})$ is provable in time $t(\vec{n})$ if there is a PV proof π concluding $e(\vec{x})$ that has time complexity $\mu_\pi(\vec{n}) \leq t(\vec{n})$.

We argue that this definition is unsatisfactory, as it cannot be used to fairly compare the complexity of proving two different statements. This is because the time complexity of PV proofs is a function of the **input length** rather than the **description length** of the statements — if the description length of the statement is long, one cannot expect a shorter proof, as the statement itself is part of the proof.

Example 1.2. For a concrete example, we can consider the following two statements e_1 and e_2:

- e_1 states that $M_1(x) = M_1(x)$, where M_1 is a fixed algorithm running in time n^{1000}.
- e_2 states that $M_2(x) = M_2(x)$, where M_2 is a fixed algorithm running in time n.

These two statements are equally trivial to prove as any string is equal to itself. However, it is impossible to prove e_1 with time complexity smaller than n^{1000} (whereas e_2 can be proved in, e.g., n^2 time), as the equation e_1 will be the last line of the proof, and its propositional translation has size at least n^{1000}.

Note that for the same reason, it is common to measure proof complexity as a function of the *formula size* rather than the *number of variables* in the formula in the context of propositional proof complexity.

This calls for the definition of time complexities of *describing* PV equations as a basis of comparing time complexities of *proving* different equations. For a PV equation e, we use the function $\beta_e^{\mathsf{TC}}(\vec{n})$ to denote the time complexity to merely write down the equation e as conclusion. That is, $\beta_e^{\mathsf{TC}}(\vec{n})$ denotes the complexity overhead **necessary for any proof concluding** e.

Consequently, we can define the time complexity of *proving* e as follows: It is said to admit a time-μ proof if there is a proof π such that

$$\mu_\pi^{\mathsf{TC}}(\vec{n}) \leq \mu(\beta_e^{\mathsf{TC}}(\vec{n})).$$

We will use $\mu(\beta)$ to denote the time complexity of proving an equation — the variable β denotes $\beta_e^{\mathsf{TC}}(\vec{n})$, the time complexity of describing the equation e. For instance, both equations e_1 and e_2 in Example 1.2 will admit $\mu(\beta) = O(\beta)$ time (i.e. linear-time) PV proofs.

1.1.4 What is $\beta_e^{\mathsf{TC}}(\vec{n})$? Is It Just the Size of $[e]^{\vec{n}}_{\mathsf{Cook}}$?

At first glance, one might assume that $\beta_e(\vec{n})$, the time complexity of describing the equation e, is exactly the size of its propositional translation $[e]^{\vec{n}}_{\mathsf{Cook}}$.

There is no doubt that $\beta_e^{\mathsf{TC}}(\vec{n})$ is at least the size of its propositional translation $[e]^{\vec{n}}_{\mathsf{Cook}}$. Recall that for any PV proof π of e, its propositional translation, $[\pi]^{\vec{n}}_{\mathsf{Cook}}$, is an Extended Frege proof that concludes $[e]^{\vec{n}}_{\mathsf{Cook}}$. As we have defined $\mu_\pi(\vec{n})$ as the size of $[\pi]^{\vec{n}}_{\mathsf{Cook}}$, the size of its conclusion $[e]^{\vec{n}}_{\mathsf{Cook}}$ is a part of the *unavoidable time overhead* for any proof concluding e.

Let $\beta_e^+(\vec{n})$ be the size of $[e]^{\vec{n}}_{\mathsf{Cook}}$. It turns out that it is necessary to define $\beta_e^{\mathsf{TC}}(\vec{n})$ as the summation of $\beta_e^+(\vec{n})$ and **another term** that, at a high level, describes the time complexity of the "time complexity analysis" of functions used in e. It accounts for the time complexity of proofs that are necessary to **define** the functions in e. For readers familiar with the theory PV, recall that only **provably** polynomial-time functions are allowed, and such **proofs** will necessarily present before writing down the equation e.

The necessity is for both conceptual and technical reasons; in particular, it allows us to prove a "feasible translation theorem" (see Section 6.2) that is used in the proof of the hierarchy theorem. Because this is quite subtle and related to the exact formulation of PV, we postpone the detailed discussion to Section 1.3, Section 2 and technical sections.

1.1.5 What is the Hierarchy Theorem, and Why is it Nontrivial?

Recall that the time hierarchy theorem [HS65, HS66] shows that for every $d \in \mathbb{N}$, there is a language L_d that is decidable in time $O(n^{d+1})$, and is not decidable in time $O(n^d)$. Our feasibility hierarchy theorem is an analogy of it: For every $d \in \mathbb{N}$, there is a PV equation e_d that is provable in time $\beta^{O(d)}$, but is not provable in time $O(\beta^d)$. This means that PV-provable equations form a non-collapsing infinite hierarchy based on the time complexity of their proofs.

We note that the hierarchy theorem is nontrivial as the lower bound states that e_d is not provable within $O(\beta^d)$ time, regardless of the length of the proof. This shows that one cannot hope for generically speeding up arithmetic proofs (in terms of time complexity) by writing more sophisticated proofs, just as one cannot hope for generically speeding up algorithms by writing more sophisticated programs.

1.2 Technical Motivations of the Work

The primary goal of this paper is to develop a theory of the *quantitative* (time) complexity of arithmetic proofs. Beyond its intrinsic philosophical interest, this work is motivated by two recent lines of research in theoretical computer science.

1.2.1 Meta-mathematics of Complexity Theory

Bounded theories such as PV [Coo75] and S^1_2 [Bus86] formalize a substantial portion of results in algorithms, combinatorics, and complexity theory (see Section 1.6 for related work). Therefore, the *unprovability* of complexity conjectures (e.g. P vs NP) in bounded theories provides a formal barrier. Indeed, *unconditional* unprovability results have been known for circuit upper bounds [CKKO21, ABM23] and strong complexity lower bounds [PS21, LO23].

The most prominent technique in proving strong unprovability results is the *witnessing theorem* (see, e.g., [Oli25]), and other techniques either apply to only weak theories (see, e.g., [Kra11a, ABM23]) or are only known for specific sentences (see [KP89]). However, an inherent limitation of the witnessing theorems is it does not fully explored the proof complexity structure of the theories; indeed, witnessing theorems usually hold even for the true universal theory, which barely has any structure (see, e.g., [LO23]). As Krajíček [Kra24] mentions, relying solely on witnessing theorems is insufficient because they do not change even when the true universal theory is added:

> *"... we need to leave theory T_{PV}[4] aside and work with theories* PV *or* S^1_2*. This implies that an argument cannot rely just on witnessing theorems as they do not change if T_{PV} is added."*

In particular, it is unlikely that the witnessing theorem is sufficient to prove the unconditional unprovability of any statement formalized as a universal sentence, e.g., $\mathsf{P} = \mathsf{NP}$.[5] This calls for the exploration of new techniques from logic that works on strong theories such as PV.

We suggest that the study of time complexity of PV proofs may lead to new techniques for proving strong unprovability results. As a proof-of-concept, we introduce a new conjecture under which it is easy to show the unprovability of $\mathsf{NP} = \mathsf{coNP}$. This conjecture can be seen as a proof-complexity analogue of $\mathsf{P} \not\subseteq \mathsf{P}\text{-uniform } \mathsf{SIZE}[n^k]$, which, intriguingly, is proved **using the time hierarchy theorem** [SW14].

[4] T_{PV} stands for the true universal theory in the language of PV.

[5] It is well-known that "the algorithm A solves search-SAT" can be formalized as the following universal sentence: For any propositional formula φ that has a satisfying assignment x, $A(\varphi)$ outputs a satisfying assignment of φ. Note that this is without loss of generality, as the search-to-decision reduction of SAT can be formalized in PV.

1.2.2 Cryptography via Feasible Arithmetic Proofs

Recently, bounded arithmetic and its connection to propositional proof systems have been used as a crucial technical component in the construction of advanced cryptographic primitives, such as indistinguishability obfuscation [JJ22, MDS25, JJMP25] and SNARGs [JKLV24, JKLM25]. The efficiency of cryptographic primitives is closely tied to the size of certain propositional proofs (see, e.g., [MDS25, Theorem 1.1]). Thus, improving the concrete size of these proofs could lead to more efficient cryptographic systems.

One might suspect that cryptographic constructions are only concerned with the size of the *propositional proofs*, and thus there is no need to study the complexity of *arithmetic proofs*. Perhaps surprisingly, the notion of the time complexity of arithmetic proofs naturally popped up in recent research on building *succinct* cryptography primitives, most importantly, $i\mathcal{O}$ for Turing machines [JJ22, JJMP25] and SNARGs with short common reference strings (CRS) [JKLM25].

Before further extending on this line of work, we need to review some basic definitions in cryptography. A randomized algorithm $i\mathcal{O}(1^\lambda, \cdot)$ is said to be an *indistinguishability obfuscation for circuits* if it satisfies the following conditions:

- (*Correctness*). For all circuit C of size $\mathsf{poly}(\lambda)$, one can evaluate $C(x)$ given $i\mathcal{O}(1^\lambda, C)$ and x.
- (*Security*). $i\mathcal{O}(1^\lambda, C_1)$ and $i\mathcal{O}(1^\lambda, C_2)$ are computationally indistinguishable against $\mathsf{poly}(\lambda)$ size adversaries for functionally equivalent circuits C_1, C_2 of size $\mathsf{poly}(\lambda)$.

One of the main results of [JJ22] is that, if we relax the security condition to hold only on circuits that are provably equivalent in polynomial-sized EF, there is a construction of $i\mathcal{O}$ from *polynomial* security assumptions, whereas other constructions (see, e.g., [JLS24]) need *sub-exponential* security assumptions.

A main disadvantage of $i\mathcal{O}$ for circuits is that it only allows evaluation on a particular input length, and the size of the obfuscated circuit grows with the input length. Motivated by both practical and theoretical applications of $i\mathcal{O}$ (e.g. SNARG [SW21]), a major open problem is to obfuscate a *program* that supports evaluation on any input lengths up to $\mathsf{poly}(\lambda)$. This allows to make the size of the obfuscated program a *fixed polynomial* in λ — rather than a polynomial in the input length, an unbounded polynomial in λ.

Extending their construction of $i\mathcal{O}$ for circuits, Jain and Jin [JJ22] provided a construction of $i\mathcal{O}$ for Turing machines that are PV provably equivalent from standard assumptions. A closer look at the construction reveals that both the size and the **time complexity** of the PV proof matter in the construction:

- The PV proof is hard-coded in the obfuscation, and therefore the *size of the obfuscated program* grows with the size of the PV proof.
- The *evaluation time complexity* of obfuscated programs grows with the size of the propositional translation of the PV proof — or, as we defined, the **time complexity** of the PV proof.

Note that both the size and evaluation time complexity of obfuscated programs are crucial for downstream applications, see, e.g., [MDS25, JJMP25] for detailed exposition. In particular, the evaluation time complexity in [MDS25] is *(quasi-)linear* in the time complexity of the PV proofs — the time complexity of PV proofs becomes the bottleneck for achieving obfuscations with efficient evaluation.

Therefore, it will be instructive to study **time-bounded** PV **proofs** for understanding the capability and limitation of the Jain-Jin framework. For instance, one may wonder whether any PV provable equation can be proved in a fixed polynomial time complexity, and thus the evaluation time complexity of the Jain-Jin $i\mathcal{O}$ for Turing machines will always be a fixed polynomial. Specifically, suppose there is a time-t size-s PV proof π, is it always possible to find a time-t' size-s' PV proof π' such that $t' \ll t$, with the overhead that $s' > s$? Our feasibility hierarchy theorem shows that this is impossible, just as one cannot simulate any polynomial time algorithm in a fixed polynomial-time by writing a more sophisticated program.

1.3 Our Results, in More Detail

We focus on the equational theory PV for simplicity in propositional translations. All statements in PV are of the form $t_1(\vec{x}) = t_2(\vec{x})$, where the open variables $\vec{x}$ are considered to be universally quantified (see Section 3 and Appendix A for more details).

Definition of the complexity measures. Let $e : t_1(\vec{x}) = t_2(\vec{x})$ be an PV equation and π be a PV proof concluding e. Recall that by Cook's propositional translation [Coo75], for any input length $\vec{n} = (n_1, \ldots, n_k)$ of the variables $\vec{x} = (x_1, \ldots, x_k)$ in the conclusion e, we can generate a proof $[\pi]^{\vec{n}}_{\mathsf{Cook}}$ of length $\mathsf{poly}(n_1, \ldots, n_k)$ in Extended Frege concluding a propositional formula $[e]^{\vec{n}}_{\mathsf{Cook}}$ formalizing:

$$\forall x_1 \in \{0,1\}^{n_1}\ \forall x_2 \in \{0,1\}^{n_2}\ \ldots \forall x_k \in \{0,1\}^{n_k}\ t_1(\vec{x}) = t_2(\vec{x}).$$

We show that there is a standard propositional translation $[\cdot]^{\vec{n}}_{\mathsf{PV}}$ of PV equations and proofs such that the following holds:

Theorem 1.1 (informal)**.** *Let π be a* PV *proof and e be a* PV *equation. There are complexity measures $\mu^{\mathsf{TC}}_{\pi}(\vec{n})$, $\beta^{+}_{e}(\vec{n})$, $\beta^{\mathsf{TC}}_{e}(\vec{n})$ such that the following properties hold:*

- *(Efficiently computable). There are fixed-parameter tractable (FPT) algorithms[6] computing $\mu^{\mathsf{TC}}_{\pi}(\vec{n})$, $\beta^{+}_{e}(\vec{n})$, and $\beta^{\mathsf{TC}}_{e}(\vec{n})$ given π or e and the binary encoding of the input length $\vec{n}$, where the length of π and e are parameters (see Section 5).*
- *(Size). $|[e]^{\vec{n}}_{\mathsf{Cook}}| = \beta^{+}_{e}(\vec{n}) \leq \beta^{\mathsf{TC}}_{e}(\vec{n})$ and $|[\pi]^{\vec{n}}_{\mathsf{Cook}}| = \mu^{\mathsf{TC}}_{\pi}(\vec{n})$ (see Theorem 4.3).*
- *(No Obvious Asymptotic Redundancy). Unnecessary padding is introduced in the propositional translations $[\cdot]_{\mathsf{PV}}$ for proofs and equations to ensure efficient computation of complexity measures. Nevertheless, the size of padding is linear in the size of the propositional translation without padding (see Section 4.5).*
- *(Nontrivial). For $k \geq 1$, there are* PV *proofs π_1, π_2 of length $O(k)$ such that*

$$\mu^{\mathsf{TC}}_{\pi_1}(n) = O(k \cdot n) \quad \textit{and} \quad \mu^{\mathsf{TC}}_{\pi_2}(n) = \Omega(n^{2^k}).$$

  That is, proof length does not directly determine time complexity (see Section 4.6).

The meaning of $\beta^{+}_{e}(\vec{n})$ and $\mu^{\mathsf{TC}}_{\pi}(\vec{n})$ are clearly stated by the theorem: $\beta^{+}_{e}(\vec{n})$ is the size of $[e]^{\vec{n}}_{\mathsf{Cook}}$ (the propositional translation of the equation), while $\mu^{\mathsf{TC}}_{\pi}(\vec{n})$ is the size of $[\pi]^{\vec{n}}_{\mathsf{Cook}}$ (the propositional translation of the proof). We define the function $\mu^{\mathsf{TC}}_{\pi}(\vec{n})$ as the time complexity of the proof π.

Recall that the complexity measure $\beta^{\mathsf{TC}}_{e}(\vec{n})$ denotes the **unavoidable time complexity overhead** for any PV proof of e (see Section 4.6), and is used to fairly compare the time complexity of proving different equations. As mentioned in Section 1.1.4 and the second bullet of Theorem 1.1, $\beta^{\mathsf{TC}}_{e}(\vec{n}) \geq \beta^{+}_{e}(\vec{n})$. In this paper, we call $\beta^{+}_{e}(\vec{n})$ the propositional complexity of e, and $\beta^{\mathsf{TC}}_{e}(\vec{n})$ the time complexity of e. The exact definition of $\beta^{\mathsf{TC}}_{e}(\vec{n})$ is deferred to Section 2 due to space limits.

Remark 1.1. It is worth mentioning that when the length of π and e are sufficiently small, the FPT algorithms allow us to compute the complexity measures for π and e in $\mathsf{polylog}(\vec{n})$ time, which is significantly more efficient than actually generating the propositional translation of e and π on input length $\vec{n}$. The computational complexity difference between computing the time complexity measures and generating the propositional translation of formulas and proofs is a key technical observation to prove our main theorem.

1.3.1 Feasible Deduction Theorem

As the definition of the time complexity of proofs is quite technical, it is important to develop technical tools that simplify the time complexity analysis of proofs.

We prove the feasible deduction theorem — analogous to computational composition, applying Modus Ponens incurs minimal time complexity overhead. Formally:

[6] Fix a search problem and a function $k(\cdot)$ such that for each input x, $k = k(x)$ is said to be the *parameter* of the input x. An algorithm of the search problem is said to be an FPT algorithm if there is a function f such that for any input x, the running time of the algorithm is at most $f(k(x)) \cdot \mathsf{poly}(|x|)$; see, e.g., [CFK+15].

Theorem 1.2 (informal, see Theorem 6.2)**.** *There is a constant d such that the following holds. Let $e_1(\vec{x}), e_2(\vec{x})$ be* PV *equations, and $e_1(\vec{x}) \Rightarrow e_2(\vec{x})$ be an equation formalizing that "for any $\vec{x}$, $e_1(\vec{x})$ is true implies $e_2(\vec{x})$ is true". Suppose that π_1 is a* PV *proof of $e_1(\vec{x})$ and $\pi_\Rightarrow$ is a* PV *proof of $e_1(\vec{x}) \Rightarrow e_2(\vec{x})$, then there is a* PV *proof π_2 of $e_2(\vec{x})$ such that*

$$\mu^{\mathsf{TC}}_{\pi_2}(\vec{n}) \leq d \cdot (\mu^{\mathsf{TC}}_{\pi_1}(\vec{n}) + \mu^{\mathsf{TC}}_{\pi_\Rightarrow}(\vec{n})).$$

1.3.2 Feasible Proof Generation Theorem

Theorem 1.1 shows that the time complexity of a proof tightly matches the size of its propositional translation. The feasible proof generation theorem further states that for any proof π, the time complexity of the *correctness proof* of its propositional translation matches the its time complexity.

Theorem 1.3 (informal, see Theorem 6.4)**.** *There is a constant $d \geq 1$ such that the following holds. Let π be a* PV *proof concluding $e(\vec{x})$. Then there is a* PV *proof π_{gen} of the following statement:*

> *For all $\vec{n}$, $[\pi]^{\vec{n}}_{\mathsf{Cook}}$ (i.e. the propositional translation of π on the input length $\vec{n}$) is a valid Extended Frege proof concluding $[e]^{\vec{n}}_{\mathsf{Cook}}$ (i.e. the propositional translation of e on the input length $\vec{n}$).*

Moreover, $\mu^{\mathsf{TC}}_{\pi_{\mathsf{gen}}}(\vec{n}) = O((\mu^{\mathsf{TC}}_{\pi}(\vec{n}))^d)$.

Here, $O(\cdot)$ hides a constant that depends on π, while d is a fixed constant. This theorem essentially shows that the *correctness proof* of the propositional translation [Coo75] can be formalized in PV, and the correctness proof incurs only a fixed polynomial-time complexity overhead.

Remark 1.2. It might be instructive to think of the feasible proof generation theorem as a proof complexity analogue of the following result in computational complexity: For any Turing machine $A(x)$ that runs in time $T(n)$, there is a Turing machine $G(1^n)$ that runs in time $O((T(n))^2)$ and outputs the description of the circuit $C : \{0,1\}^n \to \{0,1\}$ such that $A(x) = C(x)$ for every $x \in \{0,1\}^n$ [AB09, Section 6].

1.3.3 Feasible Translation Theorem

Recall that at a high level, $\beta^{\mathsf{TC}}_e(\vec{n})$ captures the *unavoidable time complexity overhead* for any PV proof of e. The feasible translation theorem shows that it indeed corresponds to the time complexity of the correctness proof of the feasible translation:

Theorem 1.4 (informal, see Theorem 6.3)**.** *There is a constant $d \geq 1$ such that the following holds. Let $e(\vec{x})$ be an equation in* PV*. Then there is a* PV *proof π_{tr} of the following statement:*

$$\forall \vec{x}\ e(\vec{x}) \text{ if and only if for every } \vec{n},\ [e]^{\vec{n}}_{\mathsf{Cook}} \text{ is a tautology.}$$

Moreover, $\mu^{\mathsf{TC}}_{\pi_{\mathsf{tr}}}(\vec{n}) = O((\beta^{\mathsf{TC}}_e(\vec{n}))^d)$.

Remark 1.3. Note that in Theorems 1.2 to 1.4, we only analyze the time complexity overhead of particular formalizations of the statements. For instance, the equation $e_1(\vec{x}) \Rightarrow e_2(\vec{x})$ in Theorem 1.2 follows from a construction in [CU93]. Similar results are expected to hold for other reasonable formalizations, possibly with a slightly greater time complexity overhead.

1.3.4 Uniform Proof Complexity Classes

Similar to the complexity classes $\mathsf{DTIME}[t(n)]$ and $\mathsf{NTIME}[t(n)]$, we may define proof complexity classes as sets of equations that can be proved with bounded time complexity.

Definition 1.4 (informal, see Definition 7.3)**.** Let $s : \mathbb{N} \to \mathbb{N}$ be a function. $\mathsf{EqTIME}[s]$ is defined as the set of PV equations $e(\vec{x})$ that have proofs π such that $\mu^{\mathsf{TC}}_{\pi}(\vec{n}) = O(s(\beta^{\mathsf{TC}}_e(\vec{n})))$.

The proof complexity classes are defined based on the time complexity of proofs as a function of the time complexity of equations rather than the input length of variables. This is similar to the proof complexity in propositional proof systems: The proof length is considered as a function of the *formula size* rather than the number of variables, as formula size denotes the complexity to *describe* the question. As mentioned in Section 1.1.4, we use *time complexity* $\beta_e^{\mathsf{TC}}(\vec{n})$ of an equation as the variable of $s(\beta)$, rather than its *propositional complexity* $\beta_e^{+}(\vec{n})$, because complexity analysis of functions in e should also be considered as a part of the description of the question.

1.4 Main Theorem: Feasibility Hierarchy Theorem

We are now ready to state the feasibility hierarchy theorem:

Theorem 1.5 (Corollary 7.8)**.** *There is $c \geq 1$ such that for every $d \geq 1$, $\mathsf{EqTIME}[\beta^d] \subsetneq \mathsf{EqTIME}[\beta^{cd}]$.*

In other words, PV proofs with higher time complexity can prove strictly more equations than those with lower time complexity: there are PV equations that can be proved with time complexity $O(\beta^{cd})$, but do not have PV proofs with time complexity $O(\beta^d)$, regardless of the length of proofs. There is an infinite hierarchy of PV-provable equations that does not collapse, based on the time efficiency of their proofs.

Explicit witness: self-referential sentences. Indeed, the separation in Theorem 1.5 for $d \geq 1$ is achieved by an explicit equation e_d, which is a modification of the self-referential sentence used in the proof of Gödel's (first) incompleteness theorem [Göd31] (see also [Coo75] for Gödel's incompleteness theorem for PV). At a high level, e_d asserts that itself does not admit a proof with time complexity at most $(\beta_{e_d}^{\mathsf{TC}}(\vec{n}))^{d+1}$.

Analogous to Gödel's second incompleteness theorem, we prove that another equation Con_d, called the bounded feasibility consistency sentence, is also a witness of Theorem 1.5 for sufficiently large d. At a high level, Con_d states that there is no PV proof of time complexity β^d that derives a contradiction.

It is worth noting that both the self-referential sentence e_d and the bounded feasibility consistency sentence Con_d have *fixed polynomial* time complexity in their input length, i.e., the exponent is independent of d. That is, both statements have time complexities asymptotically independent of d, while the time complexities of their proofs increase rapidly in d.

Comparison to time hierarchy theorem. The proof of Theorem 1.5 is involved, and is deferred to Section 7; see also Section 2.4 for an overview of the proof. To get a glimpse of our technique, we compare Theorem 1.5 with the time hierarchy theorem [HS65, HS66].

Recall that the proof of the time hierarchy theorem has two main components: a time-bounded adaptation of Turing's diagonalization argument [Tur37] and an efficient universal Turing machine. The proof of Theorem 1.5 follows a similar methodology using tools from logic:

- (*Lower bound*). The lower bound adapts Gödel's incompleteness theorem [Göd31, Coo75] to the time-bounded setting, as discussed above.
- (*Upper bound*). The upper bound is proved by formalizing Cook's consistency proof of PV [Coo75] in PV. This may seem paradoxical, as Gödel's second incompleteness theorem states that PV cannot prove its own consistency (see [Coo75, Theorem 4.4]). We note, however, that following Cook's consistency proof of PV, the bounded feasibility consistency Con_d of PV is provable in PV, and this proof is efficient in the sense that it incurs only a fixed polynomial time complexity overhead.

Analogously to our upper bound, the acceptance problem for unrestricted Turing machines is undecidable, whereas for Turing machines with an explicit time bound, it becomes decidable with a fixed polynomial time overhead. Note that there are additional technical issues in formalizing the self-referential statements for the upper bound, which we will explain in Section 2.4.

1.5 Uniformity vs Nonuniformity in Proof Complexity

Inspired by the feasibility hierarchy theorem, we propose two proof complexity conjectures that formalize quantitative separations between uniform (i.e. bounded arithmetic) and non-uniform proofs (i.e. propositional proof systems).

1.5.1 Fixed Polynomial Lower Bound against Extended Frege

Our first conjecture is a fixed polynomial lower bound against Extended Frege for PV:

Conjecture 1.1 (informal, see Conjecture 8.1)**.** For every $k \geq 1$, there is a family of formulas $\{\varphi_n\}_{n\in\mathbb{N}}$ that can be generated by a polynomial-time algorithm $A(1^n)$ such that φ_n is a PV-provable tautology, and for every constant $c \geq 1$, φ_n requires Extended Frege proofs of size at least $c \cdot |\varphi_n|^k$ for infinitely many n.

Conjecture 1.1 is stronger than Cook's conjecture that Extended Frege is not p-bounded[7]: it states that Extended Frege is not p-bounded, and that for every fixed $k \geq 1$, there is an explicit PV-provable sentence φ_n witnessing that PV is not $O(|\varphi_n|^k)$-size bounded. In a sense, it can be viewed as a proof complexity analog of the conjectured circuit lower bound $\mathsf{P} \not\subseteq \mathsf{SIZE}[n^k]$ for any $k \geq 1$.

Although Conjecture 1.1 is not directly related to the theory of time complexity for PV, we argue that the feasibility hierarchy theorem provides evidence for this conjecture. Suppose that Conjecture 1.1 is false. Then there is a constant k such that PV proofs with arbitrarily high polynomial time complexity are as strong as $O(|\varphi_n|^k)$-size Extended Frege proofs. This will be an unexpected collapse in proof complexity.

In particular, we further conjecture that for any $k \geq 1$, there is a $d \gg k$ such that the propositional translation $\{\varphi_n\}_{n\in\mathbb{N}}$ of the witness e_d (or Con_d) of the feasibility hierarchy theorem requires $\omega(|\varphi_n|^k)$-size Extended Frege proofs.

1.5.2 Fixed Polynomial Lower Bound againt "Uniform EF"

Since Conjecture 1.1 is stronger than the longstanding open problem of proving that Extended Frege is not p-bounded, it may not be within reach of current techniques. To address this, we introduce a weaker conjecture that establishes a lower bound against "uniform" Extended Frege proofs.

Conjecture 1.2 (informal, see Conjecture 8.2)**.** For every $k \geq 1$, there is a family $\{\varphi_n\}_{n\in\mathbb{N}}$ that can be generated by a polynomial-time algorithm $A(1^n)$ such that the following holds.

- PV proves: for every $n \in \mathbb{N}$, φ_n is a tautology;
- For every $c \geq 1$, PV cannot prove: for every $n \in \mathbb{N}$, φ_n admits an Extended Frege proof of size at most $c \cdot |\varphi_n|^k$.

This conjecture is weaker than Conjecture 1.1 as it only asserts that φ_n does not PV-provably admit short Extended Frege proofs. It allows φ_n to have short EF proofs that cannot be generated efficiently by polynomial-time algorithms. Alternatively, such proof generation algorithms may exist, but their correctness proof cannot be formalized in PV. Similar to Conjecture 1.1, we further conjecture that explicit witnesses of the feasibility hierarchy theorems are also witnesses of Conjecture 1.2.

1.5.3 Application: Unprovability of $\mathsf{NP} = \mathsf{coNP}$

Beyond its intrinsic interest, we show that proving Conjecture 1.2 yields a breakthrough unprovability result in the meta-mathematics of complexity theory:

Theorem 1.6 (informal, see Theorem 8.2)**.** *Under Conjecture 1.2,* PV_1 *cannot prove* $\mathsf{NP} = \mathsf{coNP}$.

[7] A Cook-Reckhow propositional proof system $V(\phi, \pi)$ is said to be p-bounded if there is a constant k such that for every tautology ϕ, there is a proof π of size $O(|\phi|^k)$ such that $V(\phi, \pi) = 1$; see, e.g., [CR79, Kra19].

We note that Theorem 1.6 is robust to changes to the exact formalization in PV (see Section 8 for more details). Moreover, since $\mathsf{P} = \mathsf{NP}$ implies $\mathsf{NP} = \mathsf{coNP}$ under any reasonable formalization, it follows that $\mathsf{P} = \mathsf{NP}$ is also unprovable in PV under Conjecture 1.2. In other words, the complexity conjecture $\mathsf{NP} \neq \mathsf{coNP}$ is consistent with PV — there is a (possibly nonstandard) model of PV in which $\mathsf{NP} \neq \mathsf{coNP}$.

Theorem 1.6 builds on the main theorem of [Coo75] (see also Theorem 3.3), which essentially follows from the propositional translation of PV.

This result contributes to a line of recent work on the unprovability of complexity upper bounds [KO17, BM20, BKO20, CKKO21, ABM23]. It is worth noting that existing unconditional unprovability results focus on circuit upper bounds such as $\mathsf{P} \subseteq \mathsf{SIZE}[n^k]$ [CKKO21] and $\mathsf{NEXP} \subseteq \mathsf{P}_{/\mathsf{poly}}$ [ABM23]. As far as we know, there is no straightforward way to generalize these techniques to the unprovability of *uniform* upper bounds (e.g. $\mathsf{P} = \mathsf{NP}$ or $\mathsf{NP} = \mathsf{coNP}$), which is particularly relevant to the goal of proving the *independence* of $\mathsf{P} \neq \mathsf{NP}$ in PV. We hope that Theorem 1.6 could bring new insights to showing the unprovability of uniform upper bounds.

1.6 Related Works

Super-polynomial separations in proof complexity. There is a rich literature on proving conditional and unconditional super-polynomial separations in proof complexity. Here we provide an incomplete survey.

- (*Unconditional super-polynomial lower bound*). For instance, Urquhart [Urq87] proved that there is an explicit family of formulas that admit polynomial-size Frege proofs but require exponential-size Resolution proofs (see [Hak85, Bus87] for an alternative proof). Interested readers are referred to [BN21] for a comprehensive exposition of known lower bounds. We also note that a recent line of work highlights a connection between separations of propositional proof systems and subclasses of TFNP, see, e.g., [dRGR22, GHJ+24].

- (*Candidates hard formulas for stronger systems*). As super-polynomial lower bounds against strong proof systems such as Frege or Extended Frege seem out of reach of current techniques, people introduced candidate hard formulas, including *reflection principles* (see, e.g., [Kra19, Section 19.2]) and *proof complexity generators* (see, e.g., [Raz15, Kha22, Kra24]).

- (*Separations of bounded theories*). Separations of bounded theories can be viewed as proof complexity analogs of complexity class separations. Unconditional separations are known in certain *relativized* settings [Tha05, Jeř07b]. Buss [Bus95b] proved that the S_2, T_2 hierarchy does not collapse unless the polynomial-time hierarchy collapses. Several other separations are known under cryptographic assumptions [KP98, CT06, ILW23] and complexity-theoretic assumptions [Kra24].

Quantitative results in proof complexity. As far as we know, the only non-trivial quantitative proof complexity lower bound against strong systems is a quadratic lower bound against natural Frege systems due to Krajíček [Kra95, Chapter 13]. The exact exponents of polynomial-size proofs are sometimes analyzed (see, e.g., [Kra95, Chapter 13] and [Bus87]), but there has been little progress in optimizing the exact exponents.

Gödel's incompleteness in proof complexity. The problem of proving Gödel's incompleteness theorems for bounded theories was proposed by Parikh (see [Bus99]). This was later established for several bounded theories [Coo75, WP87, Bus86, Nel14].

A result of Naumov [Nau08] is in some sense a "dual" of our result: Instead of defining the complexity of arithmetic proofs via the size of its propositional translation, Naumov defines the complexity of propositional proofs via the size of the shortest arithmetic description of the proof (in, say, Peano Arithmetic). A lower bound for this measure is proved via a self-referential sentence. Note that our results are significantly more involved as we use bounded theories rather than Peano Arithmetic. Also, we prove an *upper bound* that matches our lower bound, up to a fixed polynomial time complexity overhead.

Our hierarchy theorem is quite similiar to an early result of Friedman (unpublished) and Pudlák [Pud86]. They proved that for standard first-order arithmetic theory that allows binary encoded numbers, there are sentences that can be described in $O(\log n)$ symbols, provable in $n^{O(1)}$ symbols, and requires at least $n^{\Omega(1)}$ symbols to prove. Their result can be viewed as a hierarchy theorem for arithemtic proof size, whereas our result is a hierarchy theorem for time complexity — size of propositional translations. It may not be surprising that these two results use very similar techniques, i.e., Gödel-style self-referential sentences. Nevertheless, the feasibility hierarchy theorem is significantly more involved as it is harder and less intuitive to analyze the time complexity of proofs.

Meta-mathematics of algorithm and complexity theory. There are two main lines of work in the meta-mathematics of algorithm design and complexity theory:

- (*Bounded reverse mathematics*). The program of bounded reverse mathematics aims to formalize results in algorithm design [Ngu08, HS08, LC11, CN10, Ječ23], combinatorics [Oja04], and complexity theory [Raz95, Ječ05, Ječ07a, Pic14, Pic15b, Pic15a, MP20, CLO24b, AT24, LLR24, CKK+25] in fragments of bounded arithmetic. In several cases, natural statements about algorithms and complexity theory are indeed equivalent to fundamental principles such as variants of the pigeonhole principle [Ječ05, CLO24b, AT24].
- (*Unprovability of complexity conjectures*). Another direction is to unconditionally prove the unprovability of complexity upper bounds [KO17, BKO20, BM20, CKKO21, ABM23] or complexity lower bounds [Kra11b, PS21, LO23, CLO24a, CG25]. The unprovability of lower bounds (that are believed to be true) can be viewed as a barrier to proving the lower bounds. On the other hand, the unprovability of upper bounds (that are believed to be false) can be viewed as a necessary intermediate step towards proving the lower bounds, as proving a lower bound is also a way to prove the unprovability of its negation.

We refer readers to [Oli25] for more detailed discussions.

Acknowledgement

We are grateful to Ryan Williams for his supports during the project and for proofreading an earlier draft. We thank Igor C. Oliveira for reading an early draft and bringing [Pud86] into our attention. We thank Zhengzhong Jin, Surya Mathialagan, and Abhishek Jain for discussion about the applications of bounded arithmetic proofs in cryptography. We also thank Marco L. Carmosino and Stefan Grosser for insightful discussions and anonymous conference reviewers for constructive comments. This project received supports from National Science Foundation under Grant CCF-2127597.

2 Technical Overview

In this section, we provide an overview of our main results: our definition of time complexity (see Theorem 1.1), two basic properties of time complexity (see Theorems 1.3 and 1.4), and the feasibility hierarchy theorem (see Theorem 1.5). Theorem 1.6 is relatively simple; interested readers are referred to Section 8 for a self-contained proof.

2.1 Definition of Time Complexity of Proofs

Recall that in Theorem 1.1, the complexity measures $\beta_e^+(\vec{n})$ and $\mu_\pi^{\mathsf{TC}}(\vec{n})$ are defined to be the size of a standard propositional translation of e and π [Coo75], respectively. We defer the definition of $\beta_e^{\mathsf{TC}}(\vec{n})$ to our discussion of the feasible translation theorem in Section 2.2. We will focus on the *time efficiency* of computing $\beta_e^+(\vec{n})$ and $\mu_\pi^{\mathsf{TC}}(\vec{n})$.

Let e be an equation and π be a proof. For simplicity, we assume that e, defined as $t_1(x) = t_2(x)$, has only one variable x. The formal definition of the propositional translation $[\cdot]^n_{\mathsf{Cook}}$ is technical (see Section 4.4 and [Kra19]), but it follows a simple idea:

- (*Equations*). For the PV equation e, both $t_1(x)$ and $t_2(x)$ are explicitly defined polynomial-time functions. Thus for any fixed $n \in \mathbb{N}$, we can construct, by the standard translation of algorithms to circuits (see [AB09, Chapter 6]), circuits $C_1, C_2 : \{0,1\}^n \to \{0,1\}$ that are equivalent to t_1 and t_2 on the input length n. By introducing auxiliary variables for each gate in the circuit, we construct a formula ϕ_n such that $C_1 \equiv C_2$ if and only if ϕ_n is a tautology.
- (*Proofs*). Suppose that π is a proof concluding the equation $e : t_1(x) = t_2(x)$, and $n \in \mathbb{N}$, our goal is to prove in Extended Frege that the formula ϕ_n constructed above is a tautology. If e is introduced by a deduction rule with premises $e_1(\vec{y}_1), \ldots, e_k(\vec{y}_k)$ in π, it can be verified that there are input lengths $\vec{m}_1, \ldots, \vec{m}_k$ for the variables $\vec{y}_1, \ldots, \vec{y}_k$ and an explicit polynomial-size Extended Frege proof

$$(\pi_e): \quad \frac{[e_1]^{\vec{m}_1}_{\mathsf{Cook}} \quad [e_2]^{\vec{m}_2}_{\mathsf{Cook}} \quad \cdots \quad [e_k]^{\vec{m}_k}_{\mathsf{Cook}}}{[e]^n_{\mathsf{Cook}}}.$$

  That is, we can deduce $[e]^n_{\mathsf{Cook}}$ from $[e_1]^{\vec{m}_1}_{\mathsf{Cook}}, \ldots, [e_k]^{\vec{m}_k}_{\mathsf{Cook}}$ with a polynomial-size Extended Frege proof. Therefore, we can recursively generate the translation of the proofs concluding $e_1, \ldots, e_k$ on the input lengths $\vec{m}_1, \ldots, \vec{m}_k$ and append the proof π_e to obtain a proof of $\phi_n = [e]^n_{\mathsf{Cook}}$.

Suppose that π and e are fixed, it can be verified that there are polynomial-time algorithms generating $[e]^n_{\mathsf{Cook}}$ and $[\pi]^n_{\mathsf{Cook}}$ given 1^n, where the exponent of the polynomial depends on π and e.

However, directly defining $\beta^+_e(n)$ and $\mu^{\mathsf{TC}}_\pi(n)$ as the size of $[e]^n_{\mathsf{Cook}}$ and $[\pi]^n_{\mathsf{Cook}}$ may not satisfy the efficiency requirement in Theorem 1.1: Both $\beta^+_e(n)$ and $\mu^{\mathsf{TC}}_\pi(n)$ should be computable in time $\mathsf{polylog}(n)$ (rather than $\mathsf{poly}(n)$), and the exponent of the running time should be *independent* of π and e. The main technical issue comes from the *induction rule* and the function introduced by *limited recursion*; we take the latter one as an example. Suppose that $f(y)$ is a function introduced as

$$f(\varepsilon) := c; \quad f(s_i(y)) := h_i(y, f(y)) \quad (i \in \{0,1\}),$$

where h_0, h_1 are existing functions. (We ignore the functions k_0, k_1 that upper bounds the growth rate of the recursion scheme, see Appendix A.1.) Let $\ell_f(n)$ be an output length upper bound of f on the input length n. In the propositional translation of f on the input length n, we need to recursively obtain the propositional translation of $h_i(y,z)$ on the following input lengths:

$$\begin{array}{lll}
(1) & |y| = 0, & |z| = \ell_f(0); \\
(2) & |y| = 1, & |z| = \ell_f(1); \\
 & \vdots & \vdots \\
(n) & |y| = n-1, & |z| = \ell_f(n-1).
\end{array}$$

To compute its size, we need to take a summation over n terms corresponding to the size of the propositional translation of h_i on these n pairs of input lengths. This requires $O(n) \gg \mathsf{polylog}(n)$ time. A similar issue occurs for computing the size of $[\pi]^n_{\mathsf{Cook}}$ with the induction rule.

Let $S(m_y, m_z)$ be the size of the propositional translation of h_i on the input length $|y| = m_y, |z| = m_z$. The idea to resolve the problem is simple: Instead of taking a summation over n terms, we **pad** the propositional translation to size exactly $n \cdot S(n, \ell_f(n))$. Note that this is an obvious size upper bound as both S and ℓ_f are non-decreasing functions. On the other hand, as both S and ℓ_f are polynomials, we have

$$\sum_{i=0}^{n-1} S(i, \ell_f(i)) \geq \frac{n}{2} \cdot S\left(\frac{n}{2}, \ell_f\left(\frac{n}{2}\right)\right) = \Omega(n \cdot S(n, \ell_f(n))),$$

which means that the upper bound is tight asymptotically.

Let $\beta_e^+(n)$ and $\mu_\pi^{\mathsf{TC}}(n)$ be the size for $[e]^n_{\mathsf{Cook}}$ and $[\pi]^n_{\mathsf{Cook}}$ (with the padding above). By a tedious but straightforward time complexity analysis, we can further prove that there are FPT algorithms computing these two measures: The time complexity of computing $\beta_e^+(n)$ is $\exp(\exp(O(|e|))) \cdot \mathsf{polylog}(n)$, and the time complexity of computing $\mu_\pi^{\mathsf{TC}}(n)$ is $\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(n)$.

Links to formal proofs. The formal definition of the time complexity can be found in Section 4 (see Section 4.3 for a self-contained definition). A detailed complexity analysis of the FPT algorithms is available in Section 5.

2.2 Feasible Translation Theorem

Let $e : t_1(x) = t_2(x)$ be an equation (with one variable x for simplicity). The feasible translation theorem (see Theorem 1.4) states that the time complexity of the correctness proof of the propositional translation of e is a fixed polynomial in $\beta_e^{\mathsf{TC}}(n)$, i.e., the time complexity of e.

As we mentioned before, the difference between the time complexity $\beta_e^{\mathsf{TC}}(n)$ of e and its propositional complexity $\beta_e^+(n)$ is that $\beta_e^{\mathsf{TC}}(n)$ takes into account (the time complexity of) the *complexity analysis* of functions in e. Indeed, for the standard formalization of PV, this only happens for functions introduced by *limited recursion*. To introduce a function $f(y)$ by

$$f(\varepsilon) := c; \quad f(s_i(y)) := h_i(y, f(y)) \quad (i \in \{0,1\})$$

from existing functions h_0, h_1, it is required that there are functions k_0, k_1 and valid proofs of an equation formalizing

$$|h_i(y,z)| \le |z| + |k_i(y)| \quad (i \in \{0,1\}). \tag{2.1}$$

This ensures the output length $f(y)$ is most

$$|f(y)| \le |c| + \sum_{0 \le i < |y|} |k_i(y_{\le i})|, \tag{2.2}$$

which is a polynomial in $|y|$.

We note that the proof of Equation (2.1) is not considered in the propositional complexity $\beta_e^+(n)$, but it is an *unavoidable time complexity overhead* to state the equation e, so it should be taken into account in $\beta_e^{\mathsf{TC}}(n)$. Also, it is crucial in the correctness proof of the propositional translation. As we mentioned in Section 2.1, the propositional translation of $e : t_1(x) = t_2(x)$ first converts t_1, t_2 into circuits C_1, C_2 and produces a formula formalizing $C_1 \equiv C_2$. Suppose that a function f, introduced by limited recursion, is used in one of the terms (say t_1). To define the circuit C corresponding to the function f on an input length, we calculate the output length based on the upper bound induced by Equation (2.2). To prove the correctness of the circuit in PV, however, we will have to make use of the PV proof of Equation (2.1).

Let $\ell_f(n)$ be the maximum output length of f on the input length n (called the *bounding value*, see Appendix A.3). Informally, for every function call to the function symbol f on the input length $|y| = m_y$ in the equation e, we need to take into account the time complexity of the proof of Equation (2.1) on the input length $|y| = m_y$ and $|z| = \ell_f(m_y)$ in $\beta_e^{\mathsf{TC}}(n)$. Similar to $\beta_e^+(n)$ and $\mu_e^{\mathsf{TC}}(n)$, one can verified that there is an FPT algorithm to compute $\beta_e^{\mathsf{TC}}(n)$ given the equation e and the input length n encoded in binary.

The proof of the feasible translation theorem follows directly from formalizing the standard correctness proof of the propositional translation of equations (see, e.g., [Coo75, Kra19]) in PV and analyzing its time complexity.

One technical issue is that the statement

"$\forall \vec{x}\ e(\vec{x})$ if and only if for every $\vec{n}$, $[e]^{\vec{n}}_{\mathsf{Cook}}$ is a tautology"

in Theorem 1.4 is not a universal sentence and therefore cannot be directly formalized in PV. Nevertheless, one can notice that the sentence can be formalized as the conjunction of two universal sentences that are both provable in PV with small time complexity overhead:

- (*Completeness*). For every $\vec{x} = (x_1, \ldots, x_k)$ such that $e(\vec{x})$ does not hold, $\mathsf{Assign}_e(\vec{x})$ outputs a falsifying assignment of $[e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}$, where $\mathsf{Assign}_e(\vec{x})$ is a PV function that depends on e.
- (*Soundness*). For every assignment w to $[e]^{\vec{x}}_{\mathsf{Cook}}$ such that the input variables encode $\vec{x}$, if w is a satisfying assignment, then $e(\vec{x})$ holds.

Links to formal proofs. The formal proof of the feasible translation theorem can be found in Section 6.2; the time complexity analysis crucially relies on the feasibility deduction theorem (see Theorem 6.2), which we cannot cover here due to page limitation.

2.3 Feasible Proof Generation Theorem

Similar to the feasible translation theorem, the feasible proof generation theorem can also be proved by formalizing the standard correctness proof (see, e.g., [Coo75, Kra19]) in PV and analyzing the time complexity of that correctness proof. Instead, we provide an alternative approach that proves a more general version of the theorem.

Let π be a proof (with one variable x for simplicity). Recall that the time complexity $\mu^{\mathsf{TC}}_\pi(n)$ of a proof π can be computed in time $f(|\pi|) \cdot \mathsf{polylog}(n)$, where $f(k) = \exp(\exp(O(k)))$. Suppose that π is given as an input and is encoded by $\langle \pi \rangle := \pi \circ 1 \circ 0^{f(\pi)}$, i.e., padding sufficiently many 0's, the computation of $\mu^{\mathsf{TC}}_\pi(\vec{n})$ takes polynomial time in its input length (see Sections 5.7 and 6.3.1 for more details).

Although the generation of $[\pi]^n_{\mathsf{Cook}}$ may take super-polynomial time, it can be verified that if $\mu^{\mathsf{TC}}_\pi(\vec{n}) \leq m$, the generation algorithm halts in fixed polynomial time in m. Similarly, the generation algorithm of $[e]^n_{\mathsf{Cook}}$ halts in fixed polynomial time in m if $\beta^{\mathsf{TC}}_e(\vec{n}) \leq m$. Furthermore, both these properties can be formalized and proved in PV if π and e are encoded by padding sufficiently many 0's. As a result, we can formalize the following statement in PV:

$$(\nabla) : \textit{for every padded } \langle \pi \rangle, 1^m, \textit{and } 1^n, \textit{if } \pi \textit{ is a valid } \mathsf{PV} \textit{ proof concluding } e, \textit{and } \mu^{\mathsf{TC}}_\pi(\vec{n}) \leq m, \textit{then } [\pi]^n_{\mathsf{Cook}} \textit{ is a valid } \mathsf{PV} \textit{ proof concluding } [e]^n_{\mathsf{Cook}}.$$

This statement can be proved in PV by formalizing the standard correctness proof.

We can now derive the feasibility hierarchy theorem from the provability of (∇). For any fixed proof π^*, the correctness proof of the propositional translation of π can be formalized by substituting $\langle \pi \rangle$ by $\langle \pi^* \rangle$ and m by $\mu^{\mathsf{TC}}_{\pi^*}(\vec{n})$ in (∇). (One can also choose an alternative formalization, but any straightforward formalization should be provably equivalent to this formalization with fixed polynomial time complexity overhead.) Subsequently, the correctness proof for π^* follows directly from the correctness proof of (∇).

Moreover, as the time complexity of the proof of (∇) is a fixed polynomial in its input length $|\langle \pi \rangle| + m + n$, the time complexity of the correctness proof for π^*, which follows by a simple substitution, is at most

$$(|\langle \pi^* \rangle| + \mu^{\mathsf{TC}}_{\pi^*}(n) + n)^{O(1)} = O_{\pi^*}\left(\mu^{\mathsf{TC}}_{\pi^*}(n)\right)^{O(1)},$$

where $O(1)$ hides an absolute constant and O_{π^*} hides a constant depending on π^*. This completes the time complexity analysis of the correctness proof for the propositional translation of π.

Links to formal proofs. The proof of the feasible proof generation theorem is in Section 6.3. The algorithms for computing the propositional translation $[\pi]^n_{\mathsf{Cook}}$ and $[e]^n_{\mathsf{Cook}}$ when π, e are given as the input are explained in Section 6.3.1. The formalization and proof of (∇) is in Section 6.3.2.

2.4 Feasibility Hierarchy Theorem

At a high level, e_d is a modification of a self-referential sentence: there is no PV proof π concluding e_d with time complexity β^{d+1}, where β denotes the time complexity of the equation e_d. It can be verified that the standard self-referential construction [Coo75] incurs a minor time complexity overhead (see Section 7). However, there are subtle technical issues with the encoding of π and the comparison of its time complexity — both the time complexity of π and e_d are functions of their input lengths, and it is unclear how we should compare these two functions.

Self-referential sentence. We now describe the self-referential sentence e_d in more details. The equation e_d will have two variables: a proof π and a string x. The string x will be used in two places:

- (*Input length for complexity comparison*). Recall that $\mu_\pi^{\mathsf{TC}}(n_\pi, n_x)$ and $\mu_{e_d}^{\mathsf{TC}}(n_\pi, n_x)$ denotes the time complexity of π and e_d. Instead of comparing these two functions, we will compare the values of these two functions on the input length $n_\pi = |x|$ and $n_x = |x|$.
- (*Time bound*). Note that $\mu_\pi^{\mathsf{TC}}(|x|, |x|)$ may not be efficiently computable when π is given as an input. We will try to compute $\mu_\pi^{\mathsf{TC}}(|x|, |x|)$ and use the length of $|x|$ as a time bound.

Therefore, the sentence e_d indeed formalizes:

for every π and x, we attempt to compute $\mu_\pi^{\mathsf{TC}}(|x|, |x|)$ by running the algorithm for $|x|$ steps, then either $\mu_\pi^{\mathsf{TC}}(|x|, |x|)$ does not halt in $|x|$ steps, or $\mu_\pi^{\mathsf{TC}}(|x|, |x|) \geq (\beta_{e_d}^{\mathsf{TC}}(|x|, |x|))^{d+1}$, or π is not a valid PV proof concluding e_d.

Proof of the lower bound. The lower bound for e_d is a simple adaption of the proof of Gödel's first incompleteness theorem [Göd31, Coo75]. Let $\beta = \beta_{e_d}^{\mathsf{TC}}(n_\pi, n_x)$ be the time complexity of e_d. Assume for contradiction that e_d has a PV proof π^* that has time complexity $O(\beta^d)$.

- Recall that the algorithm computing $\mu_\pi^{\mathsf{TC}}(|x|, |x|)$ runs in time $\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(|x|)$, which is much smaller than $|x|$ when $|\pi| = O(1)$. Therefore, there is an $n_0 \in \mathbb{N}$ such that for every $n \geq n_0$, $\mu_\pi^{\mathsf{TC}}(1^n, 1^n)$ halts in at most n steps.
- Moreover, as π^* has time complexity $O(\beta^d)$, there is an $n_0' \in \mathbb{N}$ such that for every $n \geq n_0'$,

$$\mu_{\pi^*}^{\mathsf{TC}}(1^n, 1^n) \leq (\beta_{e_d}^{\mathsf{TC}}(1^n, 1^n))^{d+1}.$$

Since e_d is a provable sentence in PV, we know by the soundness of PV that it is true in the standard model. Subsequently, $e_d(\pi/\pi^*, x/1^n)$ is a true equation for every n. Therefore, if we choose $n \geq \max(n_0, n_0')$ such that the two bullets above hold, it must be the case that π^* is not a valid PV proof concluding e_d. This yields a contradiction.

Proof of the upper bound. The upper bound proof for e_d is significantly more involved. Due to page limitations, we will only explain a brief intuition of the proof.

The key technical ingredient of the upper bound is a PV equation Con_d called the *bounded feasibility consistency sentence*. It states that there is no PV proof of contradiction with time complexity at most n^d. In more detail, $\mathsf{Con}_d(\pi, x)$ has two variables π and x, and it states that

for every π and x, either π is not a PV proof concluding $s_0(x) = s_1(x)$, or $\mu_\pi^{\mathsf{TC}}(|x|, |x|) \geq d \cdot |x|^d$.

Similar to the definition of e_d, $\mu_\pi^{\mathsf{TC}}(|x|, |x|)$ may not be efficiently computable. This can be resolved by the same trick of running the algorithm computing $\mu_\pi^{\mathsf{TC}}(|x|, |x|)$ for $|x|$ steps, which we omit here.

The proof of the upper bound involves two steps:

- (*Efficient proof of* Con_d). We show that there is a PV proof of $\mathsf{Con}_d(\pi, x)$, and, in addition, the time complexity of the proof is a fixed polynomial in $(|\pi| + |x|)^d$. Indeed, the PV proof of $\mathsf{Con}_d(\pi, x)$ formalizes Cook's consistency proof of PV in [Coo75, Corollary 2.17] in the setting that π has bounded time complexity. In more detail, Cook's consistency proof of PV combines two technical tools:
  (i) (*Propositional translation*). The first tool is the correctness of the propositional translation: Given a proof π of $s_0(x) = s_1(x)$ and an input length n, we can produce an EF proof concluding the propositional translation of $s_0(x) = s_1(x)$, which can further derive a contradiction $\bot$. In the setting that π has bounded time complexity, this is exactly the generalized version of the feasible proof generation theorem, as we mentioned in Section 2.3.
  (ii) (*Soundness of* EF). The second tool is the soundness of EF, namely, there is no EF proof of $\bot$. This is known to be provable in PV (see Theorem 3.2).

  In addition, one can verify that each part incurs a fixed polynomial time complexity overhead.
- (*Reduction from* e_d *to* $\mathsf{Con}_{O(d)}$). We then prove that there is a *reduction* from e_d to $\mathsf{Con}_{O(d)}$, namely there is a constant $c \geq 1$ such that if e_d is provable, $\mathsf{Con}_{c \cdot d}$ is also provable with a fixed polynomial time complexity overhead. This is proved by adapting Gödel's proof that the self-referential sentence (in Peano Arithmetic) is implied by the consistency of Peano Arithmetic [Göd31] (see also [Coo75, Equation 4.10] for the proof in PV).

By combining these two parts, we can prove e_d by first proving $\mathsf{Con}_{O(d)}$ and applying the reduction. Each of the steps has a fixed polynomial running time in $(\beta_e^{\mathsf{TC}}(n_\pi, n_x))^d$, and therefore the total time complexity is at most a fixed polynomial in $(\beta_e^{\mathsf{TC}}(n_\pi, n_x))^d$.

Another explicit witness. Note that by the time-efficient reduction from e_d to $\mathsf{Con}_{O(d)}$ and the lower bound for e_d, we can also derive a time complexity lower bound for the bounded feasibility consistency sentence. This, combined with the efficient proof of Con_d, shows that the feasibility consistency sentence is also an explicit witness of the feasibility hierarchy theorem.

Links to formal proofs. The proof of the feasibility hierarchy theorem is in Section 7: In Section 7.1 we prove an time complexity upper bound for proving the bounded feasibility consistency sentence; both the upper and lower bounds for e_d are proved in Section 7.2; in Section 7.3, we prove that the bounded feasibility consistency sentence is also an explicit witness of the feasibility hierarchy theorem.

3 Preliminaries

We assume basic familiarity with bounded arithmetic (see, e.g., [Kra95, Bus86]), proof complexity (see, e.g., [Kra19, BN21]), and computational complexity (see, e.g., [AB09]).

Notation. Let $\mathbb{N} := \{0, 1, \ldots, n, \ldots\}$ be the set of natural numbers, $[n] = \{1, 2, \ldots, n\}$. We use 1^n to denote the all-1 string of length n.

For a string x, we use $|x|$ to denote the length of the string, and $x_{\leq i}$ (resp. $x_{<i}$) to denote the prefix of x of length i (resp. $i-1$). We use $\vec{x}$ as shorthand of a vector of variables $(x_1, \ldots, x_k)$, and $\vec{n} \in \vec{\mathbb{N}}$ as the shorthand of a vector of numbers $(n_1, \ldots, n_k)$. We use the standard notation $x \in \mathsf{Log}$ to denote that x is an abbreviation of $|X|$ for another variable X; in particular, $\forall x \in \mathsf{Log}\ \varphi(x)$ is an abbreviation of $\forall X\ \varphi(|X|)$, and $\exists x \in \mathsf{Log}\ \varphi(x)$ is an abbreviation of $\exists X\ \varphi(|X|)$.

For a formula ϕ and term t, we use $\phi[x/t]$ to denote the formula obtained by substituting all free occurrences of x to the term t.

For a function $f(\vec{n})$, we use $O(f(\vec{n}))$ to denote a function $g(\vec{n})$ satisfying that there are constants $c, n_0 \geq 1$ such that for $\vec{n} = (n_1, \ldots, n_k)$ satisfying that $n_1, \ldots, n_k \geq n_0$, we have $f(\vec{n}) \leq c \cdot g(\vec{n})$. We use $O_d(f(\vec{n}))$ to emphasize that the constant c may depend on another constant d.

3.1 Equational Theory PV

We will primarily use Cook's formalization of the PV as an equational theory [Coo75], see Appendix A for more detail. Our results are robust with respect to formalizations and should applied to any equivalent formalizations of PV. However, the time complexity measure might differ up to a fixed polynomial.

We will use the standard notation $\mathsf{PV} \vdash s = t$ to denote that there is a PV proof of the equation $s = t$. The equational theory PV is defined as the set of equations with PV proofs.

3.2 First-Order Theory PV_1

The theory PV_1 is the first-order extension of PV. In more details, the language of PV_1 consists of all PV function symbols, and for every provable equation $s(\vec{x}) = t(\vec{x})$ in PV, $\forall \vec{x}\ s(\vec{x}) = t(\vec{x})$ is an axiom of PV_1. We refer readers to [Kra95] for more detailed discussions.

Theorem 3.1 (witnessing theorem, see, e.g., [Kra95]). *Let $\varphi(\vec{x}, y)$ be a quantifier-free formula in the language of PV_1, and $\vec{x}, y$ be the variables of φ. Suppose that $\mathsf{PV}_1 \vdash \forall \vec{x}\ \exists y\ \varphi(\vec{x}, y)$, then there is a PV-function f such that $\mathsf{PV}_1 \vdash \forall \vec{x}\ \varphi(\vec{x}, f(\vec{x}))$.*

3.3 Simulation of Proof Systems by EF

Definition 3.1. Let M be a PV function and $k, m_0 \geq 1$ be integers. We define UB_M^{k,n_0} be conjunction of two sentences Sound_M^{k,m_0} and $\mathsf{Complete}_M^{k,m_0}$, where

$$\begin{aligned}
\mathsf{Sound}_M^{k,m_0} := {} & \forall n, m \in \mathsf{Log},\ m \geq m_0, \forall \text{ formula } \varphi \in \{0,1\}^m \text{ with } n \text{ variables such that:} \\
& \forall \text{ assignment } x \in \{0,1\}^n : \varphi(x) = 1 \text{ or } \forall \text{ witness } \pi \text{ of size at most } m^k : M(\varphi, \pi) = 0; \\
\mathsf{Complete}_M^{k,m_0} := {} & \forall n, m \in \mathsf{Log},\ m \geq m_0, \forall \text{ formula } \varphi \in \{0,1\}^m \text{ with } n \text{ variables such that:} \\
& \exists \text{ assignment } x \in \{0,1\}^n : \varphi(x) = 0 \text{ or } \exists \text{ witness } \pi \text{ of size at most } m^k : M(\varphi, \pi) = 1.
\end{aligned}$$

In particular, let $V_{\mathsf{EF}}(\varphi, \pi)$ be a straightforward PV function that verifies whether π is an EF proof of φ, i.e., it outputs 1 (resp. 0) if π is (resp. is not) a valid EF proof of the formula φ, then $\mathsf{Complete}_{V_{\mathsf{EF}}}^{O(1),\Theta(1)}$ means that EF is p-bounded.

Theorem 3.2 ([Coo75]). $\mathsf{PV}_1 \vdash \mathsf{Sound}_{V_{\mathsf{EF}}}^{k,m_0}$ *for any constants $k, m_0 \geq 1$.*

Note that PV_1 is a conservative extension of PV [Coo75, Kra95]; any PV_1 provable equations in the language of PV are also provable in PV. Therefore, if we formalize $\mathsf{Sound}_{V_{\mathsf{EF}}}^{k,m_0}$ as a PV equation in a straightforward way, it is also provable in PV.

The main theorem of [Coo75] shows that a proof system is provably sound in PV if and only if it is PV-provably p-simulated by M.

Theorem 3.3 ([Coo75]). *Let M be a PV function. Then $\mathsf{PV}_1 \vdash \mathsf{Sound}_M^{k,m_0}$ if and only if PV_1 proves that EF p-simulates M. That is, there are constants k', m_0' such that PV_1 proves the following statement: For every formula φ and proof π such that $|\varphi| \geq m_0'$, $|\pi| \leq |\varphi|^k$, if $M(\varphi, \pi) = 1$, there is a EF proof π_{EF} such that $V_{\mathsf{EF}}(\varphi, \pi_{\mathsf{EF}}) = 1$ and $|\pi_{\mathsf{EF}}| \leq |\pi|^{k'}$.*

4 Formal Definition of the Time Complexity of PV

In this section, we formally define the time complexity of PV equations and proofs.

- In Section 4.1, we propose a definition (called *Cook complexity*) based on Cook's propositional translations. Cook complexity is indeed a class of complexity measures — every propositional translation induces corresponding Cook complexity measures. We need to deal with two technical issues mentioned in Section 4.2.
- In Section 4.3, we provide a self-contained formal definition of time complexity that formalizes the time complexity of PV equations and proofs.
- We prove in Section 4.4 that time complexity can be viewed as Cook complexity for a natural propositional translation. This propositional translation involves unnecessary padding. Nevertheless, we prove in Section 4.5 that the padding does not incur a super-constant overhead.
- In Section 4.6, we provide examples that proofs of the same length differ significantly in terms of their time complexity. This shows that time complexity is not a trivial measure derived from the length of the proof.

4.1 High-Level Idea: Complexity Measure from Cook's Translation

Cook's propositional translation [Coo75] for PV consists of two translations. Let $k \in \mathbb{N}$ be the number of variables and $s(\vec{x}) = t(\vec{x})$ be a PV equation.

- First, a translation from PV equations $s(\vec{x}) = t(\vec{x})$ to families of *polynomial-size* propositional formulas $\{\varphi_{\vec{n}}\}_{\vec{n}\in\mathbb{N}^k}$, denoted by $[s = t]^{\vec{n}}_{\mathsf{Cook}}$, is defined such that $\varphi_{\vec{n}}$ is a formula that is a tautology if and only if $s(\vec{x}) = t(\vec{x})$ for any $\vec{x} = (x_1, x_2, \ldots, x_k) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \cdots \times \{0,1\}^{n_k}$.
- Then it is proved that given any PV proof π of the equation $s = t$ and any $\vec{n} \in \mathbb{N}^k$, there is a *polynomial-size* EF proof of $\varphi_{\vec{n}}$. For simplicity, we also denote the EF proof by $[\pi]^{\vec{n}}_{\mathsf{Cook}}$.

These two translations reveal two aspects of the feasibility of PV. The first translation can be interpreted as the *efficiency of verification*: Given any input $\vec{u} = (u_1, \ldots, u_k)$ such that $u_i \in \{0,1\}^{n_i}$, there is an algorithm in $\mathsf{poly}(\vec{n})$ time, namely evaluating the formula $[s = t]^{\vec{n}}_{\mathsf{Cook}}$ that determines whether $s(\vec{u}) = t(\vec{u})$. The second translation can be interpreted as the *efficiency of a proof*: If $s(\vec{x}) = t(\vec{x})$ admits a PV proof π, given any input length $\vec{n} \in \mathbb{N}^k$, there is a uniform mean of verification in $\mathsf{poly}(\vec{n})$ time, namely verifying the EF proof $[\pi]^{\vec{n}}_{\mathsf{Cook}}$, that the formula $[s = t]^{\vec{n}}_{\mathsf{Cook}}$ is a tautology.

Fix any standard propositional translation $[\cdot]_{\mathsf{Cook}}$ for equations and functions. We can introduce the propositional complexity and the proof length complexity corresponding to $[\cdot]_{\mathsf{Cook}}$ as follows:

Definition 4.1 (propositional complexity)**.** Let $e : s(\vec{x}) = t(\vec{x})$ be a PV equation. The propositional complexity of the equation under the translation $[\cdot]_{\mathsf{Cook}}$ is the function β_e^{Cook}, where $\beta_e^{\mathsf{Cook}}(\vec{n})$ is the bit-length of the proposition formula $[e]^{\vec{n}}_{\mathsf{Cook}}$.

Definition 4.2 (proof length complexity)**.** Let e be a PV equation and π be a PV proof of e. The proof length complexity of π under the translation $[\cdot]_{\mathsf{Cook}}$ is defined as the function μ_π^{Cook}, where $\mu_\pi^{\mathsf{Cook}}(\vec{n})$ is the bit-length of the EF proof $[\pi]^{\vec{n}}_{\mathsf{Cook}}$.

For simplicity, we use the name Cook complexity for both propositional and proof length complexity measure: The propositional complexity of a PV equation e and the proof length complexity π of a PV proof are called the Cook complexity of e and π, respectively.

Remark 4.3. Note that Cook complexity is a **class of complexity measures**, as one can choose to work with different propositional translation $[\cdot]_{\mathsf{Cook}}$. These complexity measures may differ drastically; indeed, as mentioned in Section 1.1.1, one can add unnecessary padding so that the size of the propositional translation depends only on the size of the PV proof.

We make two additional remarks on the definition.

1. In this paper, we will not consider the encoding scheme for propositional formulas or EF proofs. Our theorems are robust with respect to formalizations, though the exact constants in our theorems will depend on the particular formalization we will use — it remains an interesting open problem to optimize the constants with efficient encoding schemes (see also Section 4.4 for more discussion).

2. It is clear that the EF proof $[\pi]^{\vec{n}}_{\mathsf{Cook}}$ of the propositional formula $\varphi_{\vec{n}} := [e]^{\vec{n}}_{\mathsf{Cook}}$ should at least contain $\varphi_{\vec{n}}$ in any reasonable encoding schemes. Therefore, $\mu_\pi^{\mathsf{Cook}}(\vec{n}) \geq \beta_\pi^{\mathsf{Cook}}(\vec{n})$. Given that the propositional complexity is a lower bound of the proof length complexity, it might be possible to measure the proof length complexity as a function of the propositional complexity of its conclusion. For instance, we may say that a PV proof π of the equation e has proof complexity $O(\beta^c)$ if there are constants n_0, c' such that for $n_1, n_2, \ldots, n_k \geq n_0$,

$$\mu_\pi^{\mathsf{Cook}}(n_1, \ldots, n_k) \leq c' \cdot \left(\beta_\pi^{\mathsf{Cook}}(n_1, \ldots, n_k)\right)^c.$$

It follows immediately from Cook's theorem of propositional translation [Coo75] that the Cook complexity of any PV equation or proof is always bounded by a polynomial.

Proposition 4.1. *For every valid* PV *equation* e, $\beta_e^{\mathsf{Cook}}(\vec{n}) = O((n_1 + \cdots + n_k)^c)$ *for some constant* $c \in \mathbb{N}$.

Proposition 4.2. *For every valid* PV *proof* π, $\mu_e^{\mathsf{Cook}}(\vec{n}) = O((n_1 + \cdots + n_k)^c)$ *for some constant* $c \in \mathbb{N}$.

4.2 Technical Issues

Cook complexity serves as a natural definition of the complexity of PV proofs. Nevertheless, there are technical issues that we need to take into account to ensure that the complexity measure admits desired mathematical properties.

Lack of feasible translation property. At a high level, the time complexity of an equation $e : s(\vec{x}) = t(\vec{x})$ characterizes the total cost to *verify* the equation on any particular input $\vec{u}$. The Cook complexity of an equation interprets the cost of verifying the equation as the size of its propositional translation $[e]^{\vec{n}}_{\mathsf{Cook}}$. Such interpretation does not match with the informal notion unless the *correctness of propositional translation* can be proved with little cost. In particular, the statement

$$\forall \vec{x}\ s(\vec{x}) = t(\vec{x}) \leftrightarrow [e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}} \text{ is a tautology} \tag{4.1}$$

should be provable, and the proof should not incur significant overhead in terms of the time complexity of proofs.

To be more specific, let $\beta_e(\cdot)$ be a complexity measure for equations and $\mu_e(\cdot)$ be a complexity measure for proofs, we say that the pair of measures (β_e, μ_π) admit feasible translation property if there is a fixed constant $d \in \mathbb{N}$ such that for any equation $e : s(\vec{x}) = t(\vec{x})$, Equation (4.1) admits a proof π such that $\mu_\pi(\vec{n}) = O((\beta_e(\vec{n}))^d)$. In more detail, Equation (4.1) should be formalized as two PV equations:

- (*Completeness*). For every $\vec{x}$ such that $s(\vec{x}) \neq t(\vec{x})$, $\mathsf{Assign}_e(\vec{x})$ outputs a falsifying assignment of $[e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}$, where $\mathsf{Assign}_e(\vec{x})$ is a PV function that depends on e.
- (*Soundness*). For every assignment w such that the input variables encode $\vec{x}$, if w is a satisfying assignment of $[e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}$, then $s(\vec{x}) = t(\vec{x})$.

The formalizations of both equations are straightforward.

We now argue informally that Cook complexity $(\mu_\pi^{\mathsf{Cook}}, \beta_\pi^{\mathsf{Cook}})$ may not have feasible translation property. Assume for contradiction that Cook complexity has feasible translation property with translation overhead $d \in \mathbb{N}$. Let $f(\vec{x}, y)$ be a PV function introduced by limited recursion via other PV functions $g(\vec{x})$,

$h_i(\vec{x}, y, z)$, $k_i(\vec{x}, y)$, $i \in \{0,1\}$, where $k_i(\vec{x}, y)$ is used to bound the output length of the recursion scheme $h_i(\vec{x}, y, z)$ by a PV proof of

$$\mathsf{ITR}(h_i(\vec{x}, y, z), z \circ k_i(\vec{x}, y)) = \varepsilon. \tag{4.2}$$

In general, it could be the case that the computational complexity of k_i is much larger than the computational complexity of f, g, and h_i, as the efficiency of an efficient algorithm may require inefficient proofs. For a concrete example, the following algorithm runs in polynomial time under Cramér's conjecture, which might be true but not feasibly provable:

- Given 1^n, enumerate $n, n+1, n+2, \ldots$ and run the AKS algorithm [AKS04] to find a prime number.

For simplicity, we assume that f, g, h_i, k_i do not have any additional parameter $\vec{x}$. Suppose that we want to prove a certain property of $f(y)$, say $e : f(y) = p(y)$ for another PV function $p(y)$. As the propositional translation of the equation $e : f(y) = p(y)$ does not take into account the complexity of $k_i(y)$, it could be the case that the computational complexity of $k_i(y)$ is $\Omega(\beta_e(y)^{d+1})$ for $i \in \{0,1\}$. Moreover, the proof π_i^k of Equation (4.2) may have time complexity $\mu_{\pi_i^k}(y) = \Omega(\beta_e(y)^{d+1})$ as it is also not taken into account.

To add more detail, the propositional translation of the function $f(y)$ consists of intermediate variables intended to encode the output of $f(\varepsilon), f(y_1), f(y_1y_2), \ldots, f(y)$, where $y = y_1y_2 \ldots y_n$. The output length of $f(y_{\le i})$ is bounded by

$$m_j(y) := |g| + \sum_{0 \le j < i} k_0(y_{\le j}) + \sum_{0 \le j < i} k_0(y_{\le j}),$$

and thus the total number of intermediate variables is $m_0(y) + m_1(y) + \cdots + m_n(y)$. (Note that other intermediate variables and constraints will need to be introduced to ensure the correctness of the computation.) However, to prove that the propositional translation is correct (as required by the feasible translation property), we also need to prove that the output length of $f(y_{\le j})$ is indeed at most $m_j(y)$. This may require a white-box inspection of the computation $k_0(y_{\le j}), k_1(y_{\le j})$ as well as the proof of Equation (4.2), while both of them may have complexity $\Omega(\beta_e(y)^{d+1}) \gg \beta_e(y)^d$.

Inefficiency of computing Cook complexity. One of the key characteristics of the Cook complexity measures is that they are *efficiently computable* for fixed equations and proofs. Let $\vec{n} = (n_1, \ldots, n_k) \in \mathbb{N}^k$. For any fixed equation $e : s(\vec{x}) = t(\vec{x})$, there is a polynomial-time algorithm $E_\beta(1^{n_1+\cdots+n_k})$ that outputs $\beta^e_{\mathsf{Cook}}(\vec{n})$. Similarly, there is a polynomial-time algorithm $E_\mu(1^{n_1+\cdots+n_k})$ that outputs $\mu^e_{\mathsf{Cook}}(\vec{n})$. These two properties are easy corollaries of the fact that Cook's translations for both equations and proofs, namely the functions

$$(1^{n_1}, \ldots, 1^{n_k}) \mapsto [e]^{\vec{n}}_{\mathsf{Cook}}, \quad (1^{n_1}, \ldots, 1^{n_k}) \mapsto [\pi]^{\vec{n}}_{\mathsf{Cook}},$$

are polynomial-time computable for any PV equation e and proof π. Indeed, for any fixed PV equation e and proof π, it is proved in [Coo75] that there are straightforward PV functions that compute these two functions (and thus also E_β and E_μ).

However, a technical issue of Cook complexity is that if we parameterize the proof length $k = |\pi|$, the algorithm may not be efficient. Recall that Cook complexity of equations and proofs is defined with respect to a fixed propositional translation $[\cdot]_{\mathsf{Cook}}$. Depending on the exact definition of $[\cdot]_{\mathsf{Cook}}$, the algorithm in [Coo75] may take $O(n^{f(k)})$ time to compute the Cook complexity of a proof π for some function f, where $n = n_1 + \cdots + n_k$ denotes the sum of input lengths. This means that given any proof π of super-constant length, it is impossible to even formalize the statement

$$\text{"}\pi \textit{ has Cook complexity at most } n^{10}\text{"}$$

in PV, as the computation of Cook complexity is infeasible.

It could be much more interesting if the complexity measure is efficiently computable given proofs as input, say in $\mathsf{poly}(k, \vec{n})$ time, or at least $f(k) \cdot \mathsf{poly}(n)$ time for some function f (i.e. there is an *FPT algorithm* to compute the complexity measure). In the latter case, the statement "π *has Cook complexity at most* n^{10}" could be formalized for a proof π of super-constant length by padding $f(k)$ dummy bits after the proof π of length k.

Time complexity of proofs and equations: a bird's eye view. We will define time complexity of PV equations and proofs by modifying Cook complexity to address the two technical issues above. Our modification can be summarized as follows at a high level:

- (*Feasible Translation Property*). We will define the complexity of an equation e by taking into account both the size of its propositional translation $[e]^{\vec{n}}_{\mathsf{Cook}}$ as well as the correctness proof of the translation. For our formalization of PV, this includes the complexity of bounding functions k_i for limited recursions and their correctness proof.
- (*Computational Efficiency*). We will carefully choose a particular formalization of the propositional translation $[\cdot]_{\mathsf{Cook}}$ such that there is an FPT algorithm to compute the Cook complexity under the propositional translation. This propositional translation has obvious redundancy in dealing with recursively defined functions and the induction rule, but it can be proved that such redundancy does not incur a super-linear blowup.

4.3 Time Complexity of Proofs and Equations

We now define the time complexity (TC) of equations and proofs that resolve the technical issues we mentioned above. We provide a self-contained definition here for completeness.

Let $\ell_t(x_1, \ldots, x_k)$ be the *bounding value* of the term $t(x_1, \ldots, x_k)$, i.e., the output of $t(x_1, \ldots, x_k)$ is of length at most $\ell_t(|x_1|, \ldots, |x_k|)$ (see Appendix A.3 for more detail). Let $c = O(1)$ be a constant to be determined later, which is called the formalization overhead.

4.3.1 Additional Time Complexity

We will first define the additional time complexity (ATC) of an equation or a proof line in the context of a proof π as follows.

- (*ATC of Equations*). Let π be a valid PV proof consisting of proof lines $e_1, \ldots, e_m$, and $e : t_1(x_1, \ldots, x_k) = t_2(x_1, \ldots, x_k)$ be an equation such that each function symbol in e has been defined in the proof π. We will define the ATC of the equation e in the context π as a function $\beta^+_{e,\pi}(n_1, \ldots, n_k)$, where n_i denotes the length of x_i.
- (*ATC of Proof Lines*). Let π be a valid PV proof consisting of proof lines $e_1, \ldots, e_m$, e be a valid proof line after π, and $x_1, \ldots, x_k$ be the variables in e. We will define the ATC of the new proof line e in the context of π as a function $\mu^+_{e,\pi}(n_1, \ldots, n_k)$, where n_i denotes the length of x_i.

Let π be a PV proof and $e_1, e_2, \ldots, e_m$ be the proof lines in π. Each e_i is introduced either by axioms or from previous equations by deduction rules. Let $e : t_1(\vec{x}) = t_2(\vec{x})$ be a new equation, $\vec{x} = (x_1, \ldots, x_k)$.

ATC of Equations. We will define the ATC of the terms $t_1(\vec{x})$ and $t_2(\vec{x})$ in the context of π, denoted by $\beta^+_{t_1,\pi}(\vec{n})$ and $\beta^+_{t_2,\pi}(\vec{n})$ and define the ATC of the equation e as $\beta^+_{e,\pi}(\vec{n}) := \beta^+_{t_1,\pi}(\vec{n}) + \beta^+_{t_2,\pi}(\vec{n}) + c \cdot (\ell_{t_1}(\vec{n}) + \ell_{t_2}(\vec{n}))$; the last term is to consider the complexity of comparing the output of t_1 and t_2.

Let $t \in \{t_1, t_2\}$. We define $\beta^+_{t,\pi}$ by structural induction on the definition of t.

- (*Base Case*). If t is ε, x, $s_0(x)$, $s_1(x)$, or $\mathsf{TR}(x)$, $\beta^+_{t,\pi}(n) := c \cdot n$. Otherwise, i.e., t is $\mathsf{ITR}(x_1, x_2)$ or $\circ(x_1, x_2)$, $\beta^+_{t,\pi}(n) := c \cdot n_1 \cdot n_2$.
- (*Functions via Composition*). If t is a function introduced by composition, i.e., $f_{t'}(\vec{x}) = t(\vec{x})$ for some valid term t' in the context of π, $\beta^+_{t,\pi}(\vec{n}) := \beta^+_{t',\pi}(\vec{n})$.

- (*Functions via Limited Recursion*). Suppose that t is a function $f(\vec{u}, v)$ introduced by limited recursion from $g(\vec{u})$, $h_i(\vec{u}, v, z)$, $k_i(\vec{u}, v)$, $i \in \{0,1\}$, $\vec{u} = (u_1, \ldots, u_i), v$ are variables (i.e. $u_1, \ldots, u_i, v \in \{x_1, \ldots, x_k\}$). Let $m_1, \ldots, m_i, m_v$ be the input length of $u_1, \ldots, u_i, v$, respectively. The ATC of t is

$$\begin{aligned}&\beta^+_{t,\pi}(m_1, \ldots, m_i, m_v)\\ &:= c \cdot \left(\beta^+_{g,\pi}(m_1, \ldots, m_i) + m_v \cdot \sum_{\sigma \in \{0,1\}} \beta^+_{h_\sigma,\pi}(m_1, \ldots, m_i, m_v, \ell_f(m_1, \ldots, m_i, m_v)) \right). \end{aligned} \tag{4.3}$$

  where ℓ_f is the bounded value of f.

- (*Composition*). Suppose that t is of form $f(s_1, \ldots, s_i)$ for a function f and terms $s_1, \ldots, s_i$. The ATC of t is defined as

$$\beta^+_{t,\pi}(\vec{n}) := c \cdot \left(\beta^+_{f,\pi}(\ell_{s_1}(\vec{n}), \ldots, \ell_{s_i}(\vec{n})) + \sum_{j \in [i]} \beta^+_{s_i,\pi}(\vec{n}) \right),$$

  where $\ell_{s_1}, \ldots, \ell_{s_i}$ are the bounding values of $s_1, \ldots, s_i$, respectively.

Remark 4.4. In the definition of ATC for functions introduced via limited recursion (see Equation (4.3)), the second term $m_v \cdot \beta^+_{h,\pi}(m_1, \ldots, m_i, m_v, \ell_f(m_1, \ldots, m_i, m_v))$ is to capture the complexity of the recursive computation of $f(\vec{u}, v)$:

$$\begin{aligned}&f(\vec{u}, \varepsilon) = g(\vec{u}),\\ &f(\vec{u}, v_1) = h_{v_1}(\vec{u}, v_1, f(\vec{u}, \varepsilon)),\\ &\ldots\\ &f(\vec{u}, v_1 v_2 \ldots v_{m_v}) = h_{v_{m_v}}(\vec{u}, v_1 v_2 \ldots v_{m_v - 1}, f(\vec{u}, v_1 v_2 \ldots v_{m_v - 1})).\end{aligned}$$

In this procedure, we need to evaluate h for m_v times, each of which is on input length at most m_1, m_2, $\ldots$, m_i, m_v, $\ell_f(m_1, \ldots, m_i, m_v)$ for its $i + 2$ variables.

We note that this estimation has obvious redundancy. If we attempt to evaluate f, we may find out that the actual input lengths are much smaller, and thus the actual computational complexity is smaller than the estimation in Equation (4.3). Nevertheless, we argue that Equation (4.3) is a good definition as it only incurs a small overhead (see Theorem 4.4).

ATC of Proof Lines. Suppose that e is a new proof line after π, we will define the ATC of proof line e in the context of π. Consider the axiom or deduction rule that we use to introduce the new line e.

- (*Axioms for Functions*). Suppose that e is a proof line introduced as an axiom for the functionality of initial functions or introduced functions, we define $\mu^+_{e,\pi}(\vec{n}) := c \cdot \beta^+_{e,\pi}(\vec{n})$. For instance, if e_i is one of

$$\mathsf{TR}(s_i(x)) = x, \quad \mathsf{ITR}(x, \varepsilon) = x, \quad x \circ \varepsilon = x, \quad (i \in \{0,1\}),$$

  the ATC of e is defined as $\mu^+_{e,\pi}(n) := c \cdot n$.

- (*Function Introduction*). Suppose that e is a proof line that introduces a function symbol by the composition rule or the limited recursion rule the time complexity of e is defined as 0.

- (*Logical Rules*). Suppose that e is derived from a logical rule. If the logical rule is (L1), (L2), or (L3), we simply define $\mu^+_{e,\pi}(\vec{n}) := c \cdot \beta^+_{e,\pi}(\vec{n})$.

  If the logical rule is (L4), i.e., e is of form $s_1[y/s_3] = s_2[y/s_3]$, where s_1, s_2, s_3 are PV terms such that $e' : s_1 = s_2$ is in π, and y is a variable. Assume without loss of generality that y is different from $x_1, \ldots, x_k$ and y has at least one occurrence in e'. In such case, variables in s_3 are among $x_1, \ldots, x_k$. Let $n_1, \ldots, n_k$ be the input length of $x_1, \ldots, x_k$, and n_y be the input length of y. We define

$$\mu^+_{e,\pi}(n_1, \ldots, n_k) := c \cdot \left(\beta^+_{e',\pi}(n_1, \ldots, n_k, n_y := \ell_{s_3}(n_1, \ldots, n_k)) + \beta^+_{s_3,\pi}(n_1, \ldots, n_k) \right).$$

- (*Induction*). Suppose that e is derived from the induction rule. Without loss of generality, we assume that x_k is the induction variable. Let $f_1(\vec{x})$, $f_2(\vec{x})$, $g(x_1,\ldots,x_{k-1})$, $h_i(\vec{x},z)$ be PV function symbols that have been introduced in π. Moreover, for $\sigma \in \{1,2\}$, $i \in \{0,1\}$, the equations

$$e'_g : f_1(x_1,\ldots,x_{k-1},\varepsilon) = f_2(x_1,\ldots,x_{k-1},\varepsilon) \tag{4.4}$$
$$e'_{\sigma,i} : f_\sigma(x_1,\ldots,x_{k-1},s_i(x_k)) = h_i(x_1,\ldots,x_{k-1},x_k,f_\sigma(x_1,\ldots,x_k)) \tag{4.5}$$

  are in π. Let n_z be the length of z in h_i. The ATC of e is defined as

$$\mu^+_{e,\pi}(\vec{n}) := c \cdot \left(\beta^+_{e'_g,\pi}(n_1,\ldots,n_{k-1}) + \sum_{\sigma\in\{1,2\}} \sum_{i\in\{0,1\}} n_k \cdot \beta^+_{e'_{\sigma,i},\pi}(n_1,\ldots,n_{k-1},n_k) \right). \tag{4.6}$$

Remark 4.5. The definition of ATC of new proof lines for the induction rule consists of two terms: The first term $\beta^+_{e'_g,\pi}(n_1,\ldots,n_{k-1})$ captures the cost of the proof in the base case

$$f_1(x_1,\ldots,x_{k-1},\varepsilon) = g(\vec{x}) = f_2(x_1,\ldots,x_k,\varepsilon),$$

and the second term

$$\sum_{\sigma\in\{1,2\}} \sum_{i\in\{0,1\}} n_k \cdot \beta^+_{e'_{\sigma,i},\pi}(n_1,\ldots,n_{k-1},n_k)$$

captures the additional cost in the induction case:

$$\begin{aligned}
&f_1(x_1,\ldots,x_{k-1},\sigma_1) = h_{\sigma_1}(x_1,\ldots,x_{k-1},f_i(x_1,\ldots,x_{k-1},\varepsilon)) = f_2(x_1,\ldots,x_{k-1},\sigma_1)\\
&f_1(x_1,\ldots,x_{k-1},\sigma_1\sigma_2) = h_{\sigma_2}(x_1,\ldots,x_{k-1},f_i(x_1,\ldots,x_{k-1},\sigma_1)) = f_2(x_1,\ldots,x_{k-1},\sigma_1\sigma_2)\\
&\vdots\\
&f_1(x_1,\ldots,x_{k-1},\sigma_1\ldots\sigma_{n_k}) = h_{\sigma_{n_k}}(x_1,\ldots,x_{k-1},f_i(x_1,\ldots,x_{k-1},\sigma_1\ldots\sigma_{n_k-1})) = f_2(x_1,\ldots,x_{k-1},\sigma_1\ldots\sigma_{n_k}),
\end{aligned}$$

where $\sigma_1\sigma_2\ldots\sigma_{n_k} = x_k$. Each of these n_k lines requires a copy of the proof of $e'_{\sigma,i}$ in Equation (4.5) on an input length upper bounded by $|x_1| \le n_1,\ldots,|x_{k-1}| \le n_{k-1},|x_k| \le n_k$. Similar to Remark 4.4, there is obvious redundancy as the input length upper bound is not tight — the actual input length to x_k is at most j in the i-th line. Nevertheless, this only incurs a small overhead as we will prove in Theorem 4.4.

We also note that the additional cost is simply for applying the transitivity of equality; it does not include the cost of proving the instances of the equations $e'_{\sigma,i}$ in Equation (4.5), e.g., the first equation

$$f_1(x_1,\ldots,x_{k-1},\sigma_1) = h_{\sigma_1}(x_1,\ldots,x_{k-1},f_i(x_1,\ldots,x_{k-1},\varepsilon)).$$

The cost of proving the equations $e'_{\sigma,i}$ is considered as the additional cost of previous lines.

4.3.2 Acquired Input Lengths

The ATC of an equation or proof line e in the context of a proof π formalizes the *additional cost* to verify or prove the equation e when π is considered "known". To fully verify or prove the equation e from π, we may need to verify or prove one or more lines in π on one or more input lengths. This motivates the definition of acquired input lengths in π to verify or prove e.

Let π be a PV proof consisting of lines $e_1,\ldots,e_m$, and e be a PV equation. We define the set of acquired input lengths (AIL) of e_i by an equation e on the input lengths $\vec{n}$ as $\beta\text{-AIL}_{e,e_i}(\vec{n}) \subseteq \vec{\mathbb{N}}$, and the set of acquired input lengths of e_i by a new proof line e on the input lengths $\vec{n}$ as $\mu\text{-AIL}_{e,e_i}(\vec{n}) \subseteq \vec{\mathbb{N}}$.

AIL of equations. Intuitively, the AIL of a proof line e_i by an equation e is the set of input lengths for e_i we need to obtain to verify the equation e. This only happens if there is a function f in the equation e defined via limited recursion from (g, h_0, h_1, k_0, k_1) such that e_i is $\mathsf{ITR}(h_i(\vec{x}, y, z), z \circ k_i(\vec{x}, y)) = \varepsilon$, i.e., the upper bound of the recursion scheme h_i. The formal definition is as follows.

We first define the AIL of a PV term t, denoted as $\beta\text{-}\mathsf{AIL}_{t,e_i}(\vec{n})$, by structural induction on t. The AIL of an equation $e : t_1 = t_2$ is defined as $\beta\text{-}\mathsf{AIL}_{e,e_i}(\vec{n}) := \beta\text{-}\mathsf{AIL}_{t_1,e_i}(\vec{n}) \cup \beta\text{-}\mathsf{AIL}_{t_2,e_i}(\vec{n})$. In the base case, if t is ε, a variable, or an initial function, $\beta\text{-}\mathsf{AIL}_{t,e_i}(\vec{n}) := \varnothing$.

- (*Functions via Composition*). If t is a function introduced by composition, i.e., $f_{t'}(\vec{x}) = t$ for some term t', we define $\beta\text{-}\mathsf{AIL}_{t,e_i}(\vec{n}) := \beta\text{-}\mathsf{AIL}_{t',e_i}(\vec{n})$.
- (*Functions via Limited Recursion*). Suppose that t is a function $f(\vec{x}, y)$ introduced by limited recursion from $g(\vec{x})$, $h_\sigma(\vec{x}, y, z)$, $k_\sigma(\vec{x}, y)$, $\sigma \in \{0, 1\}$, where $\vec{x} = (x_1, \ldots, x_k)$. Let $\vec{n} = (n_1, \ldots, n_k)$ be the input lengths of $x_1, \ldots, x_k$, and n_y be the input length of y. Let ℓ_f be the bounding value of f, we define

$$\Delta(\vec{n}, n_y) := \beta\text{-}\mathsf{AIL}_{g,e_i}(\vec{n}) \cup \beta\text{-}\mathsf{AIL}_{h_0,e_i}(\vec{n}, n_y, \ell_f(\vec{n}, n_y)) \cup \beta\text{-}\mathsf{AIL}_{h_1,e_i}(\vec{n}, n_y, \ell_f(\vec{n}, n_y)). \tag{4.7}$$

  If e_i is not $\mathsf{ITR}(h_\sigma(\vec{x}, y, z), z \circ k_\sigma(\vec{x}, y)) = \varepsilon$ for some $\sigma \in \{0, 1\}$, $\beta\text{-}\mathsf{AIL}_{t,e_i}(\vec{n}, n_y) := \Delta(\vec{n}, n_y)$. Otherwise,

$$\beta\text{-}\mathsf{AIL}_{t,e_i}(\vec{n}, n_y) := \Delta(\vec{n}, n_y) \cup \{(\vec{n}, n_y, \ell_f(\vec{n}, n_y))\},$$

  where the tuple $(\vec{n}, n_y, \ell_f(\vec{n}, n_y))$ means that the acquired input lengths of $\vec{x}$, y, and z in the equation $e_i : \mathsf{ITR}(h_\sigma(\vec{x}, y, z), z \circ k_\sigma(\vec{x}, y)) = \varepsilon$ are $\vec{n}$, n_y, and $\ell_f(\vec{n}, n_y)$, respectively.
- (*Composition*). Suppose that t is of form $f(s_1, \ldots, s_j)$ for a function f and terms $s_1, \ldots, s_j$. The AIL of t is defined as

$$\beta\text{-}\mathsf{AIL}_{t,e_i}(\vec{n}) := \beta\text{-}\mathsf{AIL}_{f,e_i}(\vec{n}) \cup \bigcup_{j' \in [j]} \beta\text{-}\mathsf{AIL}_{s_{j'},e_i}(\vec{n}). \tag{4.8}$$

AIL of new proof lines. Suppose that e is a valid new proof line after π, we will define the AIL of e_i by e on the input lengths $\vec{n}$, denoted as $\mu\text{-}\mathsf{AIL}_{e,e_i}(\vec{n})$, by considering what axiom or deduction rule is used to introduce e.

- (*Axioms*). If e is introduced by axioms, including axioms for initial functions and axioms for introduced functions, $\mu\text{-}\mathsf{AIL}_{e,e_i}(\vec{n}) := \beta\text{-}\mathsf{AIL}_{e,e_i}(\vec{n})$.
- (*Function Introduction*). If e is to introduce a function symbol via the composition or the limited recursion rule, $\mu\text{-}\mathsf{AIL}_{e,e_i}(\vec{n}) := \varnothing$.
- (*Logical Rules*). Suppose that e is derived from a logical rule. If e_i is not a premise of the logical rule, we define $\mu\text{-}\mathsf{AIL}_{e,e_i}(\vec{n}) := \beta\text{-}\mathsf{AIL}_{e,e_i}(\vec{n})$.
  - If $e : t_1 = t_2$ is derived from (L1), (L2), or (L3), and e_i is one of the premises, we define $\mu\text{-}\mathsf{AIL}_{e,e_i}(\vec{n}) := \beta\text{-}\mathsf{AIL}_{e,e_i}(\vec{n}) \cup \{\vec{n}\}$.
  - Suppose that $e : t_1[x_j/v] = t_2[x_j/v]$ is derived from (L4) and $e_i : t_1 = t_2$ is the premise, where v is a term. Let $\ell_v(\vec{n})$ be the bounding value of the term v and $\vec{n}' := (n_1, \ldots, n_{j-1}, \ell_v(\vec{n}), n_{j+1}, \ldots, n_k)$. We define $\mu\text{-}\mathsf{AIL}_{e,e_i}(\vec{n}) := \beta\text{-}\mathsf{AIL}_{e,e_i}(\vec{n}) \cup \{\vec{n}'\}$.
- (*Induction*). Suppose that e is derived from the induction rule. Without loss of generality, we assume that x_k is the induction variable. Let $f_1(\vec{x})$, $f_2(\vec{x})$, $g(x_1, \ldots, x_{k-1})$, $h_i(\vec{x}, z)$ be PV function symbols that have been introduced in π. Moreover, for $\sigma \in \{1, 2\}$, $j \in \{0, 1\}$, the equations

$$e'_g : f_1(x_1, \ldots, x_{k-1}, \varepsilon) = f_2(x_1, \ldots, x_{k-1}, \varepsilon)$$
$$e'_{\sigma,j} : f_\sigma(x_1, \ldots, x_{k-1}, s_j(x_k)) = h_j(x_1, \ldots, x_{k-1}, x_k, f_\sigma(x_1, \ldots, x_k))$$

  are in π. Consider the following cases.

- If e_i is e'_g, we define $\mu\text{-AIL}_{e,e_i}(\vec{n}) := \beta\text{-AIL}_{e,e_i}(\vec{n}) \cup \{(n_1, \ldots, n_{k-1}, 0)\}$.
- If e_i is $e'_{\sigma,j}$, we define

$$\mu\text{-AIL}_{e,e_i}(\vec{n}) := \beta\text{-AIL}_{e,e_i}(\vec{n}) \cup \{(n_1, \ldots, n_{k-1}, n_k)\}.$$

- Otherwise, we define $\mu\text{-AIL}_{e,e_i}(\vec{n}) := \beta\text{-AIL}_{e,e_i}(\vec{n})$.

Acquisition closure. Let $\pi : e_1, \ldots, e_m$ be a PV proof. A set S is said to be an acquisition map if each element in S is a pair $(e_j, \vec{n}')$, where e_j is the j-th proof line of π, and let $x_1, \ldots, x_{k_j}$ be the variables in e_j, then $\vec{n}' = (n'_1, \ldots, n'_{k_j}) \in \mathbb{N}$.

We define the acquisition extension of the acquisition map S: For a pair $(e_{j_1}, \vec{m}'_1)$ in the set, $j_2 < j_1$, and $\vec{m}'_2 \in \mu\text{-AIL}_{e_{j_1},e_{j_2}}(\vec{m}'_1)$, we add $(e_{j_2}, \vec{m}'_2)$ in to the set. The *acquisition closure* of S, denoted by $\mu\text{-AM}_\pi(S)$, is defined as the minimal set containing S that is closed under acquisition extension.

4.3.3 Time Complexity of Proofs and Equations

We are finally ready to define the time complexity of proofs and equations by taking the summation over the time cost (i.e. the ATC) of all equations or proof lines on all acquired input lengths.

Definition 4.6 (time complexity of proofs)**.** Let π be a PV proof with proof lines $e_1, \ldots, e_m$, where e_m is the conclusion of π. The time complexity (TC) of the proof π concluding e_m, denoted by $\mu^{\mathsf{TC}}_\pi(\vec{n})$, is defined as

$$\mu^{\mathsf{TC}}_\pi(\vec{n}) := \sum_{(e_i,\vec{l}) \in \mu\text{-AM}_\pi(\{(e_m,\vec{n})\})} \mu^+_{e_i,\pi_{<i}}(\vec{l}), \tag{4.9}$$

where $\pi_{<i} := e_1, \ldots, e_{i-1}$.

Definition 4.7 (time complexity of equations)**.** Let e be a PV equation and π be a PV proof (called the *context*) such that all function symbols in e are introduced in π. Let $e_1, \ldots, e_m$ are the proof lines in π. The time complexity (TC) of the equation e in the context of π, denoted by $\beta^{\mathsf{TC}}_{e,\pi}(\vec{n})$, is defined as

$$\beta^{\mathsf{TC}}_{e,\pi}(\vec{n}) = \beta^+_{e,\pi}(\vec{n}) + \sum_{(e_i,\vec{l}) \in \mu\text{-AM}_\pi(\beta\text{-AIL}^*_{e,\pi}(\vec{n}))} \mu^+_{e_i,\pi_{<i}}(\vec{l}), \tag{4.10}$$

where $\pi_{<i} := e_1, \ldots, e_{i-1}$, and $\beta\text{-AIL}^*_{e,\pi}(\vec{n})$ is defined as

$$\beta\text{-AIL}^*_{e,\pi}(\vec{n}) := \bigcup_{i \le m} \left\{ (e_i, \vec{l}) \mid \vec{l} \in \beta\text{-AIL}_{e,e_i}(\vec{n}) \right\}.$$

Remark 4.8. We note that Equation (4.10) formalizes the intuition mentioned in Section 1.1.4 and Section 2.2. The first term considers the size of the propositional translation of e, while the second term considers the proof complexity of all lines in the context π, in which we define the function symbols used in the equation e and prove any necessary properties. As the context π will present in any proof π' of e, the second term of Equation (4.10) will necessarily contribute to $\mu^{\mathsf{TC}}_{\pi'}(\vec{n})$ as defined in Equation (4.9).

Remark 4.9. We emphasize that the definition is the same when there is no variables. In such case, the input length $\vec{n}$ is replaced by a placeholder $\varnothing$, where $|\varnothing| := 0$, and the ATC $\beta^+_{e,\pi}(\varnothing)$ will be a constant. We also note that even if $\vec{n} = \varnothing$, the acquisition map $\mu\text{-AM}_\pi(\{(e_m, \varnothing\})$ may contain pairs $(e_i, \vec{l})$ for $\vec{l} \neq \varnothing$, as the intermediate step e_i may have variables.

4.4 Connection to Cook Complexity

Now we show that the complexity measures $\mu_\pi^{\mathsf{TC}}(\vec{n})$ and $\beta_{e,\pi}^+(\vec{n})$ are Cook complexity measures for a fixed propositional translation. That is, there is a propositional translation $[\cdot]_{\mathsf{Cook}}$ such that if we set the formalization overhead c to be sufficiently large, $\mu_\pi^{\mathsf{TC}}(\vec{n}) = |[\pi]_{\mathsf{Cook}}^{\vec{n}}|$ and $\beta_{e,\pi}^+ = |[e]_{\mathsf{Cook}}^{\vec{n}}|$. Formally:

Theorem 4.3. *There is a constant $c \geq 1$ and a fixed propositional translation $[\cdot]_{\mathsf{Cook}}$ such that $\mu_\pi^{\mathsf{TC}}(\vec{n}) = |[\pi]_{\mathsf{Cook}}^{\vec{n}}|$ and $\beta_{e,\pi}^{\mathsf{TC}}(\vec{n}) \geq \beta_{e,\pi}^+(\vec{n}) = |[e]_{\mathsf{Cook}}^{\vec{n}}|$ when the formalization overhead is chosen to be c.*

Proof. Let c be a constant to be determined later. We will describe the propositional translation $[\cdot]_{\mathsf{Cook}}$ of equations and proof and specify the constant c at the same time. Note that we will prove that $\mu_\pi^{\mathsf{TC}}(\vec{n}) \geq |[\pi]_{\mathsf{Cook}}^{\vec{n}}|$ and $\beta_{e,\pi}^+(\vec{n}) \geq |[e]_{\mathsf{Cook}}^{\vec{n}}|$, as we can pad the propositional translation.

Translation to propositional formulas. We first verify that $\beta_{e,\pi}^+(\vec{n}) \geq |[e]_{\mathsf{Cook}}^{\vec{n}}|$. Indeed, we will describe an algorithm $\mathsf{GenProp}_{\pi,e}(\vec{n})$ such that for every valid proof π and equation e in the context, any input length $\vec{n}$ for $\vec{x}$, it produces a propositional formula that is a tautology if and only if $\forall x\ t_1(\vec{x}) = t_2(\vec{x})$.

Specifically, we first translate the terms $t_1(\vec{x})$ and $t_2(\vec{x})$ on input length $\vec{n}$ to propositional formulas; in more detail, for each term $t \in \{t_1, t_2\}$, the propositional translation $[t]_{\mathsf{Cook}}^{\vec{n}}$ of t is defined by introducing variables and constraints as follows:

- (*Input Variables*). We introduce $n_1 + \cdots + n_k + k$ variables $x_{1,0}, x_{1,1}, \ldots, x_{1,n_1}, \ldots, x_{k,0}, x_{k,1}, \ldots, x_{k,n_k}$ encoding the input $x_1, \ldots, x_k$. That is, for each $j \in [k]$, $x_{j,0}, \ldots, x_{j,n_j} \in \{0,1\}$ encodes a string of length at most n_j that is supposed to substitute the variable x_j.
- (*Output Variables*). We introduce $\ell_t(\vec{n})$ variables encoding the output of the term $t(\vec{x})$, where $\ell_t(\vec{n})$ is the bounding value of t.
- (*Internal Variables and Constraints*). We introduce internal variables and constraints to ensure that given any input $\vec{x}$ encoded by the input variables:
  - if all constraints are satisfied, the output variables must encode $t(\vec{x})$;
  - there is a satisfying assignment such that the output variables encode $t(\vec{x})$.

  These variables and constraints are defined by induction on the structure of t. Without loss of generality, we may assume that constraints are represented by a 3-CNF.

Next, we introduce additional variables and constraints to compare the output of $t_1(\vec{x})$ and $t_2(\vec{x})$. In more detail, these variables and constraints encodes the following statement:

- $(\star)$: If t_1 and t_2 are given the same input, and all constraints for the internal computation of t_1 and t_2 are satisfied, the output variables of t_1 and t_2 must be the same.

We can see that $\forall \vec{x}\ t_1(\vec{x}) = t_2(\vec{x})$ if and only if the statement above is a tautology.

Note that the statement $(\star)$ can be formalized as a formula of form $\phi \to \psi$, where ϕ is a 3-CNF encoding the constraints to input and the internal computation of t_1 and t_2, and ψ is a 3-CNF encoding the comparison to the output variables of t_1 and t_2. The generation of ψ, so it suffices to generate ϕ, which, in turn, reduces to the problem of generating $[t]_{\mathsf{Cook}}^{\vec{n}}$.

The algorithm is defined by structural induction on $[t]_{\mathsf{Cook}}^{\vec{n}}$. Meanwhile, we will prove that $[t]_{\mathsf{Cook}}^{\vec{n}}$ is a formula of size at most $\beta_{t,\pi}^+(\vec{n})$.

- (*Base Case*). If t is ε, x, $s_0(x)$, $s_1(x)$, or $\mathsf{TR}(x)$, we can generate an explicit linear-size formula corresponding to the computation of t. The size is bounded by $\beta_{t,\pi}^+(\vec{n}) = c \cdot n$ if the formalization overhead c is chosen as a sufficiently large constant. Similarly, if t is $\mathsf{ITR}(x,y)$ or $\circ(x,y)$, we can generate an explicit $O(|x| \cdot |y|)$-size formula and the size is bounded by $\beta_{t,\pi}^+(\vec{n}) = c \cdot |x| \cdot |y|$ if the formalization overhead c is chosen as a sufficiently large constant.

- (*Functions via Composition*). If t is a function introduced by composition, i.e., $f_{t'}(\vec{x}) = t(\vec{x})$ for some valid term t' in the context of π, $[t]^{\vec{n}}_{\mathsf{Cook}}$ is simply defined as $[t']^{\vec{n}}_{\mathsf{Cook}'}$, and by induction hypothesis, it is of size at most $\beta^+_{t',\pi}(\vec{n}) = \beta^+_{t,\pi}(\vec{n})$.

- (*Functions via Limited Recursion*). If t is a function $f(\vec{x})$ introduced by limited recursion from g, h_i, k_i, $i \in \{0,1\}$. Without loss of generality, assume that x_k is the recursion variable, $\vec{x} = (x_1, \ldots, x_k)$, and $n_1, \ldots, n_k$ denotes the input length of $x_1, \ldots, x_k$. The formula $[t]^{\vec{n}}_{\mathsf{Cook}}$ consists of the following components.

  (i) the translation of g on the input length $n_1, \ldots, n_{k-1}$;
  (ii) n_k copies of the translation of $h_\sigma(\vec{x}, z)$ on the input length $n_1, \ldots, n_k, |z| = \ell_f(n_1, \ldots, n_k)$;
  (iii) corresponding intermediate variables such that the n_k copies of the translation of $h_\sigma(\vec{x}, z)$ simulate the following computation:

$$\begin{aligned} z_0 &:= g(x_1, \ldots, x_{k-1}) = f(x_1, \ldots, x_{k-1}, \varepsilon) \\ z_1 &:= h_{\sigma_1}(x_1, \ldots, x_{k-1}, \varepsilon, z_0) = f(x_1, \ldots, x_{k-1}, \sigma_1) \\ &\vdots \quad \vdots \\ z_{n_k} &:= h_{\sigma_{n_k}}(x_1, \ldots, x_{k-1}, \sigma_1\sigma_2 \ldots \sigma_{n_k-1}, z_{n_k-1}) = f(x_1, \ldots, x_{k-1}, \sigma_1\sigma_2 \ldots \sigma_{n_k}) \end{aligned} \tag{4.11}$$

  where $x_k = \sigma_1\sigma_2 \ldots \sigma_{n_k}$, and each z_i is of length $\ell_f(x_1, \ldots, n_k)$.

  By the induction hypothesis, the first part is of size $\beta^+_{g,\pi}(n_1, \ldots, n_{k-1})$, and the second part is of size $n_k \cdot \beta^+(n_1, \ldots, n_k, \ell_f(n_1, \ldots, n_k))$. It can be verified that the third part incurs a linear-size overhead. Therefore, the overall size of the formula is bounded by

$$O(\beta^+_{g,\pi}(n_1, \ldots, n_{k-1}) + n_k \cdot \beta^+(n_1, \ldots, n_k, \ell_f(n_1, \ldots, n_k))) \le \beta^+_{t,\pi}(\vec{n})$$

  if c is sufficiently large.

- (*Composition*). If t is of form $f(s_1, \ldots, s_m)$ for a function f and terms $s_1, \ldots, s_i$, its propositional translation consists of the following components:

  (i) for each $s \in \{s_1, \ldots, s_j\}$, the propositional translation of s on the input length $\vec{x}$;
  (ii) the propositional translation of $f(z_1, \ldots, z_j)$, where $|z_1| = \ell_{s_1}(\vec{n}), \ldots, |z_j| = \ell_{s_j}(\vec{n})$;
  (iii) corresponding intermediate variables that ensure the output variables of $s_1, \ldots, s_j$ are equal to the input variables of $f(z_1, \ldots, z_j)$.

  By the induction hypothesis, the first part is of size $\beta^+_{s,\pi}(\vec{n})$ for each $s \in \{s_1, \ldots, s_j\}$, and the second part is of size $\beta^+_{f,\pi}(\ell_{s_1}(\vec{n}), \ldots, \ell_{s_j}(\vec{n}))$. It can be verified that the third part incurs a linear-size overhead, and therefore the overall size of the formula is at most

$$O\left(\beta^+_{f,\pi}(\ell_{s_1}(\vec{n}), \ldots, \ell_{s_i}(\vec{n})) + \sum_{j \in [i]} \beta^+_{s_i,\pi}(\vec{n})\right) \le \beta^+_{t,\pi}(\vec{n})$$

  if c is sufficiently large.

In summary, for any equation $e : t_1 = t_2$ and its propositional translation $\phi \to \psi$, the formula ϕ is of size at most $\beta^+_{t_1,\pi}(\vec{n}) + \beta^+_{t_2,\pi}(\vec{n})$, and the formula ψ is of linear size in the output length of t_1 and t_2. The total size of $\phi \to \psi$ is at most

$$O(\beta^+_{t_1,\pi}(\vec{n}) + \beta^+_{t_2,\pi}(\vec{n}) + \ell_{t_1}(\vec{x}) + \ell_{t_2}(\vec{x})) \le \beta^+_{e,\pi}(\vec{n})$$

when c is sufficiently large.

Translation to EF proofs. Next, we verify that $\mu_\pi^{\mathsf{TC}}(\vec{n}) \geq \mu_\pi^{\mathsf{Cook}}(\vec{n})$ by proposing an algorithm $\mathsf{GenProof}_\pi(\vec{n})$ such that for every proof π concluding e, it outputs an EF proof of $[e]^{\vec{n}}_{\mathsf{Cook}}$ of size $\mu_\pi^{\mathsf{TC}}(\vec{n})$.

Let $\pi : e_1, \ldots, e_m$, $\Delta = \mu\text{-}\mathsf{AM}_\pi(\{(e_m, \vec{n})\})$ be the acquisition map, and $\Delta_i := \{(e_i, \vec{p}) \in \Delta \mid \vec{p} \in \mathbb{N}\}$. In the i-th round, our algorithm will generate EF proofs of $[e_i]^{\vec{p}}_{\mathsf{Cook}}$ for each $\vec{p} \in \Delta_i$, and we will verify that the total length of new proofs lines added for $\vec{p} \in \Delta_i$ in the i-th round is at most $\mu^+_{e_i,\pi_{<i}}(\vec{p})$, where $\pi_{<i} := e_1, \ldots, e_{i-1}$.

Now fix any $i \in [m]$ and $\vec{p} \in \Delta_i$. The algorithm considers the deduction rule that introduces the proof line e_i after $\pi_{<i} = e_1, \ldots, e_{i-1}$.

- (*Axioms for Functions*). Suppose that e_i is a proof line introduced as an axiom for the functionality of initial functions or introduced functions. It can be verified that in such cases, there is an explicit EF proof of $[e_i]^{\vec{p}}_{\mathsf{Cook}}$, and the size of the proof is linear in the size of $[e_i]^{\vec{p}}_{\mathsf{Cook}}$. The algorithm outputs the EF proof of size at most $O(\beta^+_{e_i,\pi}(\vec{p})) \leq c \cdot \beta^+_{e_i,\pi}(\vec{p}) = \mu^+_{e_i,\pi}(\vec{p})$, where the inequality holds if c is a sufficiently large constant.

- (*Function Introduction*). Suppose that e_i is a proof line that introduces a function symbol by the composition rule or the limited recursion rule, it is not an equation and thus does not have a propositional translation.

- (*Logical Rules*). Suppose that e_i is derived from a logical rule. Assume that the rule is (L1), i.e., derive $e_i : t = s$ from $e_j : s = t$ for some $j < i$. We know by the definition of AILs and acquisition closure that $\vec{p} \in \Delta_j$, as $\vec{p} \in \Delta_i$ and $\{\vec{p}\} \in \mu\text{-}\mathsf{AIL}_{e_i,e_j}(\vec{p})$. Subsequently, by the induction hypothesis, we have already generated the EF proof of $[e_j]^{\vec{p}}_{\mathsf{Cook}}$ in the j-th round. It can be verified that from $[e_j]^{\vec{p}}_{\mathsf{Cook}}$, we can prove $[e_i]^{\vec{p}}_{\mathsf{Cook}}$ in EF, and the size of the EF proof is linear in the size of $[e_i]^{\vec{p}}_{\mathsf{Cook}}$. In such case, the new proofs lines are of length at most $O(\beta^+_{e_i,\pi}(\vec{p})) \leq c \cdot O(\beta^+_{e_i,\pi}(\vec{p})) \leq \mu^+_{e_i,\pi}(\vec{p})$.

  The proof is similar if the rule is (L2), (L3), or (L4) — it can be verified that from the existing EF proofs generated in previous rounds, we can prove the propositional translation of e_i on the input length $\vec{p}$ in EF, and the total length of new proof lines is at most $\mu^+_{e_i,\pi}(\vec{p})$.

- (*Induction*). Suppose that e_i is derived from the induction rule. Without loss of generality, we assume that x_k is the induction variable and $\vec{x} = (x_1, \ldots, x_k)$. Let $f_1(\vec{x}), f_2(\vec{x}), g(x_1, \ldots, x_{k-1}), h_i(\vec{x}, z)$ be PV function symbols that have been introduced in $\pi_{<i}$. Moreover, for $\sigma \in \{1, 2\}$, $b \in \{0, 1\}$, the equations

$$e'_g : f_1(x_1, \ldots, x_{k-1}, \varepsilon) = f_2(x_1, \ldots, x_{k-1}, \varepsilon)$$
$$e'_{\sigma,b} : f_\sigma(x_1, \ldots, x_{k-1}, s_b(x_k)) = h_b(x_1, \ldots, x_{k-1}, x_k, f_\sigma(x_1, \ldots, x_k))$$

  are in $\pi_{<i}$. Note that as $\vec{p} \in \Delta_i$, by the definition of AILs and the induction hypothesis, we know that the algorithm has produced EF proofs of e'_g on the input length $(p_1, \ldots, p_{k-1})$ and $e'_{\sigma,i}$ on the input length $\vec{p}$. It suffices to prove in EF the propositional translation of e on the input length $\vec{p}$ from e'_g and $e'_{\sigma,i}$, and the total length of the new proof lines is at most $\mu^+_{e,\pi}(\vec{p})$.

  The EF proof is as follows:

  - For each $j \in \{0, 1, \ldots, p_k\}$, we introduce auxiliary variables $z_{j,1}, \ldots, z_{j,\ell}$, where $\ell := \ell_f(\vec{p})$.
  - For each $j \in \{0, 1, \ldots, p_k\}$, let ϕ_j be a 3-CNF that includes the constraints such that if ϕ_j is satisfied, then
$$z_j = \begin{cases} f_1(x_1, \ldots, x_{k-1}, \varepsilon) & \text{if } i = 0; \\ h_{\sigma_i}(x_1, \ldots, x_k, z_{j-1}) & \text{otherwise.} \end{cases}$$
    where σ_i is the i-th leftmost bit of x_k.

– For each $j \in \{0, 1, \ldots, p_k\}$, let ψ_j be a 3-CNF that is satisfied if and only if

$$z_j = f_1(x_1, \ldots, x_{k-1}, x_{k,\leq j}) = f_2(x_1, \ldots, x_{k-1}, x_{k,\leq j}),$$

where $x_{k,\leq j}$ denotes the prefix of x_k of length j. We prove that $\alpha_j := \phi_1 \wedge \cdots \wedge \phi_j \to \psi_j$. The case for α_0 follows from the proof of the propositional translation of e'_g, and the case for α_{j+1} follows from the case for α_j and the proof of the propositional translation of e'_{1,x_j} and e'_{2,x_j}. We note that e'_{1,x_j} and e'_{2,x_j} might be an overkill as the input length x_k for ψ_j is j, while e'_{1,x_j} and e'_{2,x_j} work when x_k is of length at most $p_k \geq j$. Nevertheless, this is not an issue as we also assumed such relaxation when we define the ATC of proofs (see Remark 4.5).

The total length of new proof lines to prove α_j is linear in the size of the formula α_j, which is in turn at most $O(\beta^+_{e'_g,\pi}(p_1, \ldots, p_{k-1}))$ for $j = 0$, or $O(\beta^+_{e'_{1,x_j}}(p_1, \ldots, p_k, \ell) + \beta^+_{e'_{1,x_j}}(p_1, \ldots, p_k, \ell))$ when $j \geq 1$. Therefore, the total length of new proof lines is bounded by

$$O(\beta^+_{e'_g,\pi}(p_1, \ldots, p_{k-1})) + O(\beta^+_{e'_{1,x_j}}(p_1, \ldots, p_k, \ell) + \beta^+_{e'_{1,x_j}}(p_1, \ldots, p_k, \ell)) \leq \mu^+_{e,\pi}(\vec{p}),$$

where the inequality holds when c is sufficiently large.

Since $\vec{n} \in \Delta_m$, in the m-th round, the algorithm will output an EF proof of $[e_m]^{\vec{n}}_{\mathsf{Cook}}$, and the total size of the proof is at most

$$\sum_{(e_i,\vec{p})\in\Delta} \mu^+_{e_i,\pi_{<i}}(\vec{p}) = \mu^{\mathsf{TC}}_\pi(\vec{n}).$$

This completes the proof. □

4.5 Redundancy in the Definition of Time Complexity

Theorem 4.3 shows that the time complexity of equations and proofs are upper bounds of Cook complexity for a specific propositional translation. This does not rule out the possibility that there are much better propositional translations under which the Cook complexity is much smaller than the time complexity. In particular, as mentioned in Remarks 4.4 and 4.5, we indeed relax the upper bound to ensure that the computation of the time complexity is efficient.

In this subsection, we show that the relaxation in Remarks 4.4 and 4.5 does not incur a super-linear overhead. Let $\mu'^{\mathsf{TC}}_\pi(\vec{n})$ and $\beta'^{\mathsf{TC}}_{e,\pi}(\vec{n})$ be the complexity measure following the same definition as the time complexity, but with the following two modification:

- In the ATC of equations: Suppose that $f(\vec{u}, v)$ is a function introduced by limited recursion from $g(\vec{u}), h_i(\vec{u}, v, z), k_i(\vec{u}, v)$, $i \in \{0, 1\}$, $\vec{u} = (u_1, \ldots, u_i)$, v are variables. Let $m_1, \ldots, m_i, m_v$ be the input length of $u_1, \ldots, u_i, v$, respectively. We define the ATC of f as

$$\begin{aligned}&\beta'^+_{f,\pi}(m_1, \ldots, m_i, m_v)\\ &:= c \cdot \left(\beta'^+_{g,\pi}(m_1, \ldots, m_i) + \sum_{j=0}^{m_v-1} \sum_{\sigma\in\{0,1\}} \beta'^+_{h_\sigma,\pi}(m_1, \ldots, m_i, m_v, \ell_f(m_1, \ldots, m_i, j))\right).\end{aligned} \tag{4.12}$$

- In the ATC of proof lines: Suppose that e is derived from the induction rule. Without loss of generality, we assume that x_k is the induction variable. Let $f_1(\vec{x})$, $f_2(\vec{x})$, $g(x_1, \ldots, x_{k-1})$, $h_i(\vec{x}, z)$ be PV function symbols that have been introduced in π. Moreover, for $\sigma \in \{1, 2\}$, $i \in \{0, 1\}$, the equations

$$e'_g : f_1(x_1, \ldots, x_{k-1}, \varepsilon) = f_2(x_1, \ldots, x_{k-1}, \varepsilon) \tag{4.13}$$

$$e'_{\sigma,i} : f_\sigma(x_1, \ldots, x_{k-1}, s_i(x_k)) = h_i(x_1, \ldots, x_{k-1}, x_k, f_\sigma(x_1, \ldots, x_k)) \tag{4.14}$$

are in π. Let n_z be the length of z in h_i. The ATC of e is defined as

$$\mu'^{+}_{e,\pi}(\vec{n}) := c \cdot \left(\beta'^{+}_{e'_g,\pi}(n_1,\ldots,n_{k-1}) + \sum_{j=0}^{n_k-1} \sum_{\sigma\in\{1,2\}} \sum_{i\in\{0,1\}} \beta'^{+}_{e'_{\sigma,i},\pi}(n_1,\ldots,n_{k-1},j) \right). \tag{4.15}$$

Moreover, we choose the constant $c = 1$ is the definition of μ'^{TC}_{π} and $\beta'^{\mathsf{TC}}_{e,\pi}$.

As we removed the relaxation in Remarks 4.4 and 4.5 and set the constant $c = 1$, it can be verified that μ'_π is a size lower bound of the propositional translation of π (using $[\cdot]_{\mathsf{Cook}}$ from the proof of Theorem 4.3), and $\beta'^{+}_{e,\pi}$ is a size lower bound of the propositional translation of e (using $[\cdot]_{\mathsf{Cook}}$ from the proof of Theorem 4.3). We will prove that $\mu'^{\mathsf{TC}}_{\pi}(\vec{n}), \beta'^{\mathsf{TC}}_{e,\pi}(\vec{n})$ are equivalent to $\mu^{\mathsf{TC}}_{\pi}(\vec{n}), \beta^{\mathsf{TC}}_{e,\pi}(\vec{n})$ asymptotically.

Theorem 4.4. *For any proof π and equation e in the context π, $\mu^{\mathsf{TC}}_{\pi}(\vec{n}) = O_\pi(\mu'_\pi(\vec{n}))$ and $\beta^{\mathsf{TC}}_{e,\pi}(\vec{n}) = O_{\pi,e}(\beta'_{e,\pi}(\vec{n}))$.*

Proof Sketch. We only sketch the proof of $\beta^{\mathsf{TC}}_{e,\pi}(\vec{n}) = O(\beta'_{e,\pi}(\vec{n}))$. Indeed, as the two measures share the same definition of AILs, it suffices to prove that for any term $t(\vec{n})$ in the context π, $\beta^{+}_{t,\pi}(\vec{n}) = O(\beta'^{+}_{t,\pi}(\vec{n}))$.

We prove this by induction on t. The base case is trivial, as the two measures share the same definition of the ATC of variables, ε, and initial functions. Suppose that t is not a function introduced by limited recursion on notation, $\beta^{+}_{t,\pi}(\vec{n}) = O(\beta'^{+}_{t,\pi}(\vec{n}))$ follows directly from the induction hypothesis.

Suppose that t is a function $f(\vec{u}, v)$ introduced by limited recursion from g, h_0, h_1, k_0, k_1, where $m_1,\ldots,m_i$ denotes the input length of $\vec{u}$, and m_v denotes the input length of v. Then we can see that

$$\begin{aligned}
&\beta^{+}_{t,\pi}(m_1,\ldots,m_i,m_v) \\
&= O\left(\beta^{+}_{g,\pi}(m_1,\ldots,m_i) + m_v \cdot \sum_{\sigma\in\{0,1\}} \beta^{+}_{h_\sigma,\pi}(m_1,\ldots,m_i,m_v,\ell_f(m_1,\ldots,m_i,m_v)) \right) \\
&\le O\left(\beta'^{+}_{g,\pi}(m_1,\ldots,m_i) + m_v \cdot \sum_{\sigma\in\{0,1\}} \beta'^{+}_{h_\sigma,\pi}(m_1,\ldots,m_i,m_v,\ell_f(m_1,\ldots,m_i,m_v)) \right) \\
&\le O\left(\beta'^{+}_{g,\pi}(m_1,\ldots,m_i) + \frac{m_v}{2} \sum_{\sigma\in\{0,1\}} \beta'^{+}_{h_\sigma,\pi}\left(m_1,\ldots,m_i,m_v,\ell_f\left(m_1,\ldots,m_i,\frac{m_v}{2}\right)\right) \right) \\
&\le O\left(\beta'^{+}_{g,\pi}(m_1,\ldots,m_i) + \sum_{j=m_v/2}^{m_v-1} \sum_{\sigma\in\{0,1\}} \beta'^{+}_{h_\sigma,\pi}\left(m_1,\ldots,m_i,m_v,\ell_f\left(m_1,\ldots,m_i,j\right)\right) \right) \\
&\le O(\beta'^{+}_{t,\pi}(m_1,\ldots,m_i,m_v)).
\end{aligned}$$

Here, the constants hidden in $O(\cdot)$ may depend on the proof π, e as well as the formalization overhead c. The second inequality follows from the induction hypothesis, the third inequality follows from the fact that both ℓ_f and $\beta'^{+}_{h_\sigma,\pi}$ are polynomials in $m_1,\ldots,m_i,m_v$, and fourth inequality follows from the fact that the bounding value ℓ_f is a monotone function. □

4.6 Time Complexity and Length of Proof

As a sanity check, we show that time complexity is not a trivial measure based on the length of proofs by showing that the time complexity of proofs of the same length could vary drastically.

Proposition 4.5. *For $k \in \mathbb{N}$, there is a* PV *proof of length $O(k)$ that is of time complexity $O(k \cdot n)$.*

Proof. Let $s_0^{(k)}(x)$ be a shorthand recursively defined as

$$s_0^{(0)}(x) = x; \quad s_0^{(k+1)}(x) = s_0(s_0^{(k)}(x)).$$

We will show that there is a constant c independent of k such that the equation $e_k : x \circ s_0^{(k)}(\varepsilon) = s_0^{(k)}(x)$ admits a PV proof of length $c \cdot k$ that is of time complexity $\mu(n) \leq c \cdot k \cdot n$. Note that the size of the equation e_k is $O(1)$, so it suffices to consider the number of lines in the proof.

We prove this by induction on k in the meta-theory, where the base case is trivial. In the induction case $k \geq 1$, we know by the induction hypothesis that e_{k-1} admits a PV proof of length $c \cdot (k-1)$ that is of time complexity $c \cdot (k-1) \cdot n$. Notice that the following PV proof concludes e_k:

1. (*Induction hypothesis*): $e_{k-1} : x \circ s_0^{(k-1)}(\varepsilon) = s_0^{(k-1)}(x)$
2. (*L3 from 1*): $s_0(x \circ s_0^{(k-1)}(\varepsilon)) = s_0^{(k)}(x)$
3. (*Definition Axiom of* $\circ$): $x \circ s_0(y) = s_0(x \circ y)$.
4. (*L4 from 3*) $x \circ s_0^{(k)}(\varepsilon) = s_0(x \circ s_0^{(k-1)}(\varepsilon))$.
5. (*L2 from 2, 4*): $x \circ s_0^{(k)}(\varepsilon) = s_0^{(k)}(x)$.

Note that the length of the proof is $c \cdot (k-1) + O(1)$, which is at most $c \cdot k$ for sufficiently large c. Now we calculate the time complexity of the new proof. Note that by the definition, we can see that the set of acquired input lengths of each part of the proof is $\{n\}$. The time complexity of the proof is therefore the sum of two terms:

- The time complexity of the proof of e_{k-1} on the input length n, which is $\leq c \cdot (k-1) \cdot n$.
- The ATC of the five new proofs lines, which is $O(n)$.

Therefore, the time complexity of the proof is at most $c \cdot (k-1) \cdot n + O(n)$, which is at most $c \cdot k \cdot n$ when c is sufficiently large. This completes the induction proof. □

Proposition 4.6. *For $k \in \mathbb{N}$, there is a* PV *proof of length $O(k)$ that is of time complexity $\Omega(n^{2^k})$.*

Proof. Let $\#(x, y)$ be the function defined as

$$x \# \varepsilon := \varepsilon, \quad x \# s_i(y) := x \circ (x \# y)$$

by the rule of limited recursion on notation. That is, $x \# y$ outputs the string obtained by concatenating $|y|$ copies of x. Let $f_0(x) := x \# x$, $f_1(x) := f_0(x) \# f_0(x)$, ..., $f_k(x) := f_{k-1}(x) \# f_{k-1}(x)$. Note that these functions can be defined in a proof of length $O(k)$ by the composition rule. We conclude with a proof line $f_k(x) = f_{k-1}(x) \# f_{k-1}(x)$ by the definition axiom of f_k.

Note that by Theorem 4.3, the time complexity of the proof is at least the Cook complexity of the equation $f_k(x) = f_{k-1}(x) \# f_{k-1}(x)$, which is in turn at least the computational complexity of the function $f_k(x)$. By induction on k, we can prove that the output length of $f_k(x)$ is n^{2^k}, which also means that the computational complexity of $f_k(x)$ is at least n^{2^k}. □

5 FPT Algorithms for Time Complexity of Proofs and Equations

A crucial property of the time complexity is that there is an efficient algorithm computing the time complexity of a proof of equation given the encoding of a proof or equation *as its input*.

In more detail, fix a standard encoding of PV proofs π, PV equations, there are algorithms $A_\mu(\pi, \vec{n})$, $A_\beta(\pi, e, \vec{n})$, and a function $f(p) = \exp(\exp(O(p)))$ such that

- $A_\mu(\pi, \vec{n})$: If π encodes a PV proof and $\vec{n} = n_1, \ldots, n_k$ is the binary encoding of input lengths to variables in the conclusion of π, $A_\mu(\pi, \vec{n})$ outputs the binary representation of $\mu_\pi^{\mathsf{TC}}(\vec{n})$ in time $f(|\pi|) \cdot \mathsf{polylog}(n_1 + \cdots + n_k)$.
- $A_\beta(\pi, e, \vec{n})$: If π encodes a PV proof, e encodes an equation, and $\vec{n} = (n_1, \ldots, n_k)$ is the binary encoding of input lengths to variables in e, $A_\beta(\pi, e, \vec{n})$ outputs the binary representation of $\beta_{e,\pi}^{\mathsf{TC}}(\vec{n})$ in time $f(|\pi| + |e|) \cdot \mathsf{polylog}(n_1 + \cdots + n_k)$.

That is, there is an *FPT algorithm* for computing the time complexity of proofs and equations, where the length of the PV proof π is considered as a parameter.

Despite that $f(p)$ is a rapidly growing function, the FPT algorithms computing the time complexity of proofs and equations are more efficient than the functions definable in the proofs. Recall that in Proposition 4.6, we define functions $f_1(x) = x\#x$, $f_2(x) = f_1(x)\#f_1(x)$, ..., $f_k(x) = f_{k-1}(x)\#f_{k-1}(x)$ in a proof of size $O(k)$, where $x\#y$ is defined as

$$x\#\varepsilon := \varepsilon, \quad x\#s_i(y) := x \circ (x\#y).$$

Then the function $f_k(x)$ outputs a string of length $|x|^{2^k} \gg \exp(\exp(O(k))) \cdot \mathsf{polylog}(|x|)$ for large k and large input length $|x|$.

5.1 Preparations

Before describing the algorithms, we define notions that will be useful in the analysis of the algorithms: The postfix representation of PV proofs and equations, and the height of an occurrence of a term t in a proof or equation.

Postfix representation. We will define the postfix representation of terms, equations, proof lines, and proofs as a sequence of strings as follows.

The postfix representation of a term t, denoted by $[t]_{\mathsf{pf}}$ is defined as follows.

- If $t = \varepsilon$ or t is a variable, $[t]_{\mathsf{pf}} := \langle t \rangle$
- If t is a composition of a function f and terms $t_1, \ldots, t_k$, $[t]_{\mathsf{pf}} := \langle [t_1]_{\mathsf{pf}}, \ldots, [t_k]_{\mathsf{pf}}, t \rangle$.

The postfix representation of an equation $e : t_1 = t_2$ is defined as $[e]_{\mathsf{pf}} := \langle [t_1]_{\mathsf{pf}}, [t_2]_{\mathsf{pf}} \rangle$.

Let e be a proof line. Suppose that e is an equation, the postfix representation of the proof line e is defined as its postfix representation as an equation. Otherwise e is a proof line that introduces a new function symbol f, and we define $[e]_{\mathsf{pf}}$ as follows:

- If f is introduced by the composition rule from a term t', we define $[f]_{\mathsf{pf}} := \langle [t']_{\mathsf{pf}}, f(\vec{x}) \rangle$.
- If f is introduced by the limited recursion rule, we define $[f]_{\mathsf{pf}} := \langle f(\vec{x}) \rangle$.

Let $\pi : e_1, \ldots, e_m$ be a PV proof. The postfix representation of π, denoted by $[\pi]_{\mathsf{pf}}$, is defined as the concatenation of $[e_1]_{\mathsf{pf}}, \ldots, [e_m]_{\mathsf{pf}}$ in this order. Moreover, let e be an equation, we define $[\pi, e]_{\mathsf{pf}} :=$ as $\langle [\pi]_{\mathsf{pf}}, [e]_{\mathsf{pf}} \rangle$, i.e., the concatenation of $[\pi]_{\mathsf{pf}}$ and $[e]_{\mathsf{pf}}$.

Height of an occurrence. Let $\pi : e_1, \ldots, e_m$ be a PV proof, e be an term, and t be a term that occurs in (π, e). For each occurrence of t in (π, e), there is a corresponding entry containing t in the list $[\pi, e]_{\mathsf{pf}}$. The height of an occurrence of t in (π, e) is defined as the index of the corresponding entry in the list $[\pi, e]_{\mathsf{pf}}$.

Example 5.1. Consider the proof following $\pi : e_1, e_2, e_3, e_4, e_5$ and equation $e : f_1(\varepsilon) = \varepsilon$.

$$\begin{aligned}
e_1 &: f_1(x) := \varepsilon \circ x & [e_1]_{\mathsf{pf}} &:= \langle f_1 \rangle \\
e_2 &: f_1(x) = \varepsilon \circ x & [e_2]_{\mathsf{pf}} &:= \langle x, f_1(x), \varepsilon, x, \varepsilon \circ x \rangle \\
e_3 &: f_1(\varepsilon) = \varepsilon \circ \varepsilon & [e_3]_{\mathsf{pf}} &:= \langle \varepsilon, f_1(\varepsilon), \varepsilon, \varepsilon, \varepsilon \circ \varepsilon \rangle \\
e_4 &: x \circ \varepsilon = \varepsilon & [e_4]_{\mathsf{pf}} &:= \langle x, \varepsilon, x \circ \varepsilon, \varepsilon \rangle \\
e_5 &: \varepsilon \circ \varepsilon = \varepsilon & [e_5]_{\mathsf{pf}} &:= \langle \varepsilon, \varepsilon, \varepsilon \circ \varepsilon, \varepsilon \rangle
\end{aligned}$$

The postfix representation of e is $[e]_{\mathsf{pf}} := \langle \varepsilon, f_1(\varepsilon), \varepsilon \rangle$. The term $f_1(\varepsilon)$ has two occurrences in (π, e): The first occurrence is in e_3 and has height 8, and the second occurrence is in e and has height 21.

Notation. Let $\vec{x} = (x_1, \ldots, x_k)$ be a sequence of variables, we define $|\vec{x}| := |x_1| + |x_2| + \cdots + |x_k|$.

5.2 Algorithm for Bounding Values

Lemma 5.1. *There is a constant $c \geq 1$ such that the following holds. Let π be a valid* PV *proof and e be an equation such that every function symbol in e has been defined in π. Let t be a term that occurs in (π, e). The bounding value $\ell_t(\vec{n})$ satisfies the inequality $\log \ell_t(\vec{n}) \leq \exp(\exp(c \cdot (|\pi| + |e|))) \cdot \log(|\vec{n}|)$ (when $\ell_t(\vec{n}) > 0$).*

Proof. Fix an occurrence of t in (π, e) and denote the height by h_t, we will prove by induction on h_t that $\log \ell_t(\vec{n}) \leq \exp(\exp(c_h \cdot h_t)) \cdot \log(|\vec{n}|)$ for some large constant $c_h \geq 1$, which suffices as $h_t = O(|\pi| + |e|)$. The base case is trivial as if $h_t = 1$, t must either be ε or a variable x_i. In the former case $\ell_t(\vec{n}) = 0$, and in the latter case $\log \ell_t(\vec{n}) = \log(n_i) \leq \log(|\vec{n}|)$.

For the induction case, we consider the structure of the term t. Suppose that t is ε or a variable, we have $\ell_t(\vec{n}) \leq \log(|\vec{n}|)$ as in the base case. Otherwise, let f be an l-ary function, $t_1, \ldots, t_l$ be terms, and $t \equiv f(t_1, \ldots, t_l)$. Note that the earliest occurrences of $t_1, \ldots, t_l$ must be of height at most $h_t - 1$, so by induction hypothesis, we have

$$\log \ell_{t_1}(\vec{n}), \ldots, \log \ell_{t_l}(\vec{n}) \leq \exp(\exp(c_h \cdot (h_t - 1))) \cdot \log(|\vec{n}|). \tag{5.1}$$

Consider how the function symbol f is introduced. Suppose that f is an initial function, then $\ell_f(\vec{n}) \leq |\vec{n}|^2$, and in such case we have

$$\log \ell_t(\vec{n}) \leq \log \left(\sum_{i=1}^{l} \ell_{t_i}(n_i) \right)^2 \leq 2h_t \cdot \exp(\exp(c_h \cdot (h_t - 1)) \cdot \log(|\vec{n}|),$$

which is bounded by $\exp(\exp(c_h \cdot h_t)) \cdot \log(|\vec{n}|)$ for a sufficiently large constant c_h. (Note that the second inequality follows from $l \leq h_t$.) Otherwise, the function symbol f is introduced by either the composition rule or the limited recursion rule.

- (*Composition*). Suppose that f is introduced by the composition rule, there is a term t', a proof line $f := t'$, such that the earliest occurrence of t' is of height at most $h_t - 1$. By the induction hypothesis, we know that
$$\log \ell_{t'}(\vec{n}) \leq \exp(\exp(c_h \cdot (h_t - 1))) \cdot \log(|\vec{n}|).$$
Also, as $\ell_t(\vec{n}) := \ell_{t'}(\ell_{t_1}(\vec{n}), \ldots, \ell_{t_l}(\vec{n}))$ by the definition of bounding values, we have
$$\begin{aligned}
\log \ell_t(\vec{n}) &= \log \ell_{t'}(\ell_{t_1}(\vec{n}), \ldots, \ell_{t_l}(\vec{n})) \\
&\leq \exp(\exp(c_h \cdot (h_t - 1))) \cdot \log \left(\sum_{i=1}^{l} \ell_{t_i}(\vec{n}) \right) \\
&\leq h_t \cdot \exp(\exp(c_h \cdot (h_t - 1))) \cdot \exp(\exp(c_h \cdot (h_t - 1))) \cdot \log(|\vec{n}|) \\
&\leq \exp(\exp(c_h \cdot h_t)) \cdot \log(|\vec{n}|),
\end{aligned}$$
where the second line follows from $l \leq h_t$ and Equation (5.1), and the third line follows when c_h is a sufficiently large constant.

- (*Limited Recursion*). Suppose that f is introduced by the limited recursion rule from g, h_0, h_1, k_0, k_1, where the earliest occurrences of g, h_0, h_1, k_0, k_1 must be of height at most $h_t - 1$. By the induction hypothesis, we have
$$\ell_g(\vec{n}), \ell_{k_0}(\vec{n}), \ell_{k_1}(\vec{n}) \leq \exp(\exp(c_h \cdot (h_t - 1))) \cdot \log(|\vec{n}|).$$
Let n_y be the input length of the recursion variable, we know by the definition of bounding values that $\ell_t(\vec{n}) := n_y \cdot (\ell_{k_0}(\vec{n}) + \ell_{k_1}(\vec{n})) + \ell_g(\vec{n})$. Subsequently,
$$\begin{aligned}
\log \ell_t(\vec{n}) &\leq \log(|\vec{n}| \cdot (\ell_{k_0}(\vec{n}) + \ell_{k_1}(\vec{n})) + \ell_g(\vec{n})) \\
&\leq \log(|\vec{n}|) + 3 \cdot \exp(\exp(c_h \cdot (h_t - 1))) \cdot \log(|\vec{n}|) \\
&\leq \exp(\exp(c_h \cdot h_t)) \cdot \log(|\vec{n}|).
\end{aligned}$$

We can then set c_h to be a sufficiently large constant such that the proof above follows. □

Next, we prove that there is an efficient FPT algorithm computing the bounding value of terms. Indeed, the algorithm simply recursively computes the bounding value according to the definition.

Lemma 5.2 (algorithm for bounding values)**.** *There is an algorithm $L(\pi, e, t, \vec{n})$ that works as follows.*

- *(Input). The encoding of a valid* PV *proof π, an equation e such that all equations have been defined in π, a term t in (π, e), and $\vec{n} = (n_1, \ldots, n_k)$, where $n_1, \ldots, n_k$ denotes the input lengths of all variables $x_1, \ldots, x_k$ in the term t.*
- *(Output). The binary encoding of the bounding value $\ell_t(\vec{n})$.*
- *(Efficiency). The algorithm runs in time* $\exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(|\vec{n}|)$.

Proof Sketch. We compute $\ell_t(\vec{n})$ naïvely according to its definition (see Appendix A.3) using a recursion algorithm. The analysis of the algorithm is similar to the proof of Lemma 5.1, and thus we will only sketch the proof.

Let t be a term. We define $T(h, n)$ be the time complexity of computing $\ell_t(\vec{n})$, where h is the height of the earliest occurrence of t in (π, e), and $n := |\vec{n}|$. In the base case, we can see that $T(1, n) \le \mathsf{polylog}(n)$ as in such case t must either be ε or a variable. Otherwise, it can be verified that for some constant $c \ge 1$,

$$T(h, n) \le h \cdot T(h - 1, \ell(h, n)) + \exp(\exp(c \cdot h)) \cdot \log^c(n), \tag{5.2}$$

where $\ell(h, n) := \exp(\exp(\exp(c \cdot h)) \cdot \log n)$. In more detail, this is because we will make at most h recursive calls, each of which is on a term with smaller height, and the input length is at most the bounding value $\ell_{t'}(\vec{n})$ of a term t' in π, e; after the recursive calls, the additional time complexity is a polynomial in the bit-length of $\ell_t(\vec{n})$, which is upper bounded in $\exp(\exp(O(h_t))) \cdot \log n$ as proved in Lemma 5.1.

We then prove by induction that for some sufficiently large constant c',

$$T(h, n) \le \exp(\exp(c' \cdot h)) \cdot (\log n)^{c'}.$$

The base case is trivial. For the induction case, we can see that

$$\begin{aligned} T(h, n) &\le h \cdot T(h - 1, \ell(h, n)) + \exp(\exp(c \cdot h)) \cdot \log^c(n) \\ &\le h \cdot \exp(\exp(c' \cdot (h - 1))) \cdot (\exp(\exp(c \cdot h)) \cdot \log n)^{c'} + \exp(\exp(c \cdot h)) \cdot \log^c(n) \\ &\le \exp(\exp(c' \cdot h)) \cdot (\log n)^{c'}, \end{aligned}$$

where the last inequality follows for a sufficiently large constant c'. The theorem then follows as for any term t, the height of occurrences of t is at most $O(|\pi| + |e|)$. □

5.3 Algorithms for ATC

Next, we show that there are efficient algorithms for computing the ATC of equations and proof lines, i.e., $\beta^+_{e,\pi}(\vec{n})$ and $\mu^+_{e,\pi}(\vec{n})$.

Lemma 5.3. *There is a constant $c \ge 1$ such that the following holds. Let π be a valid* PV *proof and e be an equation such that every function symbol in e has been defined in π. Let t be a term that occurs in (π, e). The ATC of the term $\beta^+_{t,\pi}(\vec{n})$ satisfies the inequality $\log \beta^+_{t,\pi}(\vec{n}) \le \exp(\exp(c \cdot (|\pi| + |e|))) \cdot \log(|\vec{n}|)$.*

Proof. The proof is almost identical to the proof of Lemma 5.1 and is omitted. □

Lemma 5.4 (algorithm for ATC of equations)**.** *There is an algorithm $A_{\beta^+}(e, \pi, \vec{n})$ as follows:*

- *(Input). The encoding of a* PV *equation e, proof π, and $\vec{n} = (n_1, \ldots, n_k)$, where $n_1, \ldots, n_k$ denotes the input lengths of all variables $x_1, \ldots, x_k$ in the equation e.*
- *(Output). The binary encoding of $\beta^+_{e,\pi}(\vec{n})$.*
- *(Efficiency). The algorithm runs in time* $\exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(\vec{n})$.

Proof Sketch. The algorithm follows the definition of $\beta^+_{e,\pi}(\vec{n})$ straightforwardly. Let $e : t_1 = t_2$, we calculate $\beta^+_{t,\pi}(\vec{n})$ for $t \in \{t_1, t_2\}$ using a brute-force recursive algorithm $A_{\beta^+}(t, \vec{n})$. In more detail, the algorithm $A_{\beta^+}(t, \vec{n})$ considers the structure of the term t:

- If t is an initial function, it outputs $c \cdot n$ if t is s_0, s_1 or TR, and outputs $c \cdot n_1 \cdot n_2$ if it is $\circ$ or ITR.
- If t is a function introduced by the composition or the limited recursion rule, it recursively searches the ATC of at most two terms on corresponding input lengths. For instance, if t is a function $f_{t'}$ introduced by composition from the term t', the algorithm recursively searches $A_{\beta^+}(t', 1^{\vec{n}})$ and outputs the answer.
- Otherwise, t is the composition of a function f and terms $t_1, \ldots, t_i$. The algorithm recursively searches the ATC of $i + 1$ terms — f and $s_1, \ldots, s_i$ — on corresponding input lengths.

The correctness of the algorithm is trivial. The analysis of the algorithm is similar to Lemma 5.2, so we will only sketch the proof.

Let $T(h, n)$ be the time complexity of $A_{\beta^+}(t, \vec{n})$ when the earliest occurrence of t in (π, e) is of height h and $n := n_1 + \cdots + n_k$, where $\vec{n} := (n_1, \ldots, n_k)$. The function $T(h, n)$ satisfies $T(1, n) \leq \mathsf{polylog}(n)$ and for $h > 1$,

$$T(h, n) \leq h \cdot T(h - 1, \ell(h, n)) + \exp(\exp(c \cdot h)) \cdot \log^c(n),$$

for some constant c, where $\ell(h, n) := \exp(\exp(\exp(c \cdot h)) \cdot \log n)$. This is the same as Equation (5.2) and thus we can conclude that $T(h, n) \leq \exp(\exp(O(h))) \cdot \mathsf{polylog}(n)$ following the proof of Lemma 5.2. □

Lemma 5.5 (algorithm for ATC of new proof lines)**.** *There is an algorithm $A_{\mu^+}(e, \pi, \vec{n})$ as follows:*

- *(Input). The encoding of a proof π, a new proof line e, and $\vec{n} = (n_1, \ldots, n_k)$, where $n_1, \ldots, n_k$ denotes the input lengths of all variables $x_1, \ldots, x_k$ in e.*
- *(Output). The binary encoding of $\mu^+_{e,\pi}(\vec{n})$.*
- *(Efficiency). The algorithm runs in time* $\exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(|\vec{n}|)$.

Proof Sketch. The algorithm computes $\mu^+_{e,\pi}(\vec{n})$ by definition — it considers how the new proof line e is introduced after π and makes at most $|\pi| + |e|$ queries to the algorithm $A_{\beta^+}(\pi, e, \vec{n}')$, where $\log(|\vec{n}'|) \leq \exp(\exp(O(|\pi| + |e|))) \cdot \log(|\vec{n}|)$. The time complexity of the algorithm follows from Lemma 5.4. □

5.4 Algorithm for AIL

We now design and analyze algorithms for computing the AIL of equations β-$\mathsf{AIL}_{e,e_i}(\vec{n})$, AIL of new proof lines μ-$\mathsf{AIL}_{e,e_i}(\vec{n})$, as well as the acquisition closure of an acquisition map.

Lemma 5.6 (algorithm for AIL of equations)**.** *There is an algorithm $A_{\beta\text{-}\mathsf{AIL}}(e, \pi, i, \vec{n})$ as follows:*

- *(Input). The encoding of a proof $\pi : e_1, \ldots, e_m$, an equation e, an index $i \in [m]$, and $\vec{n} = (n_1, \ldots, n_k)$, where $n_1, \ldots, n_k$ denotes the input lengths of variables $x_1, \ldots, x_k$ in e.*
- *(Output). The binary encoding of the set β-$\mathsf{AIL}_{e,e_i}(\vec{n})$.*
- *(Efficiency). The algorithm runs in time* $\exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(|\vec{n}|)$.

Proof Sketch. The algorithm simply computes $\beta\text{-AIL}_{e,e_I}(\vec{n})$ according to the definition. In more detail, let $e : t_1 = t_2$, the algorithm will compute $\beta\text{-AIL}_{t_1,e_i}(\vec{n})$ and $\beta\text{-AIL}_{t_2,e_i}(\vec{n})$ and then take the union of these two sets. Let $t \in \{t_1, t_2\}$, we will compute $\beta\text{-AIL}_{t,e_i}(\vec{n})$ using a brute-force recursive algorithm $A_{\beta\text{-AIL}}(t, e, \pi, i, \vec{n})$. In more detail, the algorithm considers the structure of the term t:

- If t is ε, a variable, or an initial function, the algorithm outputs $\varnothing$.
- If t is a function symbol introduced by composition, i.e., $t = f_{t'}$ for some term t', the algorithm recursively computes $A_{\beta\text{-AIL}}(t', e, \pi, i, \vec{n})$.
- If t is a function symbol introduced by limited recursion from g, h_σ, k_σ for $\sigma \in \{0,1\}$, the algorithm makes at most three recursive calls and computes their union $\Delta(\vec{n}, n_y)$ according to Equation (4.7). If e_i is not $\mathsf{ITR}(h_\sigma(\vec{x}, y, z), z \circ k_\sigma(\vec{x}, y)) = \varepsilon$ for some $\varepsilon \in \{0,1\}$, the algorithm simply outputs $\Delta(\vec{n}, n_y)$. Otherwise, it outputs $\Delta(\vec{n}, n_y) \cup \{(\vec{n}, n_y, \ell_f(\vec{n}, n_y))\}$, where $\ell_f(\vec{n}, n_y))$ can be computed by Lemma 5.2.
- If t is a composition of a function f and terms $s_1, \ldots, s_j$, the algorithm makes $j+1$ recursive calls and outputs the union of their outputs; see Equation (4.8).

It remains to analyze the time complexity of the algorithm. As the analysis is similar to the algorithm in Lemma 5.2 and Lemma 5.4, we will only sketch the proof.

We can see that the algorithm maintains the invariant that for each recursive call to the algorithm $A_{\beta\text{-AIL}}(t, e, \pi, i, \vec{n})$, t must be a term that occurs in (π, e), and the height of the earliest occurrence of t in (π, e) must strictly decrease in any recursive call. Moreover, if t is of height t, the algorithm makes at most h recursive calls.

Let $T(h, n)$ be the time complexity of $A_{\beta\text{-AIL}}(t, e, \pi, i, \vec{n})$ when the earliest occurrence of t is of height h and $n = |\vec{n}|$. The function $T(h, n)$ must satisfies $T(1, n) \le \mathsf{polylog}(n)$ and for $h > 1$,

$$T(h, n) \le h \cdot T(h-1, \ell(h, n)) + \exp(\exp(c \cdot h)) \cdot \log^c(c)$$

for some constant c, where $\ell(h, n) := \exp(\exp(\exp(c \cdot h)) \cdot \log n)$. This is the same as Equation (5.2), and it follows that $T(h, n) = \exp(\exp(O(h))) \cdot \mathsf{polylog}(n)$. The lemma follows as $h \le |\pi| + |e|$. □

Lemma 5.7 (algorithm for AIL of new proof lines)**.** *There is an algorithm $A_{\mu\text{-AIL}}(e, \pi, i, \vec{n})$ as follows:*

- *(Input). The encoding of a proof $\pi : e_1, \ldots, e_m$, a new proof line e following π, an index $i \in [m]$, and $\vec{n} = (n_1, \ldots, n_k)$, where $n_1, \ldots, n_k$ denotes the input lengths of variables $x_1, \ldots, x_k$ in e.*
- *(Output). The binary encoding of $\mu\text{-AIL}_{e,e_i}(\vec{n})$.*
- *(Efficiency). The algorithm runs in time $\exp(\exp(|\pi| + |e|)) \cdot \mathsf{polylog}(|\vec{n}|)$.*

Proof Sketch. The algorithm computes $\mu\text{-AIL}_{e,e_i}(\vec{n})$ according to the definition. It considers how the new proof line e is introduced after π, and in each case, it makes at most one call to the algorithm in Lemma 5.6 to compute $\beta\text{-AIL}_{e,e_i}(\vec{n})$. The analysis is straightforward. □

5.5 Algorithm for Acquisition Closure

Before describing the algorithm for acquisition closure, we first prove an upper bound on the size of the set of acquired input lengths.

Proposition 5.8. *Let $\pi : e_1, \ldots, e_m$ be a PV proof. For an equation e, $i \in [m]$, and any $\vec{n} \in \vec{\mathbb{N}}$, $|\beta\text{-AIL}_{e,e_i}(\vec{n})| \le \exp(O(|\pi|))$. Similarly, for a new proof line e, $i \in [m]$, and any $\vec{n} \in \vec{N}$, $|\mu\text{-AIL}_{e,e_i}(\vec{n})| \le \exp(O(|\pi| + |e|))$.*

Proof. It suffices to prove the first upper bound $|\beta\text{-AIL}_{e,e_i}(\vec{n})| \le \exp(O(|\pi|))$, as the second upper bound follows from the fact that $|\mu\text{-AIL}_{e,e_i}(\vec{n})| \le |\beta\text{-AIL}_{e,e_i}(\vec{n})| + 1$.

Indeed, we will prove by induction on h that for any t that has a height-h occurrence in (π, e), $|\beta\text{-AIL}_{t,e_i}(\vec{n})| \le \exp(c \cdot h)$ for some constant $c \ge 1$. This suffices as for the equation $e : t_1 = t_2$, we have

$$|\beta\text{-AIL}_{e,e_i}(\vec{n})| \le |\beta\text{-AIL}_{t_1,e_i}(\vec{n})| + |\beta\text{-AIL}_{t_2,e_i}(\vec{n})|,$$

and the height of t is at most $O(|\pi| + |e|)$.

The base case is trivial as for $h = 1$, the term t can only be ε, a variable, or an initial function, which implies that $\beta\text{-AIL}_{t,e_i}(\vec{n}) = \varnothing$. In the induction case, we notice that $\beta\text{-AIL}_{t,e_i}(\vec{n})$ is the union of at most 3 sets of form $\beta\text{-AIL}_{t',e_i}(\vec{n}')$ and a set of size 1. In addition, the (at most) 3 sets of form $\beta\text{-AIL}_{t',e_i}(\vec{n}')$ satisfy that the earliest occurrence of t' must have height at most $h-1$, and thus by induction hypothesis, $|\beta\text{-AIL}_{t',e_i}(\vec{n}')| \le \exp(c \cdot (h-1))$. We can then prove that

$$|\beta\text{-AIL}_{t,e_i}(\vec{n})| \le 3 \cdot \exp(c \cdot (h-1)) + 1 \le \exp(c \cdot h)$$

by fixing c to be sufficiently large. □

Now we are ready to describe the algorithm for acquisition closure.

Lemma 5.9 (algorithm for acquisition closure)**.** *There is an algorithm $A_{\mu\text{-AM}}(\pi, S)$ as follows:*

- *(Input). The encoding of a proof $\pi : e_1, \dots, e_m$ and an acquisition map $S \subseteq \{e_1, \dots, e_m\} \times \vec{\mathbb{N}}$.*
- *(Output). The encoding of the acquisition closure $\mu\text{-AM}_\pi(S)$.*
- *(Efficiency). The algorithm runs in $\exp(\exp(O(|\pi|)) \cdot |S| \cdot \mathsf{polylog}(n)$.*

Proof Sketch. It can be verified that $\mu\text{-AM}_\pi(S) = \bigcup_{\tau \in S} \mu\text{-AM}(\{\tau\})$, so it suffices to consider the case that S contains only one pair $(e_j, \vec{n}_j)$. Consider the following recursive algorithm $A_{\mu\text{-AM}}(e_j, \vec{n}_j)$ that computes $\mu\text{-AM}\{(e_j, \vec{n}_j)\}$: It enumerates over $j' < j$ and $\vec{n}_{j'} \in \mu\text{-AIL}_{e_j,e_{j'}}(\vec{n}_j)$, recursively calls $A_{\mu\text{-AM}}(e_{j'}, \vec{n}_{j'})$, and outputs the union of all recursive calls. The correctness of the algorithm is trivial, so it suffices to analyze its time complexity.

Let $T(h, n)$ be the time complexity of $A_{\mu\text{-AM}}(e_j, \vec{n})$ where e_j occurs at height h and $|\vec{n}| \le n$. It satisfies that $T(1, n) \le \mathsf{polylog}(n)$, and for $j > 1$,

$$T(j, n) \le \exp(c \cdot h) \cdot T(j-1, \ell(h, n)). \tag{5.3}$$

for $\ell(h, n) = \exp(\exp(\exp(c \cdot h)) \cdot \log(n))$, where $c \ge 1$ is a constant. To see this, notice that the algorithm will make at most

$$\left| \bigcup_{j' < j} \mu\text{-AM}_{e'_j,e_j}(\vec{n}) \right| \le h \cdot \exp(O(h)) \le \exp(O(h)), \tag{5.4}$$

recursive calls, and each of the query $A_{\mu\text{-AM}}(e_{j'}, \vec{n}')$ made by $A_{\mu\text{-AM}}(e_j, \vec{n})$ satisfies that $j' \le j-1$ and $|\vec{n}'| \le \ell(h, |\vec{n}|)$. Note that the first inequality in Equation (5.4) follows from Proposition 5.8.

We prove by induction on j that for some sufficiently large constant $c' \ge 1$,

$$T(j, n) \le \exp(\exp(c' \cdot j) \cdot \log^{c'} n.$$

The base case is straightforward. For the induction case where $j > 1$, we have

$$\begin{aligned} T(j, n) &\le \exp(c \cdot h) \cdot T(h-1, \ell(n)) \\ &\le \exp(c \cdot h) \cdot \exp(\exp(c' \cdot (h-1))) \cdot \log^{c'}(\exp(\exp(\exp(c \cdot h)) \log n)) \\ &\le \exp(\exp(c' \cdot h)) \cdot \log^{c'}(n), \end{aligned}$$

where the inequalities hold when c' is sufficiently large. This completes the proof as $h \le |\pi|$. □

5.6 Algorithm A_μ and A_β

Building on the algorithms above, we can finally present the algorithms for computing the time complexity of proofs and equations.

Theorem 5.10 (algorithm for time complexity of proofs)**.** *There is an algorithm $A_\mu(\pi, \vec{n})$ as follows:*

- *(Input). The encoding of a* PV *proof $\pi : e_1, \ldots, e_m$ and the binary encoding of $\vec{n} \in \vec{\mathbb{N}}$, where $\vec{n} = (n_1, \ldots, n_k)$ denotes the input lengths of variables $x_1, \ldots, x_k$ in the conclusion e_m of π.*
- *(Output). The binary encoding of $\mu_\pi^{\mathsf{TC}}(\vec{n})$.*
- *(Efficiency). The algorithm runs in time $\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(|\vec{n}|)$.*

Proof. Let $n = |\vec{n}|$. The algorithm first computes $\mu\text{-}\mathsf{AM}_\pi(\{e_m, \vec{n}\})$ using the algorithm in Lemma 5.9 in $\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(n)$ time. For each pair $(e_i, \vec{l}) \in \mu\text{-}\mathsf{AM}_\pi(\{e_m, \vec{n}\})$, we can compute $\mu^+_{e_i,\pi_{<i}}(\vec{l})$ using the algorithm in Lemma 5.5 in time

$$\begin{aligned}&\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(\exp(\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(n)))\\ \le\ &\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(n),\end{aligned}$$

where the term $\exp(\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(n))$ is an upper bound of $|\vec{l}|$. We then outputs the summation of $\mu^+_{e_i,\pi_{<i}}(\vec{l})$ for all such pairs. □

Theorem 5.11 (algorithm for time complexity of equations)**.** *There is an algorithm $A_\beta(\pi, e, \vec{n})$ as follows:*

- *(Input). The encoding of a* PV *proof π (called the context), an equation e such that all function symbols in e are introduced in π, and the binary encoding of $\vec{n} \in \vec{\mathbb{N}}$, where $\vec{n} = (n_1, \ldots, n_k)$ denotes the input lengths of variables $x_1, \ldots, x_k$ in e.*
- *(Output). The binary encoding of $\beta_{e,\pi}^{\mathsf{TC}}(\vec{n})$.*
- *(Efficiency). The algorithm runs in time $\exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(|\vec{n}|)$.*

Proof. Let $n = |\vec{n}|$. Recall that

$$\beta_{e,\pi}^{\mathsf{TC}}(\vec{n}) = \beta^+_{e,\pi}(\vec{n}) + \sum_{(e_i,\vec{l}) \in \mu\text{-}\mathsf{AM}_\pi(\beta\text{-}\mathsf{AIL}^*_{e,\pi}(\vec{n}))} \mu^+_{e_i,\pi_{<i}}(\vec{l}),$$

where $\pi_{<i} := e_1, \ldots, e_{i-1}$, and $\beta\text{-}\mathsf{AIL}_{e,\pi}(\vec{n})$ is defined as

$$\beta\text{-}\mathsf{AIL}^*_{e,\pi}(\vec{n}) := \bigcup_{i \le m} \left\{ (e_i, \vec{l}) \mid \vec{l} \in \beta\text{-}\mathsf{AIL}_{e,e_i}(\vec{n}) \right\}.$$

The first term can be computed by calling the algorithm in Lemma 5.4 in time $\exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(n)$, and therefore it suffices to compute the second term.

The algorithm first computes $S = \beta\text{-}\mathsf{AIL}^*_{e,\pi}(\vec{n})$ in time $\exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(n)$ using the algorithm in Lemma 5.6. Then it calls the algorithm in Lemma 5.9 to compute the acquisition closure of S in time

$$T_1 := |S| \cdot \exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(n) = \exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(n).$$

For each pair $(e_i, \vec{l}) \in \mu\text{-}\mathsf{AM}_\pi(S)$, we can compute $\mu^+_{e_i,\pi_{<i}}(\vec{l})$ using the algorithm in Lemma 5.5 in time

$$\begin{aligned}T_2 &:= \exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(\exp(\exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(n)))\\ &= \exp(\exp(O(|\pi| + |e|)) \cdot \mathsf{polylog}(n).\end{aligned}$$

The algorithm then outputs the summation over all such pairs. The total time complexity of the algorithm is $\mathsf{poly}(T_1, T_2) = \exp(\exp(O(|\pi| + |e|))) \cdot \mathsf{polylog}(n)$. □

5.7 Formalization in PV

We now argue that the algorithms above — $L, A_{\beta^+}, A_{\mu^+}, A_{\beta\text{-AIL}}, A_{\mu\text{-AIL}}, A_{\mu\text{-AM}}$ — can be formalized in PV naturally. We only demonstrate the formalization of $L(\pi, e, t, \vec{n})$ for bounding values, and the formalization of other algorithms is similar.

Recall that the algorithm $L(\pi, e, t, \vec{n})$ runs in time $\exp(\exp(c \cdot (|\pi| + |e|)) \cdot \log^c(|\vec{n}|)$ for some constant c (see Lemma 5.2). As functions in PV must run in polynomial time, we formalize L as a PV function $f_L(\pi, e, t, \vec{n}, N)$ that takes an additional string N of length $\exp(\exp(c \cdot (|\pi| + |e|))) \cdot \log^c(|\vec{n}|)$. That is, the PV function $f_L(\pi, e, t, \vec{n}, N)$ simulates the algorithm $L(\pi, e, t, \vec{n})$ for $|N|$ steps:

- If L does not halt after $|N|$ steps, $f_L(\pi, e, t, \vec{n}, N)$ outputs ε.
- Otherwise, $f_L(\pi, e, t, \vec{n}, N)$ outputs $L(\pi, e, t, \vec{n})$ encoded in a binary string.

In more detail, let M_L be the Turing machine for the algorithm L, we can define $f_L(\pi, e, t, \vec{n}, N)$ using the limited recursion rule on the variable N from g, h_σ, k_σ, $\sigma \in \{0, 1\}$, where

- (*Base Case*). $g(\pi, e, t, \vec{n})$ outputs the initial configuration of $M_L(\pi, e, t, \vec{n}, N)$.
- (*Recursion*). For $\sigma \in \{0, 1\}$, $h_\sigma(\pi, e, t, \vec{n}, N, z)$ parses z as a configuration of $M_L(\pi, e, t, \vec{n})$, simulates M_L for one step, and outputs the configuration of M_L after this step.
- (*Length Bound*). As simulating a Turing machine for one step will increase the length of the configuration by at most $O(1)$, and the property can be proved in PV, we can define $k_\sigma(\pi, e, t, \vec{n}, N) = 0^{c_M}$ for a sufficiently large constant c_M. The equation $\mathsf{ITR}(h_\sigma(\pi, e, t, \vec{n}, N, z), z \circ k_i(\pi, e, t, \vec{n}, N)) = \varepsilon$ can be proved in PV assuming standard encoding of Turing machines and their configurations.

We note that this formalization indeed works for any *computable* function: For every Turing machine $M(x)$, the PV function $f(x, N)$ constructed above computes $M(x)$ provided as N is longer than the running time of $M(x)$.

In addition, we will prove the time complexity upper bound of L in PV by proving that $f_L(\pi, e, t, \vec{n}, N) \neq \varepsilon$ if $|N| \geq \exp(\exp(c \cdot (|\pi| + |e|)) \cdot \log^c(|\vec{n}|)$. More formally:

Lemma 5.12. *Let* $\mathsf{UB}_L(N, \pi, e, t, \vec{n})$ *be the* PV *function that outputs* 1 *if* $(\pi, e, t, \vec{n}, N)$ *is a valid input instance of* L *and* $|N| \geq \exp(\exp(c \cdot (|\pi| + |e|)) \cdot \log^c(|\vec{n}|)$*, and outputs* 0 *otherwise;* $\mathsf{ITE}(u, x, y)$ *be the* PV *function that outputs* x *if the last bit of* u *is* 1*, and outputs* y *otherwise;* $\mathsf{IsNotEps}(x)$ *be the* PV *function that outputs* 1 *if* $x \neq \varepsilon$*, and outputs* 0 *otherwise.*

Then PV *proves the equation*

$$\mathsf{ITE}(\mathsf{UB}_L(N, \pi, e, \vec{n}), \mathsf{IsNotEps}(f_L(\pi, e, t, \vec{n}, N), 1) = 1. \tag{5.5}$$

Proof Sketch. We formalize the proof of Lemma 5.1 and Lemma 5.2 in PV. For instance, we need to prove the function $T(h, n)$ satisfying that $T(1, n) = \mathsf{polylog}(n)$ and

$$T(h, n) \leq h \cdot T(h - 1, \ell(h, n)) + \exp(\exp(c \cdot h)) \cdot \log^c(n), \tag{5.6}$$

where $\ell(h, n) := \exp(\exp(\exp(c \cdot h)) \cdot \log n)$, and $n := |\vec{n}|$. This is proved by induction on h, which is directly not available in PV as T is not a polynomial-time computable function.

However, as in Equation (5.5), we are allowed to take as input a string N, we can perform a case study on whether $\mathsf{UB}_L(N, \pi, e, \vec{n}) = 1$ or $\mathsf{UB}_L(N, \pi, e, \vec{n}) = 0$. In the former case, we have $|N| \geq \exp(\exp(c \cdot (|\pi| + |e|))) \cdot |\vec{n}|$, and we can reformulate Equation (5.6) as an induction on the length of N that is available in PV. In the latter case, the LHS of Equation (5.5) will simply output 1. This completes the proof. □

Remark 5.1. Following the convention in bounded arithmetic literature (see, e.g., [Coo75, Bus86, Kra95, Jeř05, Oja04, Lê14, Pic14]), we will not attempt to formally write down the PV functions such as f_L. Such formalization is straightforward following the intuition we presented here, though the actual formalization could be extremely tedious.

Following any such formalization, the functions should satisfy Lemma 5.12 as well as *other properties* we will explain later. Most properties we will need are obvious provided that the formalization is straightforward, and we will explain the properties that need to be handled carefully.

6 Basic Properties of Time Complexity

Now we are ready to explore the basic properties of time complexity of proofs and equations. This section is organized as follows:

- In Section 6.1, we will prove that the *feasible deduction theorem*, which intuitively means that if $\varphi \to \psi$ and φ are provable with time complexity μ_1 and μ_2, respectively, then ψ is provable with time complexity (roughly) $O(\mu_1 + \mu_2)$. This is helpful when we need to prove time complexity upper bounds of complicated proofs.
- In Section 6.2, we will prove that time complexity satisfies the *feasible translation property*. That is, the correctness proof of the propositional translation $[e]^{\vec{n}}_{\mathsf{Cook}}$ of an equation e with time complexity β can be proved with time complexity $\beta^{O(1)}$.
- In Section 6.3, we will prove the *feasible proof generation theorem*, namely the correctness proof of the propositional translation $[\pi]^{\vec{n}}_{\mathsf{Cook}}$ of a proof π with time complexity μ can be proved with time complexity $\mu^{O(1)}$.

6.1 Feasible Deduction Theorem

To formalize the feasible deduction theorem, we first clarify the formalization of the implication connective $\varphi \to \psi$, which is not directly available in the language of PV. For instance, we can use the formalization of conditional equations introduced in [CU93]:

Definition 6.1 (conditional equation)**.** Let $s(\vec{x})$ be a PV term and $t_1(\vec{x}) = t_2(\vec{x})$ be a PV equation. The conditional equation $s(\vec{x}) \Rightarrow t_1(\vec{x}) = t_2(\vec{x})$ is the abbreviation of the equation

$$\mathsf{ITE}(s(\vec{x}), t_1(\vec{x}), t_2(\vec{x})) = t_2(\vec{x}),$$

where $\mathsf{ITE}(u, x, y)$ is the function defined by

$$\mathsf{ITE}(\varepsilon, x, y) := \varepsilon, \quad \mathsf{ITE}(s_0(z), x, y) := y, \quad \mathsf{ITE}(s_1(z), x, y) := x,$$

using the rule of limited recursion.

The conditional equation $s(\vec{x}) \Rightarrow t_1(\vec{x}) = t_2(\vec{x})$ formalizes the following statement: For any $\vec{x}$, if $s(\vec{x})$ is true (i.e. the last bit of $s(\vec{x})$ is 1), then $t_1(\vec{x}) = t_2(\vec{x})$. The case that $s(\vec{x}) = \varepsilon$ is considered undefined. It satisfies the desired properties of implication:

Proposition 6.1 ([CU93])**.** *Let $s(\vec{x}), t_1(\vec{x}), t_2(\vec{x})$ are* PV *terms. Then:*

- *(Modus Ponens). If* PV *proves $s(\vec{x}) = 1$ and $s(\vec{x}) \Rightarrow t_1(\vec{x}) = t_2(\vec{x})$, then* PV *proves $t_1(\vec{x}) = t_2(\vec{x})$.*
- *(Explosion).* PV *proves $s_0(z) \Rightarrow t_1(\vec{x}) = t_2(\vec{x})$.*
- *(Contraposition).* PV *proves $s(\vec{x}) \Rightarrow t_1(\vec{x}) = t_2(\vec{x})$ iff* PV *proves $\mathsf{EQ}(t_1(\vec{x}), t_2(\vec{x})) \Rightarrow \mathsf{LastBit}(s(\vec{x})) = 1$, where: $\mathsf{EQ}(x, y)$ outputs 1 if $x = y$ and 0 otherwise; $\mathsf{LastBit}(x)$ outputs the last bit of x if $x \neq \varepsilon$ or ε if $x = \varepsilon$.*

We note that although conditional equations do not allow the condition $s(\vec{x})$ to be an equation $e : s_1(\vec{x}) = s_2(\vec{x})$, we can always formalize the statement "e implies $t_1 = t_2$" by the conditional equation

$$\mathsf{EQ}(s_1(\vec{x}), s_2(\vec{x})) \Rightarrow t_1(\vec{x}) = t_2(\vec{x}),$$

where EQ is the PV function that checks whether two strings are the same. For simplicity, we denote this conditional equation by $s_1(\vec{x}) = s_2(\vec{x}) \Rightarrow t_1(\vec{x}) = t_2(\vec{x})$.

Moreover, we can formalize implication with two premises — $s_1(\vec{x}) = 1$ *and* $s_2(\vec{x}) = 1$ imply $t_1(\vec{x}) = t_2(\vec{x})$ — by the conditional equation

$$s_1(\vec{x}) \Rightarrow (s_2(\vec{x}) \Rightarrow t_1(\vec{x}) = t_2(\vec{x})).$$

For simplicity, we may drop the parenthesis by assuming that "$\Rightarrow$" is right-associative.

The feasible deduction theorem can then be formalized as follows:

Theorem 6.2 (feasible deduction theorem)**.** *There is a constant $d_{\mathsf{MP}} \in \mathbb{N}$ such that the following holds. Let $s(\vec{x}), t_1(\vec{x}), t_2(\vec{x})$ be* PV *terms. Suppose that $s(\vec{x}) = 1$ admits a* PV *proof π_s of time complexity $\mu_1(\vec{n})$, and $s(\vec{x}) \Rightarrow t_1(\vec{x}) = t_2(\vec{x})$ admits a* PV *proof $\pi_{s\Rightarrow t}$ of time complexity $\mu_2(\vec{n})$, then $t_1(\vec{x}) = t_2(\vec{x})$ admits a* PV *proof π_t of time complexity $\mu_3(\vec{n})$ such that*

$$\mu_3(\vec{n}) \leq d_{\mathsf{MP}} \cdot (\mu_1(\vec{n}) + \mu_2(\vec{n})).$$

Proof. The existence of the proof π_t follows from the first bullet of Proposition 6.1, and thus it suffices to analyze the feasibility of π_t. The proof π_t is the concatenation of π_s, $\pi_{s\Rightarrow t}$, and the following steps:

- (*Rewrite*): From $s(\vec{x}) = 1$ (the conclusion of π_s) and $\mathsf{ITE}(s(\vec{x}), t_1(\vec{x}), t_2(\vec{x})) = t_2(\vec{x})$ (the conclusion of $\pi_{s\Rightarrow t}$), we will conclude that $\mathsf{ITE}(1, t_1(\vec{x}), t_2(\vec{x})) = t_2(\vec{x})$. This part is denoted by π_{rw}. In more detail, this can be decomposed into the following steps:

  (1) Introduce a function symbol $f(\vec{x}, y) = \mathsf{ITE}(y, t_1(\vec{x}), t_2(\vec{x}))$ by the composition rule.
  (2) Derive the definition axiom $f(\vec{x}, y) = \mathsf{ITE}(y, t_1(\vec{x}), t_2(\vec{x}))$ of f.
  (3) From (L3) and $s(\vec{x}) = 1$, conclude that $f(\vec{x}, s(\vec{x})) = f(\vec{x}, 1)$.
  (4) From (2), conclude that $f(\vec{x}, s(\vec{x})) = \mathsf{ITE}(s(\vec{x}), t_1(\vec{x}), t_2(\vec{x}))$ with (L4).
  (5) From (2), conclude that $f(\vec{x}, 1) = \mathsf{ITE}(1, t_1(\vec{x}), t_2(\vec{x}))$ with (L4).
  (6) From (3), (4), (5), and $\mathsf{ITE}(s(\vec{x}), t_1(\vec{x}), t_2(\vec{x})) = t_2(\vec{x})$, conclude that $\mathsf{ITE}(1, t_1(\vec{x}), t_2(\vec{x})) = t_2(\vec{x})$ with (L1) and (L2).

- (*Simplify*): Prove that $\mathsf{ITE}(1, x, y) = x$. Then, by substituting $x/t_1(\vec{x})$ and $y/t_2(\vec{x})$, we can conclude that $\mathsf{ITE}(1, t_1(\vec{x}), t_2(\vec{x})) = t_1(\vec{x})$ with (L3). This part is denoted by π_{simp}.

- (*Equality*): From $\mathsf{ITE}(1, t_1(\vec{x}), t_2(\vec{x})) = t_1(\vec{x})$ and $\mathsf{ITE}(1, t_1(\vec{x}), t_2(\vec{x})) = t_2(\vec{x})$, we conclude that $t_1(\vec{x}) = t_2(\vec{x})$ using logical rules (L1) and (L2). This part is denoted by π_{EQ}.

We now analyze the the time complexity of the proof π_t by considering the contribution of π_{EQ}, π_{simp}, π_{rw}, $\pi_{s\Rightarrow t}$, and π_s in this order.

Let $e_1, \ldots, e_m$ be the proof lines in π_t. For each line e_i, the set of acquired input lengths (AILs) of the line e_i, denoted by $S(e_i)$, is defined as the set of $\vec{l} \in \vec{\mathbb{N}}$ such that $(e_i, \vec{l}) \in \mu\text{-}\mathsf{AM}_{\pi_t}(e_m, \vec{n})$. Recall that the time complexity of the proof π_t is defined as the summation of the ATC of e_i on the input length $\vec{l}$ for each $i \in [m]$ and each $\vec{l} \in S(e_i)$. Therefore, we need to consider $S(e_i)$ for each line e_i, and its contribution to time complexity (i.e. ATC of e_i on all input lengths $\vec{l} \in S(e_i)$).

The proof π_{EQ}. The proof π_{EQ} involves only (L1) and (L2), which propagates the AIL $\vec{n}$ to the premises. Thus the set of AILs of each line is simply $\{\vec{n}\}$. As all lines are introduced by (L1) and (L2), the ATC of each line e_i is proportional to its AIL as an equation (i.e. $\beta^+_{e_i,\pi_t}(\vec{n})$).

We next show that the ATC of each line as an equation is linearly bounded by the ATC $\beta(\vec{n})$ of $s(\vec{x}) \Rightarrow t_1(\vec{x}) = t_2(\vec{x})$, which is subsequently upper bounded by $O(\mu_2(\vec{n}))$. In more detail, let the ATC of t_1, t_2 on the input length $\vec{n}$ be $\beta_1(\vec{n}), \beta_2(\vec{n})$, respectively, consider the proof line

$$t_1(\vec{x}) = \mathsf{ITE}(1, t_1(\vec{x}), t_2(\vec{x})).$$

Its ATC as an equation on the input length $\vec{n}$ is the summation of four terms:

1. The ATC of $t_1(\vec{x})$ from the LHS on the input length $\vec{n}$, which is $\beta_1(\vec{n})$.
2. The ATC of $t_1(\vec{x})$ from the RHS on the input length $\vec{n}$, which is $\beta_1(\vec{n})$.
3. The ATC of $t_2(\vec{x})$ from the RHS on the input length $\vec{n}$, which is $\beta_2(\vec{n})$.
4. The ATC of $\mathsf{ITE}(1, u, v)$ on the input length $|u| = \ell_{t_1}(\vec{n})$ and $|v| = \ell_{t_1}(\vec{x})$. By the definition of ATC of initial functions, this is at most $c \cdot (|u| + |v|) \leq c \cdot (\beta_1(\vec{n}) + \beta_2(\vec{n}))$, where c is the formalization overhead. The inequality follows from $\ell_t(\vec{n}) \leq \beta^+_{t,\pi}(\vec{n})$.

Thus the total ATC is at most $(c+2) \cdot (\beta_1(\vec{n}) + \beta_2(\vec{n})) \leq (c+2) \cdot \beta(\vec{n})$. Therefore, the total contribution to time complexity is upper bounded by $O(\mu_2(\vec{n}))$.

The proof π_{simp}. We first consider the AILs. Note that there are only two lines in π_{simp}: $\mathsf{ITE}(1, x, y) = y$ and $\mathsf{ITE}(1, t_1(\vec{x}), t_2(\vec{x})) = t_2(\vec{x})$. The set of AILs of the second line is $\{\vec{n}\}$ — propagated from π_{EQ} as it is used as premises of (L1) and (L2). The set of AILs of the first line contains only one input length: $|x| = \ell_{t_1}(\vec{n})$ and $|y| = \ell_{t_2}(\vec{n})$, which is propagated from the second line of π_{simp} as it is used as a premise of (L3). Similar to the case of π_{EQ}, it can be verified that the ATC of each line is at most $O(\mu_2(\vec{n}))$.

The proof π_{rw}. Similarly, we argue that the total contribution to time complexity is at most $O(\mu_2(\vec{n}))$. Take line (4) in π_{rw} for example: $f(\vec{x}, s(\vec{x})) = \mathsf{ITE}(s(\vec{x}), t_1(\vec{x}), t_2(\vec{x}))$. The set of AILs of line (4) is simply $\{\vec{n}\}$ — propagated from line (6), which is subsequently propagated from π_{EQ} as line (6) of π_{rw} is used as premises of (L1) and (L2).

The ATC of the proof line (4) on the input length $\vec{n}$ is defined as the summation of two terms:

- The ATC of line (2) $f(\vec{x}, y) = \mathsf{ITE}(y, t_1(\vec{x}), t_2(\vec{y}))$ as an equation on the input length $|\vec{x}| = \vec{n}$ and $|y| = \ell_s(\vec{n})$. Subsequently, it is at most two times the ATC of the term $\mathsf{ITE}(y, t_1(\vec{x}), t_2(\vec{y}))$ on the input length $|\vec{x}| = \vec{n}$ and $|y| = \ell_s(\vec{n})$. This is at most $O(\mu_2(\vec{n}))$, since $\mathsf{ITE}(s(\vec{n}), t_1(\vec{x}), t_2(\vec{x}))$ occurs as a term in the conclusion of $\pi_{s \Rightarrow t}$.
- The ATC of the term $s(\vec{x})$ on the input length $\vec{n}$. This is at most $O(\mu_1(\vec{n}))$ as the term $s(\vec{x})$ also occurs in the conclusion of π_s.

The proof $\pi_{s \Rightarrow t}$ and π_s. First, notice that the set of AILs of the last line in $\pi_{s \Rightarrow t}$ is simply $\{\vec{n}\}$. This is propagated from line (6) of π_{rw}, which is subsequently propagated from π_{simp} and π_{EQ}. The AILs of the other lines in $\pi_{s \Rightarrow t}$ are propagated from the AILs of the last line — this is the same when we compute the time complexity of the proof $\pi_{s \Rightarrow t}$ on input length $\vec{n}$. Therefore, the total contribution to time complexity is exactly the time complexity of $\pi_{s \Rightarrow t}$ on input length $\vec{n}$, which is $\mu_2(\vec{n})$. Similarly, the contribution to time complexity by π_s is exactly the time complexity of π_s on the input length $\vec{n}$, which is $\mu_1(\vec{n})$.

Summary. In total, the time complexity of the proof π_t is the summation of $O(1)$ terms of at most $O(\mu_1(\vec{n}) + \mu_2(\vec{n}))$, where $O(\cdot)$ hides absolute constants that are independent of $s, t_1, t_2, \pi_s, \pi_{s \Rightarrow t}$. Therefore, the time complexity of π_t is at most $d_{\mathsf{MP}} \cdot (\mu_1(\vec{n}) + \mu_2(\vec{n}))$ if we set d_{MP} to be sufficiently large. □

Remark 6.2. The proof of Theorem 6.2 can be summarized as the analysis of two parts: The contribution to time complexity by the additional lines from π_s and $\pi_{s \Rightarrow t}$ to derive the conclusion, and the contribution to time complexity by the proofs $\pi_s, \pi_{s \Rightarrow t}$. The first term is small as the additional lines are simple and highly feasible, which is proved by a line-by-line inspection of the additional lines. The latter term is *exactly* the time complexity of π_s and $\pi_{s \Rightarrow t}$, as the additional lines only rely on the conclusion of π_s and $\pi_{s \Rightarrow t}$ on the input length $\vec{n}$ — that is, the additional lines neither use lines inside $\pi_s, \pi_{s \Rightarrow t}$ as premises, nor use the conclusions of $\pi_s, \pi_{s \Rightarrow t}$ on input lengths other than $\vec{n}$.

This proof explains the technical detail in the time complexity analysis of proofs — consider AILs and take summation over the ATC of each line. In the rest of the paper, we will describe the time complexity analysis of proofs only at a high level; nevertheless, it should be straightforward to fill in the technical details as we demonstrated in this proof.

6.2 Feasible Translation Theorem

Next, we prove the *feasible translation theorem*, which intuitively means that for every PV equation $e : t_1 = t_2$, the correctness of the propositional translation of e, i.e., $[e]^{\vec{n}}_{\mathsf{Cook}}$, can be provable in PV, and the proof is efficient in terms of its time complexity.

We first define some PV functions. $\mathsf{EQ}(x, y)$ is the PV function that outputs 1 if $x = y$, and outputs 0 otherwise; $\mathsf{Not}(x)$ outputs 1 if $x = 0$ and outputs 0 otherwise; $\mathsf{Satisfy}(\varphi, x)$ is the PV function that outputs 1 if x is a satisfying assignment of φ, and outputs 0 otherwise; $\mathsf{ValidA}(w)$ outputs 1 if w is a valid encoding of an assignment, and outputs 0 otherwise; $\mathsf{Input}_j(w)$ works as follows: if w is a valid assignment of the

propositional translation of an equation, and the input variables encode $\vec{x} = (x_1, \ldots, x_k)$, then $\mathsf{Input}_j(w)$ outputs x_j.

Let $\mathsf{Input}(w)$ be the shorthand of $(\mathsf{Input}_1(w), \mathsf{Input}_2(w), \ldots, \mathsf{Input}_k(w))$. Then:

Theorem 6.3 (feasible propositional translation theorem)**.** *There is a constant $d_{\mathsf{tr}} \in \mathbb{N}$ such that the following holds. For every* PV *proof π and every equation e such that all function symbols in $e : t_1(\vec{x}) = t_2(\vec{x})$ have been defined in π, there is a function $\mathsf{Assign}_e(\vec{x})$ and* PV *proofs π^1_{tr}, π^2_{tr} of the equations*

$$e^1_{\mathsf{tr}} : \mathsf{Not}(\mathsf{EQ}(t_1(\vec{x}), t_2(\vec{x}))) \Rightarrow \mathsf{Satisfy}([e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, \mathsf{Assign}_e(\vec{x})) = 0 \qquad \text{(soundness)}$$

$$e^2_{\mathsf{tr}} : \mathsf{ValidA}(w) \Rightarrow \mathsf{Satisfy}([e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, w) \Rightarrow t_1(\mathsf{Input}(w)) = t_2(\mathsf{Input}(w)) \qquad \text{(completeness)}$$

such that for $i \in \{1, 2\}$:

$$\mu^{\mathsf{TC}}_{\pi^1_{\mathsf{tr}}}(\vec{n}) = O((\beta^{\mathsf{TC}}_{e,\pi}(\vec{n}))^{d_{\mathsf{tr}}}); \quad \mu^{\mathsf{TC}}_{\pi^2_{\mathsf{tr}}}(\vec{n}, n_w) = O((\beta^{\mathsf{TC}}_{e,\pi}(\vec{n}) + n_w)^{d_{\mathsf{tr}}})$$

where $\vec{x} = (x_1, \ldots, x_k)$ are variables in e, $\vec{n}$ denotes the input length of $\vec{x}$, and n_w denotes the input length of the variables w in e^2_{tr}. Note that $O(\cdot)$ hides constants that may depend on π, e but are independent of $\vec{n}$.

Theorem 6.3 claims that the correctness proof of the propositional translation of equations can be proved with a fixed polynomial time complexity overhead. We will not attempt to calculate the constant d_{tr} in this paper as it may be sensitive to the exact formalization in PV. Nevertheless, if we do not care about the exact constant d_{tr}, our result is robust to formalizations.

6.2.1 Overview of the PV Proof

The proof $\pi^1_{\mathsf{tr}}, \pi^2_{\mathsf{tr}}$ involves three parts. In the first part, we prove the correctness of the translation of *terms*, i.e., $[t]^{\vec{n}}_{\mathsf{Cook}}$, including the appropriate formalization of soundness and completeness. We prove in the second part that e^1_{tr} holds, namely the propositional translation of equations is *complete*, using the completeness of propositional translation of terms. We then prove in the third part that e^2_{tr} holds.

We now describe each step in more detail. Note that below we argue in PV.

The first part: correctness of translation of terms. For each $t \in \{t_1, t_2\}$, we prove the correctness of the propositional translation $[t]^{\vec{n}}_{\mathsf{Cook}}$. That is:

- (*Soundness*). If all constraints are satisfied, the output variables must encode $t(\vec{x})$.
- (*Completeness*). There is a satisfying assignment such that the output variables encode $t(\vec{x})$.

These two properties can be formalized by PV equations. The formalization of the first bullet is straightforward: We introduce a PV function $\mathsf{Output}(\phi, w)$ such that given a formula ϕ that is the propositional translation of a term and an assignment w, $\mathsf{Output}(\phi, w)$ is the string encoded by the output variables under the assignment w; and for each $j \in [k]$, we introduce a PV function $\mathsf{Input}_j(\phi, w)$ that outputs j-th input string as encoded by the input variables of ϕ under the assignment w. The first bullet is then formalized as the conditional equation:

$$e^2_t : \mathsf{Satisfy}([t]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, w) \Rightarrow \mathsf{Output}([t]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, w) = t(\mathsf{Input}(w)), \tag{6.1}$$

where $\mathsf{Input}(w)$ is the shorthand of $(\mathsf{Input}_1(w), \ldots, \mathsf{Input}_k(w))$.

To formalize the second bullet, we introduce a PV function $\mathsf{AssignTerm}_t(\vec{x})$ that outputs the corresponding satisfying assignment given the input $\vec{x}$. The algorithm that finds the satisfying assignment is defined by induction on t according to the definition of $[t]^{\vec{n}}_{\mathsf{Cook}}$, and can be straightforwardly formalized in PV. The second bullet is then formalized as a PV equation:

$$e^1_t : \mathsf{AND}(\mathsf{Satisfy}([t]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, \mathsf{AssignTerm}_t(\vec{x})) = 1, \mathsf{EQ}(\mathsf{Output}([t]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, \mathsf{AssignTerm}_t(\vec{x})), t(\vec{x}))) = 1, \tag{6.2}$$

where $\mathsf{AND}(x,y)=1$ if and only if $x=1$ and $y=1$, and $\mathsf{AND}(x,y)=0$ otherwise.

We postpone the proofs of e^1_t and e^2_t to the analysis of their time complexity, as the complexity analysis follows closely from the proofs.

The second part: completeness. To prove e^1_{tr}, we first prove that $e^1_{t_1}$ and $e^1_{t_2}$ implies e^1_{tr}, which is formalized as the conditional equation

$$e^1_{t_1}(\vec{x}) \Rightarrow e^1_{t_2}(\vec{x}) \Rightarrow e^1_{\mathsf{tr}}(\vec{x}), \tag{6.3}$$

and apply Modus Ponens twice. At a high level, the proof of the conditional equation is as follows. Recall that $\mathsf{AssignTerm}_{t_1}(\vec{x})$ and $\mathsf{AssignTerm}_{t_2}(\vec{x})$ output the corresponding assignments for the propositional translation of $t_1(\vec{x})$ and $t_2(\vec{x})$, respectively. Given these two functions, we can define the PV function $\mathsf{Assign}_e(\vec{x})$ as follows:

- We assign variables in $[t_1]^{\vec{n}}_{\mathsf{Cook}}$ according to $t_1(\vec{x})$, and variables in $[t_2]^{\vec{n}}_{\mathsf{Cook}}$ according to $t_2(\vec{x})$.
- The assignment of other variables, i.e., the intermediate variables for comparing the output variables of $[t_1]^{\vec{n}}_{\mathsf{Cook}}$ and $[t_2]^{\vec{n}}_{\mathsf{Cook}}$, can be efficiently computed according after the output variables of $[t_1]^{\vec{n}}_{\mathsf{Cook}}$ and $[t_2]^{\vec{n}}_{\mathsf{Cook}}$ are fixed.

Assuming $e^1_{t_j}$, $j\in\{1,2\}$, we can prove that the output variables $[t_j]^{\vec{n}}_{\mathsf{Cook}}$ encode $t_j(\vec{x})$ under the assignment above. Since the premise of $e^1_{\mathsf{tr}}(\vec{x})$ further ensures that $t_1(\vec{x})=t_2(\vec{x})$, we can then conclude that the output variables of $[t_1]^{\vec{n}}_{\mathsf{Cook}}$ and $[t_2]^{\vec{n}}_{\mathsf{Cook}}$ encode the same string, which further implies that $[e]^{\vec{n}}_{\mathsf{Cook}}$ is satisfied by the assignment.

The third part: soundness. Similarly, the proof of e^2_{tr} follows from the conditional equation

$$e^2_{t_1}(\vec{x}) \Rightarrow e^2_{t_2}(\vec{x}) \Rightarrow e^2_{\mathsf{tr}}(\vec{x}) \tag{6.4}$$

by applying Modus Ponens twice. At a high level, the conditional equation is proved as follows. The premises of $e^2_{\mathsf{tr}}(\vec{x})$ imply

$$\mathsf{ValidA}(w)=1 \quad \text{and} \quad \mathsf{Satisfy}([e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}},w)=1.$$

Suppose that w encodes a satisfying assignment $\vec{x}$ of the propositional translation of e. Note that the premises $e^2_t(\vec{x})$ for $t\in\{t_1,t_2\}$ imply that the output variables of t_1,t_2 encodes $t_1(\vec{x})$ and $t_2(\vec{x})$, respectively, under the assignment w. In such case, we have $t_1(\vec{x})=t_2(\vec{x})$ as otherwise w cannot be a satisfying assignment. However, the premise of $e^2_{\mathsf{tr}}(\vec{x})$ implies that $t_1(\vec{x})\neq t_2(\vec{x})$, which leads to a contradiction.

6.2.2 Time Complexity Analysis of the Proof

Note that the proof of $e^1_{\mathsf{tr}}(\vec{x})$ and $e^2_{\mathsf{tr}}(\vec{x})$ are obtained by applying Modus Ponens on Equations (6.2) and (6.3), Equations (6.1) and (6.4). By feasible deduction theorem (see Theorem 6.2), it suffices to analyze the time complexity of each proof separately.

Time complexity analysis of the first part. The provability of both Equation (6.1) and (6.2) in PV are proved by induction on the height of the term t in (π,e) (see Section 4.4 for the definition of $[\cdot]_{\mathsf{Cook}}$). Note that this induction proof happens in meta-theory instead of PV, and we need to ensure d_{tr} is a constant that does not grow in the induction proof.

In the base case, i.e., t is of height 1 in (π,e), it must be ε, x, or an initial function. Suppose that t is one of ε, x, $s_0(x), s_1(x), \mathsf{TR}(x)$. Recall that we defined $\beta^{+}_{t,\pi}(n)=c\cdot n$, and thus $\beta^{\mathsf{TC}}_{e,\pi}(n)\geq c\cdot n$. It is clear that in such case, both Equation (6.1) and (6.2) are provable in PV. We set d_{tr} to be sufficiently large so that the feasibility of such PV proofs are bounded by $O((c\cdot n)^{d_{\mathsf{tr}}})$ or $O((c\cdot n+n_w)^{d_{\mathsf{tr}}})$. The cases when t is $\mathsf{ITR}(x,y)$ or $\circ(x,y)$ are similar.

Otherwise, consider the structure of the term t. For each t, we will consider the time complexity contribution of new PV proof lines in the following case study; namely, we do not count the time complexity contributions of PV proof from the induction hypothesis.

- (*Function via Composition*). Suppose that t is a function introduced by composition, i.e., t is of form $f_{t'}(\vec{x})$ for some term t', the term t' must be of lower height in (π, e). By the induction hypothesis, we can prove both Equations (6.1) and (6.2) for $t'(\vec{x})$. Notice that

$$[t]_{\mathsf{Cook}}^{|x_1|,\ldots,|x_k|} = [t']_{\mathsf{Cook}}^{|x_1|,\ldots,|x_k|}$$

  by the definition of $[\cdot]_{\mathsf{Cook}}$. This means that Equations (6.1) and (6.2) for the term t are identical to those for the term t', therefore Equations (6.1) and (6.2) for the term t can be proved in PV. This step does not contain any new PV proof lines, and therefore the contribution to time complexity is 0.

- (*Function via Limited Recursion*). Suppose that t is a function symbol f introduced by limited recursion from functions g, h_0, h_1, k_0, k_1. Without loss of generality, we assume that x_k is the induction variable. Note that these functions as well as the proof of the bounding equation

$$e_\sigma : \mathsf{ITR}(h_\sigma(x_1, \ldots, x_{k-1}, x_k, z), z \circ k_\sigma(x_1, \ldots, x_{k-1}, x_k)) = \varepsilon. \tag{6.5}$$

  are of lower height in (π, e). By the induction hypothesis, we can obtain PV proofs of Equations (6.1) and (6.2) for g, h_0, h_1, respectively. Recall that by the definition of $[\cdot]_{\mathsf{Cook}}$, we know that $[t]_{\mathsf{Cook}}^{|x_1|,\ldots,|x_k|}$ consists of the following components:

  (i) the translation of g on the input length $|x_1|, \ldots, |x_{k-1}|$;
  (ii) $|x_k|$ copies of the translation of $h_\sigma(\vec{x}, z)$ on the input length $|x_1|, \ldots, |x_k|, |z| = \ell_f(|x_1|, \ldots, |x_k|)$;
  (iii) corresponding intermediate variables such that the $|x_k|$ copies of the translation of $h_\sigma(\vec{x}, z)$ simulate the following computation:

$$\begin{aligned}
z_0 &:= g(x_1, \ldots, x_{k-1}) = f(x_1, \ldots, x_{k-1}, \varepsilon) \\
z_1 &:= h_{\sigma_1}(x_1, \ldots, x_{k-1}, \varepsilon, z_0) = f(x_1, \ldots, x_{k-1}, \sigma_1) \\
&\vdots \quad \vdots \\
z_{|x_k|} &:= h_{\sigma_{|x_k|}}(x_1, \ldots, x_{k-1}, \sigma_1\sigma_2\ldots\sigma_{|x_k|-1}, z_{|x_k|-1}) = f(x_1, \ldots, x_{k-1}, \sigma_1\sigma_2\ldots\sigma_{|x_k|}))
\end{aligned} \tag{6.6}$$

  where $x_k = \sigma_1\sigma_2\ldots\sigma_{|x_k|}$, and each z_i is of length $\ell_f(|x_1|, \ldots, |x_k|)$.

In a high level, the PV proof of Equations (6.1) and (6.2) is as follows. We prove by induction on i that the variables corresponding to z_i correctly compute the i-th line of Equation (6.6). This involves three components:

(i) The base case $i = 0$. This is provided by Equations (6.1) and (6.2) for the function symbol g.
(ii) In the induction case, we need to prove that the output length of RHS of the $(i+1)$-th line is at most $\ell_f(|x_1|, \ldots, |x_k|)$. This can be proved by Equation (6.5) and another induction in PV. The total time complexity contribution by Equation (6.5) is at most

$$(\mu^+_{e_\sigma, \pi_<}(|x_1|, \ldots, |x_{k-1}|, |x_k|, |z| := \ell_f(|x_1|, \ldots, |x_k|)))^{O(1)} \leq \beta^{\mathsf{TC}}_{e,\pi}(\vec{n})^{O(1)}, \tag{6.7}$$

  where $\pi_<$ denotes the prefix of the proof π up to the line e_σ, and $O(1)$ hides a constant that is independent of the term t.
  ⚠: We note that here $\vec{n}$ is not necessarily $|x_1|, \ldots, |x_k|$; indeed, the number of variables in the term t during the induction proof is not necessarily the same as the number of variables in e. For instance, we use above Equations (6.1) and (6.2) for the function symbol g, where g

only has $k-1$ variables. The number of variables may also increase, see the (*Composition*) case of the proof. Nevertheless, it can be verified that the inequality in Equation (6.7) still holds, as the changes in the number of variables and the input lengths are carefully captured in the definitions of ATC and AILs.

(iii) In addition, we need to prove that i-th line implies the $(i+1)$-th line. This is implied by the definition equation of f and Equations (6.1) and (6.2) for function symbols h_0 and h_1. The total time complexity contribution by the definition equation is at most

$$(\beta_t^+(t,\pi)(|x_1|,\ldots,|x_k|))^{O(1)} \leq \beta_{e,\pi}^{\mathsf{TC}}(\vec{n})^{O(1)},$$

where $O(1)$ hides a constant that is independent of t. The latter equations are available by the induction hypothesis of the outer induction in meta-theory.

The time complexity contribution of other equations, e.g., auxiliary equations to deal with the encoding and functionality of propositional formulas, is at most a fixed polynomial of $\beta_{e,\pi}^{\mathsf{TC}}(\vec{n})$. Therefore, if d_{tr} is a sufficiently large constant, the time complexity contribution of new proof lines is at most $O(\beta_{e,\pi}^{\mathsf{TC}}(\vec{n})^{d_{\mathsf{tr}}})$.

- (*Composition*). Suppose that t is a composition of a function symbol f and terms $s_1,\ldots,s_j$. Note that $f, s_1,\ldots,s_j$ are of lower height in (π, e), and therefore by the induction hypothesis, we can obtain PV proofs of Equations (6.1) and (6.2) for $f, s_1,\ldots,s_j$. By the definition of $[\cdot]_{\mathsf{Cook}}$, we know that $[t]_{\mathsf{Cook}}^{|x_1|,\ldots,|x_k|}$ consists of the following components:

  (i) for each $s \in \{s_1,\ldots,s_j\}$, the propositional translation of s on the input length $\vec{x}$;
  (ii) the propositional translation of $f(z_1,\ldots,z_j)$, where $|z_1| = \ell_{s_1}(\vec{n}),\ldots,|z_j| = \ell_{s_j}(\vec{n})$;
  (iii) corresponding intermediate variables that ensure the output variables of $s_1,\ldots,s_j$ are equal to the input variables of $f(z_1,\ldots,z_j)$.

  The proof of Equations (6.1) and (6.2) for the term $t \equiv f(s_1,\ldots,s_j)$ the following parts.

  (i) Equations (6.1) and (6.2) for terms $s_1,\ldots,s_j$. These are available by the induction hypothesis.
  (ii) Equations (6.1) and (6.2) for $f(z_1,\ldots,z_j)$ on the input length $|z_1| = \ell_{s_1}(\vec{n}),\ldots,|z_j| = \ell_{s_j}(\vec{n})$, which is available by the induction hypothesis.
  ⚠: Note that here the number of variables of the function symbol f is j, which might be different from k; see related discussion in (*Function via Limited Recursion*) case of the proof.
  (iii) Additional PV proof lines to deal with, e.g., the encoding and functionality of propositional formulas. The total time complexity contribution of these proof lines is at most a fixed polynomial of $\beta_{e,\pi}^{\mathsf{TC}}(\vec{n})$.

  Therefore, if d_{tr} is a sufficiently large constant, the time complexity contribution of new proof lines is at most $O(\beta_{e,\pi}^{\mathsf{TC}}(\vec{n})^{d_{\mathsf{tr}}})$.

Finally, we can see that the time complexity contribution of each new proof line throughout the induction proof is at most $O(\beta_{e,\pi}^{\mathsf{TC}}(\vec{n})^{d_{\mathsf{tr}}})$. As there are $O(1)$ lines in total, the time complexity of the entire proof is at most $O(\beta_{e,\pi}^{\mathsf{TC}}(\vec{n})^{d_{\mathsf{tr}}})$. Note that $O(\cdot)$ hides constants that are independent of $\vec{n}$ but may depend on π, e, which is allowed in the statement of Theorem 6.3.

Time complexity analysis of the last two parts. Next, we analyze the time complexity of the second and the third parts. The analysis of these two parts is similar, so we will analyze the second part in detail and only sketch the analysis of the third part.

Recall that in the second part, we need to prove the conditional equation $e_{t_1}^1(\vec{x}) \Rightarrow e_{t_2}^1(\vec{x}) \Rightarrow e_{\mathsf{tr}}^1(\vec{x})$; see Equation (6.3). The premises of the conditional equation include:

- $\mathsf{EQ}(\mathsf{Output}([t_1]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, \mathsf{AssignTerm}_{t_1}(\vec{x})), t_1(\vec{x}))$ in $e^1_{t_1}$;
- $\mathsf{EQ}(\mathsf{Output}([t_2]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, \mathsf{AssignTerm}_{t_2}(\vec{x})), t_2(\vec{x}))$ in $e^1_{t_2}$;
- $\mathsf{EQ}(t_1(\vec{x}), t_2(\vec{x}))$ in the premise of $e^1_{\mathsf{tr}}(\vec{x})$.

From these three equations, we can derive

$$\phi_{\mathsf{EQ}} : \mathsf{EQ}(\mathsf{Output}([t_1]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, \mathsf{AssignTerm}_{t_1}(\vec{x})), \mathsf{Output}([t_2]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, \mathsf{AssignTerm}_{t_2}(\vec{x}))),$$

and therefore we can include this equation as an additional premise. Note that this step is of a fixed polynomial time complexity overhead as we only use the transitivity of equations formalized by the function $\mathsf{EQ}(x, y)$.

Then we prove that

$$\phi_{\mathsf{EQ}} \Rightarrow \mathsf{Satisfy}([e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, \mathsf{Assign}_e(\vec{x})) = 1. \tag{6.8}$$

Recall that $\mathsf{Assign}_e(\vec{x})$ simply calls $\mathsf{AssignTerm}_{t_1}$, $\mathsf{AssignTerm}_{t_2}$ and computes the assignment of intermediate variables for comparing the output variables of $[t_1]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}$ and $[t_2]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}$. Equation (6.8) that formalizes the correctness of $\mathsf{Assign}_e(\vec{n})$ can be proved in PV provided $\mathsf{AssignTerm}_t$ for $t \in \{t_1, t_2\}$ and Assign_t are defined in a straightforward way. Moreover, it can be verified that the time complexity overhead is a fixed polynomial in the length of the propositional translation of e; in more detail, notice that $\mathsf{Assign}_e(\vec{x}), \mathsf{AssignTerm}_{t_1}(\vec{x}), \mathsf{AssignTerm}_{t_2}(\vec{x})$ runs in fixed polynomial time in the size of the propositional translation of e, and their correctness proofs are straightforward. We can then conclude that the second part incurs a fixed polynomial time complexity overhead in the size of the propositional translation of e, which is at most

$$\beta^{\mathsf{Cook}}_e(\vec{n})^{O(1)} \leq \beta^{\mathsf{TC}}_{e,\pi}(\vec{n})^{d_{\mathsf{tr}}}$$

when d_{tr} is chosen to be sufficiently large. The inequality follows by Theorem 4.3.

For the third part, notice that all the functions involved run in a fixed polynomial time in the length of the propositional translation of e, and the correctness proofs of the functions are straightforward. It can be verified that the third part incurs a fixed polynomial time complexity overhead in the size of the propositional translation of e, which is at most

$$\beta^{\mathsf{Cook}}_e(\vec{n})^{O(1)} \leq \beta^{\mathsf{TC}}_{e,\pi}(\vec{n})^{d_{\mathsf{tr}}}$$

when d_{tr} is chosen to be sufficiently large.

6.3 Feasible Proof Generation Theorem

We now proof the *feasible proof generation theorem*, which shows that if π is a valid PV proof of an equation e, not only the propositional translation of π is an EF proof of the propositional translation of e, but the fact is also provable in PV. Moreover, the proof of the fact is feasible in terms of its time complexity.

Let $\mathsf{IsEFProof}(\varphi, \pi)$ be a straightforward PV function that outputs 1 if π is an EF proof of φ, and outputs 0 otherwise. The feasible proof generation theorem states that:

Theorem 6.4 (feasible proof generation theorem)**.** *There is a constant $d_{\mathsf{gen}} \in \mathbb{N}$ such that the following holds. Let π be a* PV *proof concluding e. Then there is a* PV *proof π_{gen} of the equation*

$$\mathsf{IsEFProof}([e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, [\pi]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}) = 1 \tag{6.9}$$

such that $\mu^{\mathsf{TC}}_{\pi_{\mathsf{gen}}}(\vec{n}) = O((\mu^{\mathsf{TC}}_{\pi}(\vec{n}))^{d_{\mathsf{gen}}})$, where $\vec{x} = (x_1, \ldots, x_k)$ are variables in e.

Similar to Theorem 6.3, Theorem 6.4 can be proved by careful formalization and analysis of the standard correctness proof of the propositional translation [Coo75]. We will, however, present an alternative proof that utilizes the FPT algorithms developed in Section 5: Instead of proving Equation (6.9) for each fixed proof π, we will prove the generalization of Equation (6.9) where both e and π are given as *input variables*. Indeed, this proof will lead to a stronger result that will later be the key to proving the *feasibility hierarchy theorem*.

6.3.1 Algorithms for Propositional Translations

To state the more general version of Theorem 6.4, we need to first verify that the propositional translation $[\cdot]_{\mathsf{Cook}}$ of equations and proofs can be efficiently computed when the equations and proofs are given as input variables.

Recall that in Section 4.4, we presented the algorithm $\mathsf{GenProp}_{e,\pi}(\vec{n})$ that generates the propositional translation of e (in the context of π) and the algorithm $\mathsf{GenProof}_{\pi}(\vec{n})$ that generates the propositional translation of π. Indeed, it can be verified that both algorithms can be defined even if e, π are given as input rather than fixed in advance. Moreover, it can be verified that both algorithms run in polynomial time in their output length. By Theorem 4.3, we can conclude that $\mathsf{GenProp}_{e,\pi}(\vec{n})$ runs in time at most $\mathsf{poly}(\beta^{+}_{e,\pi}(\vec{n}))$, and $\mathsf{GenProof}_{\pi}(\vec{n})$ runs in time at most $\mathsf{poly}(\mu^{\mathsf{TC}}_{\pi}(\vec{n}))$.

Furthermore, similar to the functions for computing the time complexity of equations and proofs (see Section 5.7), the time complexity of both algorithms can be verified in PV. Let $\mathsf{GenProp}(\pi, e, \vec{n}, N)$ be the algorithm that simulates $\mathsf{GenProp}_{e,\pi}(\vec{n})$ for $|N|$ steps:

- If $\mathsf{GenProp}_{e,\pi}(\vec{n})$ does not halt in $|N|$ steps, it outputs ε.
- Otherwise, it outputs the output of $\mathsf{GenProp}_{e,\pi}(\vec{n})$.

Similarly, let $\mathsf{GenProof}(\pi, \vec{n}, N)$ be the algorithm that simulates $\mathsf{GenProof}_{\pi}(\vec{n})$ for $|N|$ steps. It is clear that these two functions can be defined in PV straightforwardly, and the time complexity of the functions is formulated as:

Lemma 6.5. *There is a constant $c \geq 1$ such that the following holds. Let $\mathsf{UB}_{\beta}(N, \pi, e, \vec{n})$ be the* PV *function that outputs 1 if $(\pi, e, \vec{n}, N)$ is a valid input of* GenProp *and $|N| \geq (\beta^{+}_{e,\pi}(\vec{n}))^{c}$, where $\vec{n}$ is encoded in binary. Let $\mathsf{IsNotEps}(x)$ be the* PV *function that outputs 1 if $x \neq \varepsilon$ and outputs 0 otherwise. Then* PV *proves the conditional equation:*

$$\mathsf{UB}_{\beta}(N, \pi, e, \vec{n}) \Rightarrow \mathsf{IsNotEps}(\mathsf{GenProp}(\pi, e, \vec{n}, N)) = 1. \tag{6.10}$$

Similarly, let $\mathsf{UB}_{\mu}(N, \pi, \vec{n})$ be the PV *function that outputs 1 if $(\pi, \vec{n}, N)$ is a valid input of* GenProof *and $|N| \geq (\mu^{\mathsf{TC}}_{\pi}(\vec{n}))^{c}$, then* PV *proves the conditional equation:*

$$\mathsf{UB}_{\mu}(N, \pi, \vec{n}) \Rightarrow \mathsf{IsNotEps}(\mathsf{GenProof}(\pi, \vec{n}, N)) = 1. \tag{6.11}$$

Proof Sketch. We formalize the complexity analysis of GenProp and GenProof in PV. Take Equation (6.10) for example. We first perform a case study on whether $\mathsf{UB}_{\beta}(N, \pi, e, t, \vec{n}) = 1$ or $\mathsf{UB}_{\beta}(N, \pi, e, t, \vec{n}) = 0$. The latter case can be resolved by the explosion principle (see Proposition 6.1).

For the former case, we prove by induction on $|M|$ that for any M such that $|M|^{c} \leq |N|$, after simulating $\mathsf{GenProp}_{e,\pi}(\vec{n})$ for $|M|^{c}$ steps, the algorithm has produced the first $|M|$ bits of $[e]^{\vec{n}}_{\mathsf{Cook}}$. This utilizes length induction on a feasibly checkable property, which is available in PV. Note that both the base case and the induction case can be proved in PV provided that the formalization of $\mathsf{GenProp}(e, \pi, \vec{n}, N)$ is straightforward. □

6.3.2 A More General Version

Now we state a more general version of Theorem 6.4. Let GenProp, GenProof be the functions introduced in Section 6.3.1. Let $\mathsf{ValidPV}(\pi, e)$ be the PV function that outputs 1 if π is a valid PV proof concluding e, and outputs 0 otherwise. Then:

Theorem 6.6 (feasible proof generation theorem, generalized version)**.**

$$\begin{aligned} \mathsf{PV} \vdash \mathsf{ValidPV}(\pi, e) &\Rightarrow \mathsf{IsNotEps}(\mathsf{GenProof}(\pi, \vec{n}, N)) \Rightarrow \mathsf{IsNotEps}(\mathsf{GenProp}(\pi, e, \vec{n}, N)) \\ &\Rightarrow \mathsf{IsEFProof}(\mathsf{GenProp}(\pi, e, \vec{n}, N), \mathsf{GenProof}(\pi, \vec{n}, N)) = 1. \end{aligned} \tag{6.12}$$

The key observation in the proof of Theorem 6.6 is that N is given as a part of the input. In such case, we could generate both $\phi := \mathsf{GenProp}(\pi, e, \vec{n}, N)$ and $\pi_{\phi} := \mathsf{GenProof}(\pi, \vec{n}, N)$ feasibly, and reason about

the correctness of both algorithms by looking at ϕ, π_ϕ. This allows us to formalize the standard proof of the correctness of the propositional translation [Coo75].

Proof. We argue in PV. Fix π, e, $\vec{n}$, and N. Suppose that $\mathsf{GenProp}(\pi, e, \vec{n}, N) \neq \varepsilon$ and $\mathsf{GenProof}(\pi, e, \vec{n}, N) \neq \varepsilon$, i.e., $\mathsf{GenProp}(\pi, e, \vec{n})$ and $\mathsf{GenProof}(\pi, e, \vec{n}, N)$ simulate $\mathsf{GenProp}_{e,\pi}(\vec{n})$ and $\mathsf{GenProof}_{e,\pi}(\vec{n})$ for $|N|$ steps and both of them halt. Let ϕ be the output of $\mathsf{GenProp}(\pi, e, \vec{n})$ and π_ϕ be the output of $\mathsf{GenProof}(\pi, e, \vec{n}, N)$. Our goal is to prove that π_ϕ is an EF proof of ϕ.

Let $\pi : e_1, \ldots, e_m$ where $e_m = e$, $\Delta := \mu\text{-}\mathsf{AM}_\pi(\{e_m, \vec{n}\})$ be the acquisition map, $\Delta_i := \{(e_i, \vec{p}) \in \Delta \mid \vec{m} \in \mathbb{N}\}$. Note that both Δ and Δ_i can be computed feasibly as we have N in hand and the running time of the entire algorithm $\mathsf{GenProof}_{e,\pi}(\vec{n})$ is at most $|N|$.

Recall that in the i-th round of the algorithm $\mathsf{GenProof}_{e,\pi}(\vec{n})$, for each $\vec{p} \in \Delta_i$, the algorithm will output EF proof lines such that:

- All EF proof lines that have been written down by the algorithm form a valid EF proof.
- If e_i is an equation (i.e. it is not for the introduction of a new function symbol), the propositional translation of e_i on the input length $\vec{p}$ is proved in the EF proof lines.

We will prove by induction on i that these two properties hold. Notice that the induction principle is available in PV as $m \leq |\pi|$, and both bullets can be verified by straightforward polynomial-time algorithms given π, i and the EF proof lines written by the algorithm. The base case $i = 0$ is trivial as there is no EF proofs that have been written down and no e_0.

Now we consider the induction case. Suppose that the algorithm have written down valid EF proof lines in the first $i-1$ rounds, and for any $j < i$ and $\vec{p} \in \Delta_j$, the propositional translation of e_j on the input length $\vec{p}$ is proved in the EF proof lines. In the i-th round, the algorithm considers the deduction rule that introduces the new proof lines e_i.

- (*Axioms for Functions*). Suppose that e_i is a proof line introduced as an axiom for the functionality of initial functions or introduced functions. In all cases, the propositional translation of e_i admits an explicit EF proof, and the algorithm outputs the proof. The correctness of the generated EF proof is provable in PV provided that it is formalized in a straightforward fashion.

- (*Function Introduction*). Suppose that e_i is a proof line that introduces a function symbol by the composition rule or the limited recursion rule, the algorithm outputs nothing in this round and the correctness is trivial.

- (*Logical Rules*). Suppose that $e_i : t = s$ is a proof line derived from a logical rule, say (L1). (The proofs for other logical rules are similar and therefore omitted.)

  In such case, there is a proof line $e_j : s = t$ for $j < i$. We need to prove $\vec{p} \in \Delta_j$, which is deferred to the end of the proof (see $(\diamond)$). Therefore, by the induction hypothesis, the propositional translation of e_j on the input length $\vec{p}$ has been written by the algorithm.

  There is an explicit EF proof concluding the propositional translation of e_i on the input length $\vec{p}$ from the propositional translation of e_j on the input length $\vec{p}$, and the algorithm $\mathsf{GenProof}$ produces the proof in the i-th round. The correctness of the algorithm (i.e. the two bullets above) in the i-th round can be formalized in PV provided that it is formalized straightforwardly.

- (*Induction*). Suppose that e_i is derived from the induction rule. Without loss of generality, we assume that x_k is the induction variable and $\vec{x} = (x_1, \ldots, x_k)$ are all variables. Let $f_1(\vec{x})$,$f_2(\vec{x})$, $g(x_1, \ldots, x_{k-1})$, $h_i(\vec{x}, z)$ be PV function symbols that have been introduced in $\pi_{<i}$. Moreover, for $\sigma \in \{1, 2\}$, $b \in \{0, 1\}$, the equations

$$e'_g : f_1(x_1, \ldots, x_{k-1}, \varepsilon) = f_2(x_1, \ldots, x_{k-1}, \varepsilon)$$
$$e'_{\sigma,b} : f_\sigma(x_1, \ldots, x_{k-1}, s_b(x_k)) = h_b(x_1, \ldots, x_{k-1}, x_k, f_\sigma(x_1, \ldots, x_k))$$

  are in $\pi_{<i}$, from which $e : f_1(\vec{x}) = f_2(\vec{x})$ is derived.

Suppose that e'_g is in the j_g-th line of π, $e'_{\sigma,i}$ is in the $j_{\sigma,i}$-th line of π. We first prove that

$$(p_1, \ldots, p_{k-1}) \in \Delta_{j_g}, \quad \text{and} \quad (p_1, \ldots, p_k) \in \Delta_{j_{\sigma,b}}$$

for every $\sigma \in \{1,2\}$ and $b \in \{0,1\}$; this is deferred to the end of the proof, see $(\diamond)$. Therefore, by the induction hypothesis, the algorithm has written down the EF proof of the propositional translation of e'_g on the input length $(p_1, \ldots, p_{k-1})$, as well as the propositional translation of $e'_{\sigma,b}$ on the input length $\vec{p}$.

The algorithm produces new proof lines concluding the propositional translation of e on the input length $\vec{p}$, see Section 4.4. Recall that the EF proof introduces new variables $z_1, \ldots, z_{p_k}$ and the following constraints: For each $j \in \{0, 1, \ldots, p_k\}$, there is a 3-CNF ϕ_j that ensures that if ϕ_k is satisfied, then

$$z_j = \begin{cases} f_1(x_1, \ldots, x_k, \varepsilon) & \text{if } i = 0; \\ h_{\sigma_i}(x_1, \ldots, x_k, z_{j-1}) & \text{otherwise;} \end{cases}$$

where σ_i is the i-th leftmost bit of x_k; ψ_j be a 3-CNF that is satisfied if and only if

$$z_j = f_1(x_1, \ldots, x_k, x_{k,\le j}) = f_2(x_1, \ldots, x_{k-1}, x_{k,\le j})$$

where $x_{k,\le j}$ denotes the prefix of x_k of length j. Then, the algorithm produces the EF proof of $\alpha_j := \phi_1 \wedge \cdots \wedge \phi_j \to \psi_j$ for each $j \in \{0, 1, \ldots, p_k\}$, where the proof of α_0 is explicit, and the EF proof of α_{j+1} follows from the proof of α_j. This can be formalized in PV with the induction principle, as the total length of the induction is at most $|N|$, and the property (i.e. the algorithm produced the EF proof of α_j) can be verified in polynomial time.

Proof of $(\diamond)$: AILs and acquisition closure. Finally, we explain how to deal with the acquired input lengths in previous rounds of the algorithm, which is necessary if e_i is introduced by a deduction rule with premise(s). Take the logical rule (L1) for example. Suppose that $e_i : t = s$ and $e_j : s = t$ for $j < i$, we need to prove that if $\vec{p} \in \Delta_i$, then $\vec{p} \in \Delta_j$.

By induction on the algorithm that generates the acquisition closure $\Delta = \mu\text{-}\mathsf{AM}_\pi(\{e_m, \vec{n}\})$, we can prove that Δ is closed under acquisition extension. This induction proof closely follows the execution of the algorithm and is therefore available in PV. Notice that if $\vec{p} \in \Delta_i$, $e_i : t = s$, and $e_j : s = t$, by the definition of AILs, we have $\vec{p} \in \mu\text{-}\mathsf{AIL}_{e_i,e_j}(\vec{p})$. This implies that $(e_j, \vec{p}) \in \Delta$ as Δ is closed under acquisition extension, which further implies that $\vec{p} \in \Delta_j$ by the definition of Δ_j. □

6.3.3 Proof of Theorem 6.4

Now we prove Theorem 6.4 from the generalized Theorem 6.6. Let $d_{\mathsf{gen}} \in \mathbb{N}$ be a constant to be determined later. Fix any PV proof π concluding e. We first prove the following lemma:

Lemma 6.7. *There are constants $d_1, d_2 \in \mathbb{N}$ such that for any π, e, there are constants $c_1, c_2 \in \mathbb{N}$ and PV proofs of the equations:*

$$e_\pi : \mathsf{GenProof}\left(\pi, \vec{n}, 1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}\right) = [\pi]^{\vec{n}}_{\mathsf{Cook}}, \qquad e_p : \mathsf{GenProp}\left(\pi, e, \vec{n}, 1^{c_1 \cdot (\beta^+_{e,\pi}(\vec{n}))^{d_1}}\right) = [e]^{\vec{n}}_{\mathsf{Cook}},$$

where $\vec{n}$ is the shorthand of $|x_1|, \ldots, |x_k|$. Moreover, the PV proofs of e_π and e_p are of time complexity at most

$$c_2 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_2}, \qquad c_2 \cdot (\beta^+_{e,\pi}(\vec{n}))^{d_2},$$

respectively.

Proof. We only prove the case for e_π as the proof for e_p is similar. Recall that the PV function $\mathsf{GenProof}(\pi, \vec{n}, N)$ simulates the algorithm generating $[\pi]^{\vec{n}}_{\mathsf{Cook}}$ for $|N|$ steps. The PV proofs of e_π is as follows:

- (*Time Complexity*). By Lemma 6.5, it is provable in PV that

$$\mathsf{UB}_\mu(N, \pi, \vec{n}) \Rightarrow \mathsf{IsNotEps}(\mathsf{GenProof}(\pi, \vec{n}, N)) = 1. \tag{6.13}$$

  By applying (L4), we substitute N by $1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}$ and obtain

$$\mathsf{UB}_\mu(1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}, \pi, \vec{n}) \Rightarrow \mathsf{IsNotEps}(\mathsf{GenProof}(\pi, e, \vec{n}, 1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}})) = 1. \tag{6.14}$$

- (*Correctness*). Similar to the proof of Lemma 6.5, we prove in PV the correctness of the algorithm $\mathsf{GenProof}$, formalized as the following equation:

$$\mathsf{IsNotEps}(\mathsf{GenProof}(\pi, \vec{n}, N)) \Rightarrow \mathsf{GenProof}(\pi, \vec{n}, N) = [\pi]^{\vec{n}}_{\mathsf{Cook}}, \tag{6.15}$$

  where π is the fixed PV proof in the statement of the lemma (i.e. not a variable), and $\vec{n}$ is the shorthand of $|x_1|, \ldots, |x_k|$. By applying (L4), we substitute N by $1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}$ and obtain

$$\mathsf{IsNotEps}(\mathsf{GenProof}(\pi, e, \vec{n}, 1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}})) \Rightarrow \mathsf{GenProof}(\pi, \vec{n}, 1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}) = [\pi]^{\vec{n}}_{\mathsf{Cook}}. \tag{6.16}$$

- (*Upper Bound*). Next, we prove that

$$\mathsf{UB}_\mu(1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}, \pi, \vec{n}) = 1, \tag{6.17}$$

  where π is the fixed PV proof in the statement of the lemma rather than a variable.

- (*Modus Ponens*). We apply Modus Ponens twice on Equations (6.14), (6.16) and (6.17), concluding e_π.

By feasible deduction theorem (see Theorem 6.2), it suffices to analyze the time complexity of Equations (6.14), (6.16) and (6.17) separately. We argue that in all cases, the time complexity of the PV proofs are a fixed polynomial independent of π in the running time of the terms (i.e. the LHS and RHS of the equations).

Take Equation (6.16) as an example. The proof of Equation (6.16) follows from Equation (6.15) by applying (L4), and therefore the total time complexity is the summation of two terms:

- The ATC of Equation (6.16), which is a fixed polynomial in the running time of the LHS and RHS of the equation.
- The time complexity of Equation (6.15) on the input length $|N| = c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}$. We argue that the time complexity of Equation (6.15) as a function of $(\vec{n}, N)$ is $\mathsf{polylog}(|\vec{n}|) \cdot N^{d'} \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d'}$ for a fixed constant d' that is independent of π. To see this, notice that $\mathsf{GenProof}(\pi, \vec{n}, N)$ is a straightforward simulation of the algorithm $[\pi]^{\vec{n}}_{\mathsf{Cook}}$ for $|N|$ steps, and the correctness of the simulation can be proved by induction on $|N|$. Provided that the simulation is straightforward, the proof should be of the same structure for any proof π.

As the running time of the LHS and RHS of Equation (6.16) are both a fixed polynomial in the time complexity of π, we can conclude that the time complexity of the proof of Equation (6.16) is a fixed polynomial in the time complexity of π. The lemma follows by choosing c_1, c_2, d_1, d_2 to be sufficiently large constants. □

Theorem 6.4 (feasible proof generation theorem)**.** *There is a constant $d_{\mathsf{gen}} \in \mathbb{N}$ such that the following holds. Let π be a* PV *proof concluding e. Then there is a* PV *proof π_{gen} of the equation*

$$\mathsf{IsEFProof}([e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, [\pi]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}) = 1 \tag{6.9}$$

such that $\mu^{\mathsf{TC}}_{\pi_{\mathsf{gen}}}(\vec{n}) = O((\mu^{\mathsf{TC}}_\pi(\vec{n}))^{d_{\mathsf{gen}}})$, where $\vec{x} = (x_1, \ldots, x_k)$ are variables in e.

Proof. The PV proof is as follows. Let $\vec{n}$ be the shorthand of $|x_1|, \ldots, |x_k|$. By Lemma 6.7, we first obtain the proof of the following equations:

$$e_\pi : \mathsf{GenProof}\left(\pi, \vec{n}, 1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}\right) = [\pi]^{\vec{n}}_{\mathsf{Cook}}, \qquad e_p : \mathsf{GenProp}\left(\pi, e, \vec{n}, 1^{c_1 \cdot (\beta^+_{e,\pi}(\vec{n}))^{d_1}}\right) = [e]^{\vec{n}}_{\mathsf{Cook}},$$

where c_1, d_1 are constants independent of π. By Lemma 6.5, we can also prove that

$$e_{\pi,c} : \mathsf{IsNotEps}\left(\mathsf{GenProof}\left(\pi, \vec{n}, 1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}\right)\right) = 1;$$
$$e_{p,c} : \mathsf{IsNotEps}\left(\mathsf{GenProp}\left(\pi, e, \vec{n}, 1^{c_1 \cdot (\beta^+_{e,\pi}(\vec{n}))^{d_1}}\right)\right) = 1.$$

Note that the time complexity of each proof is a fixed polynomial in $c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}$ by Lemmas 6.5 and 6.7.

By Theorem 6.6, we can prove in PV the equation:

$$e'_{\mathsf{FGT}} : \mathsf{ValidPV}(\pi, e) \Rightarrow \mathsf{IsNotEps}(\mathsf{GenProof}(\pi, \vec{n}, N)) \Rightarrow \mathsf{IsNotEps}(\mathsf{GenProp}(\pi, e, \vec{n}, N))$$
$$\Rightarrow \mathsf{IsEFProof}(\mathsf{GenProp}(\pi, e, \vec{n}, N), \mathsf{GenProof}(\pi, \vec{n}, N)) = 1,$$

We apply (L4) rule to substitute the variable π by the fixed PV proof π in the statement of the theorem, e by the fixed PV equation e in the statement, and N by $1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}$. Denote the equation by e_{FGT}. Note that the time complexity of the proof of e_{FGT} consists of two parts:

- The ATC of the (L4) rule to substitute N by $1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}$, which is a fixed polynomial in $1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}$.
- The time complexity of the proof of e'_{FGT}, which is a fixed polynomial in $|\pi|, |e|, |\vec{n}|, |N|$. On the input length $|N| = 1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}$, the time complexity will be a fixed polynomial in $c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}$.

By applying Modus Ponens on e_{FGT}, $e_{\pi,c}$, and $e_{p,c}$, we can obtain the equation

$$e''_{\mathsf{FGT}} : \mathsf{IsEFProof}\left(\mathsf{GenProof}\left(\pi, \vec{n}, 1^{c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}}\right), \mathsf{GenProp}\left(\pi, e, \vec{n}, 1^{c_1 \cdot (\beta^+_{e,\pi}(\vec{n}))^{d_1}}\right)\right) = 1.$$

This step incurs little time complexity overhead by the feasible deduction theorem (see Theorem 6.2). Finally, we apply the logical rule (L3) to e''_{FGT}, e_π, and e_p to conclude

$$\mathsf{IsEFProof}([e]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}, [\pi]^{|x_1|,\ldots,|x_k|}_{\mathsf{Cook}}) = 1.$$

The time complexity overhead of this step is the ATC of e''_{FGT}, e_π, and e_p, which is a fixed polynomial in $c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}$.

In conclusion, the total time complexity overhead of the proof is a fixed polynomial (i.e. the exponent is independent of π) in $c_1 \cdot (\mu_\pi^{\mathsf{TC}}(\vec{n}))^{d_1}$. This completes the proof if we choose d_{gen} to be a sufficiently large constant that is independent of π. □

7 Feasibility Hierarchy Theorems

We now prove the feasibility hierarchy theorem. Intuitively, it states that there is a constant d such that for any constant $c \geq 1$, there is a PV equation e_d with time complexity β such that:

- (*Upper Bound*). There is a PV proof of e_d with time complexity $O(\beta^{cd})$.
- (*Lower Bound*). There is no PV proof of e_d of time complexity $o(\beta^c)$.

That is, PV proofs with higher time complexity are strictly stronger than those with lower time complexity.

Indeed, the equation e_d is quite simple: it takes two variables, and its time complexity β is a fixed polynomial, i.e., the exponent is independent of d. More formally:

Theorem 7.1 (feasibility hierarchy theorem)**.** *There are constants $c_1, c_2 \geq 1$ such that the following holds. For any constant $d \geq 1$, there is a* PV *equation $e_d : s_d(\pi, x) = t_d(\pi, x)$ and its context π_d such that:*

1. *The time complexity of the statement e_d is at most $\beta := \beta^{\mathsf{TC}}_{e_d,\pi_d}(|\pi|, |x|) = O_d((|\pi| + |x|)^{c_1})$.*
2. *e_d admits a* PV *proof of time complexity $O_d(\beta^{c_2 \cdot d})$;*
3. *e_d is unprovable in* PV *with time complexity $o(\beta^d)$.*

7.1 Bounded Feasibility Consistency

Fix $d \geq 1$ to be a constant. We consider the following statement: For every PV proof π of concluding $s_0(x) = s_1(x)$, the time complexity of π on the input length $|x|$ must be at least $d \cdot |x|^d$. This statement is called the bounded feasibility consistency of PV, as it can be considered as the consistency of the fragment of PV with feasibility $d \cdot |x|^d$.

This statement cannot be formalized directly in PV as the computation of the time complexity of π takes super-polynomial time in $|\pi|$. Nevertheless, as there is an FPT algorithm to compute the time complexity of proofs (see Theorem 5.10) in $\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(n)$ time, the statement can be formalized in PV if, say, $\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(n) \leq n$.

In more detail, let $f_\mu(\pi, x)$ be the PV function that simulates the algorithm $A_\mu(\pi, |x|)$ in Theorem 5.10 for $|x|$ steps such that

- If $A_\mu(\pi, |x|)$ does not halt after $|x|$ steps, $f_\mu(\pi, x)$ outputs ε.
- Otherwise, it outputs the output of $A_\mu(\pi, |x|)$.

(See Section 5.7 for related discussion.) We define the bounded feasibility consistency statement as follows:

Definition 7.1 (bounded feasibility consistency)**.** Let $\mathsf{ValidPV}(\pi, e)$ be the PV function that outputs 1 if π encodes a valid PV proof concluding e, and outputs 0 otherwise. Let $[\![e]\!]$ be the encoding of a PV equation e, and $[\![\pi]\!]$ be the encoding of a PV proof π. Let $\mathsf{UB}_{\mu,d}$ be the PV function that outputs 1 if $f_\mu(\pi, x) \leq d \cdot |x|^d$, and outputs 0 otherwise.

The *bounded feasibility consistency* statement $\mathsf{Con}_d(\pi, x)$ is defined as the following PV conditional equation:

$$\mathsf{IsNotEps}(f_\mu(\pi, x)) \Rightarrow \mathsf{UB}_{\mu,d}(\pi, x) \Rightarrow \mathsf{ValidPV}(\pi, [\![s_0(x) = s_1(x)]\!]) = 0.$$

Proposition 7.2. *There is a constant $c_{\mathsf{Con}} \geq 1$ such that for any $d \geq 1$, the time complexity of the equation $\mathsf{Con}_d(\pi, x)$ is at most $O_d(n^{c_{\mathsf{Con}}})$, where $n = |\pi| + |x|$.*

Proof Sketch. Recall that the time complexity of an equation e in the context π is defined as its ATC $\beta^+_{e,\pi}(\vec{n})$ plus the time complexity contribution of proof lines in π — the time complexity of the proofs of the bounding inequalities for function symbols in e.

We first analyze the first former part. The ATC of $\beta^+_{e,\pi}(\vec{n})$ is a fixed polynomial in the running time of the terms in e provided that e is defined straightforwardly. As the running time of $\mathsf{Con}_d(\pi, x)$ is a fixed polynomial (i.e. the exponent is independent of d), the ATC of the equation $\mathsf{Con}_d(\pi, x)$ should also be a fixed polynomial.

It remains to analyze the latter part. Notice that the only occurrence of the constant d is in the term $\mathsf{UB}_{\mu,d}(\pi, x)$ to check whether $f_\mu(\pi, x) \leq |x|^d$, and the complexity analysis of the function $\mathsf{UB}_{\mu,d}$ (where we may need to introduce functions via limited recursion) is almost the same for any d. It can then be verified that the total time complexity contribution of the context of $\mathsf{UB}_{\mu,d}$ is a fixed polynomial (i.e. the exponent is independent of d). This completes the proof. □

We now prove the key technical lemma that $\mathsf{Con}_d(\pi, x)$ can be proved in PV with time complexity at most $n^{O(d)}$. Formally:

Lemma 7.3 (provability of bounded feasibility consistency)**.** *There is a constant $c \geq 1$ such that for any $d \geq 1$, the equation $\mathsf{Con}_d(\pi, x)$ admits a* PV *proof π_{Con} of time complexity $O_d((|\pi| + |x|)^{cd})$.*

Proof. Let $c \geq 1$ be a constant to be determined later. Fix any $d \geq 1$. We first describe a PV proof π_{Con} of $\mathsf{Con}_d(\pi, x)$ and then analyze the time complexity of the proof.

Proof of $\mathsf{Con}_d(\pi, x)$**.** We argue in PV. The main idea is as follows. Assume that $\mathsf{IsNotEps}(f_\mu(\pi, x)) = 1$, $\mathsf{UB}_{\mu,d}(\pi, x) = 1$, and $\mathsf{ValidPV}(\pi, [\![s_0(x) = s_1(x)]\!]) = 1$, our goal is to derive a contradiction. We can apply the generalized feasible proof generation theorem (see Theorem 6.4) to obtain an EF proof of $[s_0(x) = s_1(x)]^{|x|}_{\mathsf{Cook}}$, which contradicts with the soundness of EF (see Theorem 3.2).

In more detail, the proof consists of the following steps:

(1) Let $e_\perp$ be the equation $s_0(x) = s_1(x)$. We first show that there is a constant c_1 such that

$$e_p : \mathsf{IsNotEps}(\mathsf{GenProp}(\pi, e_\perp, n, 1^{c_1 \cdot n^{c_1}})) = 1,$$

where $n := |x|$, and the equation is provable in PV. This is obvious, as the PV function $\mathsf{GenProp}(\pi, e_\perp, n, c_1 \cdot n^{c_1})$ simulates the algorithm computing $[e_\perp]^n_{\mathsf{Cook}}$ for $c_1 \cdot n^{c_1}$ steps and the simulation should provably halt in fixed polynomial time provided that GenProp is defined straightforwardly.

(2) We prove that there is a constant $c_2 \geq 1$ such that PV proves

$$\begin{aligned} e_\pi : \mathsf{IsNotEps}(f_\mu(\pi, x)) \Rightarrow \mathsf{UB}_{\mu,d}(\pi, x) \Rightarrow \mathsf{ValidPV}(\pi, e_\perp) \\ \Rightarrow \mathsf{IsNotEps}(\mathsf{GenProof}(\pi, n, 1^{c_2 \cdot n^{c_2 \cdot d}})) = 1, \end{aligned}$$

where $n := |x|$ and $e_\perp := [\![s_0(x) = s_1(x)]\!]$.

Indeed, we can prove a more general statement: Let z be a fresh variable, if the time complexity of the proof π is at most $|z|$, the algorithm $\mathsf{GenProof}(\pi, n, 1^{c_2 \cdot |z|^{c_2}})$ does not output ε, which means that the algorithm computing $[\pi]^{|z|}_{\mathsf{Cook}}$ halts in $c_2 \cdot |z|^{c_2}$ steps. Let e'_π be this statement. We can then prove e_π by applying the logical rule (L4) to substitute the variable z with 1^{n^d}.

The generalized statement e'_π can be proved in PV as follows. Recall that $\mathsf{GenProof}(\pi, n, |z|)$ simulates the algorithm generating $[\pi]^n_{\mathsf{Cook}}$ for $|z|$ steps. We first prove that during the simulation, the total length of the proof lines written down by the algorithm is at most the time complexity of π. This can be proved by formalizing Theorem 4.3 in PV, which is straightforward as the proof closely follows from the execution of the algorithm $\mathsf{GenProof}(\pi, n, |z|)$. Therefore, as the time complexity of π is at most $|z|$, the total length of the proof lines written down by the algorithm simulated by $\mathsf{GenProof}(\pi, n, |z|)$ is at most $|z|$. Finally, we show that the simulation is efficient and incurs only a fixed polynomial time overhead, which is provable in PV provided that GenProof is formalized straightforwardly.

(3) Recall that by Theorem 6.6, PV proves

$$\begin{aligned} e_{FGT} : \mathsf{ValidPV}(\pi, e) \Rightarrow \mathsf{IsNotEps}(\mathsf{GenProof}(\pi, n, N)) \Rightarrow \mathsf{IsNotEps}(\mathsf{GenProp}(\pi, e, n, N)) \\ \Rightarrow \mathsf{IsEFProof}(\mathsf{GenProp}(\pi, e, n, N), \mathsf{GenProof}(\pi, n, N)) = 1, \end{aligned}$$

where $n := |x|$. We substitute N with $1^{c_2 \cdot n^{c_2 \cdot d}}$, e with $e_\perp := [\![s_0(x) = s_1(x)]\!]$, and denote the resulting equation by e'_{FGT}. Combining e_p, e_π, and e'_{FGT}, we can derive that

$$\begin{aligned} e' : \mathsf{IsNotEps}(f_\mu(\pi, x)) \Rightarrow \mathsf{UB}_{\mu,d}(\pi, x) \Rightarrow \mathsf{ValidPV}(\pi, e_\perp) \\ \Rightarrow \mathsf{IsEFProof}(\mathsf{GenProp}(\pi, e_\perp, 1^{c_1 \cdot n^{c_1 \cdot d}}), \mathsf{GenProof}(\pi, n, 1^{c_2 \cdot n^{c_2 \cdot d}})) = 1 \end{aligned}$$

(4) Recall that by Theorem 3.2, PV proves the soundness of EF. Assuming any straightforward encoding of propositional formulas, the propositional translation of $e_\perp : s_0(x) = s_1(x)$ should imply a contradiction. Therefore, we can prove in PV that:

$$e_s : \mathsf{IsEFProof}(\mathsf{GenProp}(\pi, e_\perp, 1^{c_1 \cdot n^{c_1}}), \tau) = 0,$$

where $n := |x|$. Namely, there is no EF proof of the statement $e_\perp$.

(5) We substitute τ with $\mathsf{GenProof}(\pi, n, 1^{c_2 \cdot n^{c_2 \cdot d}})$ to obtain

$$e'_s : \mathsf{IsEFProof}(\mathsf{GenProp}(\pi, e_\perp, 1^{c_1 \cdot n^{c_1}}), 1^{c_2 \cdot n^{c_2 \cdot d}}) = 0.$$

We can then conclude $\mathsf{Con}_d(\pi, x)$ by combining e'_s and e'.

Time complexity analysis. Now we analyze the time complexity of the proof above by considering the time complexity contribution of each step. Recall that $\mathsf{Con}_d(\pi, x)$ has two variables π and x. Let n_π and n_x be the input length of π and x, respectively.

(1) In the first step, we prove the equation e_p that has two variables π and x. The only acquired input length of e_p is $|\pi| = n_\pi$ and $(n =)|x| = n_x$. Moreover, as the proof closely follows the computation of the function $\mathsf{GenProp}(\pi, e_\perp, n, 1^{c_1 \cdot n^{c_1}})$, the total ATC of the proof lines should be a fixed polynomial in the running time of the algorithm, which is a fixed polynomial in n_π and n_x (i.e. the exponent does not depend on d).

(2) In the second step, we prove the equation e_π about the algorithm $\mathsf{GenProof}(\pi, n, 1^{c_2 \cdot n^{c_2 \cdot d}})$. The equation involves two variables π and x, and the only acquired input length is $|\pi| = n_\pi$ and $|x| = n_x$. Similar to the first step, the total ATC of the proof lines should be a fixed polynomial in the running time of the algorithm, which is a fixed polynomial in n_π and n_x^d.

(3) In the third step, we prove the equation e_{FGT} (and e'_{FGT} via substitution). The time complexity contribution of e'_{FGT} is negligible as it is obtained from e_{FGT} via (L4) substitution. The equation e_{FGT} has four variables π, e, x, N, and the only acquired input length is

$$|\pi| = n_\pi, \quad |e| = O(1), \quad |x| = n_x, \quad |N| = c_2 \cdot n^{c_2 \cdot d}. \tag{7.1}$$

Note that as e_{FGT} admits a PV proof (given in Theorem 6.6) that is independent of d, the time complexity of e_{FGT} is a fixed polynomial in its input length. In this case, i.e., the input length is Equation (7.1), the total time complexity contribution is a fixed polynomial in $(n_\pi + n_x)^d$.

(4) In the fourth step, we prove the equation e_s consisting of three variables π, τ, and x. The only acquired input length is

$$|\pi| = n_\pi, \quad |\tau| = \ell_{\mathsf{GenProof}}(n_\pi, n_x, c_2 \cdot n_x^{c_2 \cdot d}), \quad |x| = n_x.$$

Here, $\ell_{\mathsf{GenProof}}(\pi, x, c_2 \cdot n^{c_2 \cdot d}$ is the bounding value of GenProof; as GenProof is a PV function independent of d, its bounding value is a fixed polynomial in its input, and thus $|\tau|$ is a fixed polynomial in $(n_\pi + n_x)^d$. Similar to the third step, the total time complexity contribution is a fixed polynomial in its input length, which is a fixed polynomial in $(n_\pi + n_x)^d$.

(5) The fifth step involves only substitution and logical consequences about conditional equations, and it can be verified that the total time complexity contribution is at most a fixed polynomial in the running time of the terms, which is, in turn, a fixed polynomial in $(n_\pi + n_x)^d$.

In summary, the time complexity contribution of each part of the proof is a fixed polynomial in $(n_\pi + n_x)^d$. This completes the proof if we set the constant c to be sufficiently large. □

7.2 Proof of the Feasibility Hierarchy Theorem

Now we are ready to prove the feasibility hierarchy theorem. For any constant $d \geq 1$, the equation e_d will be constructed by modifying the self-referential statement in Gödel's incompleteness theorem [Göd31] (see also [Coo75] for the construction in PV).

We start by defining a few functions in PV:

- Let π_1, π_2 be PV proofs, we use $\langle \pi_1, \pi_2 \rangle$ to denote the concatenation of π_1 and π_2.
- Let $f_{\mu,2}(\pi, x)$ be the function that simulates $A_\mu(\pi, |x|, |x|)$ in Theorem 5.10 for $|x|$ steps (i.e. it attempts to compute the time complexity of π when the input length of two variables are both $|x|$).
- Let $f_{\beta,2}(\pi, e, x)$ be the function that simulates $A_\beta(\pi, e, |x|, |x|)$ in Theorem 5.11 for $|x|$ steps (i.e. it attempts to compute the time complexity of the equation e in the context π when the input length of two variables are both $|x|$).
- Let $\mathsf{UB}_{\mu,d,2}(\pi, \tau, x)$ be the PV function that outputs 1 if τ encodes a pair (π_e, e), where e encodes an equation in the context π_e, $\langle \pi_e, \pi \rangle$ is a valid PV proof of e, and both $f_{\mu,2}(\langle \pi_e, \pi \rangle, x)$ and $f_{\beta,2}(\pi_e, e, x)$ successfully simulates corresponding algorithms, and

  $$f_{\mu,2}(\langle \pi_e, \pi \rangle, x) \leq (f_{\beta,2}(\pi_e, e, x))^d.$$

  It outputs 0 otherwise.
- Let $\mathsf{Subst}(\tau)$ be a straightforward PV function such that
  - If τ does not encode a pair (π_e, e), where e is a PV equation in the context π_e with at least one variable, we define $\mathsf{Subst}(\pi_e, e) := \varepsilon$.
  - Suppose τ encodes a pair (π_e, e), where e is a PV equation $t_1(x, \vec{y}) = t_2(x, \vec{y})$ with at least one variable in the context π_e. Let x be the variable with the smallest index,

    $$\mathsf{Subst}(\tau) := (\llbracket \pi_e \rrbracket, \llbracket t_1(\tau, \vec{y}) = t_2(\tau, \vec{y}) \rrbracket).$$

    That is, it outputs the encoding of the equation $e[x/\tau]$, where x denotes the first variable in e.

Let $r_d : \mathsf{UB}_{\mu,d,2}(\pi, \mathsf{Subst}(y), x) = 1$ be an equation and p_d be its context, where y is the variable with the smallest index in r_d. Let τ be the encoding of the pair (r_d, p_d), and

$$q_d := \mathsf{Subst}(\tau) = (\llbracket p_d \rrbracket, \llbracket \mathsf{UB}_{\mu,d,2}(\pi, \mathsf{Subst}(\tau), x) = 1 \rrbracket).$$

This means that

$$q_d := (\llbracket p_d \rrbracket, \llbracket \mathsf{UB}_{\mu,d,2}(\pi, q_d, x) = 1 \rrbracket);$$

that is, q_d encodes a pair (π_e, e), where e is an equation in the context π_e that formalizes the statement ϕ: "for any π and x, π is not a valid PV proof of ϕ with time complexity β^d, where β is the time complexity of the statement ϕ."

Let $\pi_d := p_d$ and $e_d : \mathsf{UB}_{\mu,d,2}(\pi, q_d, x) = 1$, i.e., $q_d = (\llbracket \pi_d \rrbracket, \llbracket e_d \rrbracket)$. We now prove that the equation e_d satisfies the properties in Theorem 7.1.

Proposition 7.4 (time complexity of e_d)**.** *For $d \geq 1$, the time complexity of e_d in the context of π_d is a fixed polynomial in $|\pi| + |x|$, i.e., the exponent of the polynomial is independent of the d.*

Proof Sketch. Note that the only occurrence of the constant d in the definition of e_d is in $\mathsf{UB}_{\mu,d,2}(\pi, \tau, x)$. Assuming straightforward formalization of the PV function $\mathsf{UB}_{\mu,d,2}$, the time complexity of e_d should be a fixed polynomial in the running time of the terms in e_d, which is in turn a fixed polynomial in the input length $|\pi| + |x|$. □

7.2.1 Proof of the Lower Bound

Lemma 7.5 (lower bound for e_d)**.** *For $d \geq 1$, there is no* PV *proof π concluding e_d such that the time complexity of $\langle \pi_d, \pi \rangle$ is at most $o(\beta^d)$, where $\beta := \beta^{\mathsf{TC}}_{e_d, \pi_d}(\pi, x)$.*

Proof. Fix any $d \geq 1$. Suppose, towards a contradiction, that there is a PV proof π_1^* of the equation e_d with time complexity $\mu = o(\beta^d)$. Since the time complexity of the proof $\langle \pi_d, \pi^* \rangle$ is at most $o(\beta^d)$, there is a number $n_0 \in \mathbb{N}$ such that for every $n \geq n_0$,

$$\mu^{\mathsf{TC}}_{\langle \pi_d, \pi^* \rangle}(n) \leq \left(\beta^{\mathsf{TC}}_{e_d, \pi_d}(n)\right)^d. \tag{7.2}$$

Recall that the algorithms $A_\mu(\langle \pi_d, \pi^* \rangle, |x|, |x|)$ in Theorem 5.10 and $A_\beta(\pi_d, e, |x|)$ in Theorem 5.11 run in time at most

$$\exp(\exp(|\pi_d| + |\pi^*| + |e_d|)) \cdot \mathsf{polylog}(|x|) = \mathsf{polylog}(|x|) \ll |x|,$$

where the first equality follows as $|\pi_d| + |\pi^*| + |e_d| = O(1)$. Therefore, there is a number $n_1 \in \mathbb{N}$ such that for every $n \geq n_1$, both $A_\mu(\langle \pi_d, \pi^* \rangle, n, n)$ and $A_\beta(\pi_d, e, n, n)$ halts in at most n steps during the simulation by $f_{\mu,2}$ and $f_{\beta,2}$ in $\mathsf{UB}_{\mu,d,2}$.

Fix $n := \max(n_0, n_1) + 1$. We can see that

$$\mathsf{UB}_{\mu,d,2}([\![\pi^*]\!], q_d, 1^n) = 1 \tag{7.3}$$

is a true equation in the standard model. In more detail:

- As $n \geq n_1$, the simulation of $A_\mu(\langle \pi_d, \pi^* \rangle, n, n)$ by $f_{\mu,2}(\langle \pi_e, \pi \rangle, 1^n)$ halts in n steps. By Theorem 5.10, it outputs $\mu^{\mathsf{TC}}_{\langle \pi_d, \pi^* \rangle}(n)$ (in the standard model).
- Similarly, $f_{\beta,2}(\pi_e, e, 1^n)$ outputs $\beta^{\mathsf{TC}}_{e_d, \pi_d}(n)$ (in the standard model).
- Since $n \geq n_0$, we know by Equation (7.2) that

  $$f_{\mu,2}(\langle \pi_e, \pi \rangle, 1^n) \leq \left(f_{\beta,2}(\pi_e, e, 1^n)\right)^d,$$

  which implies that $\mathsf{UB}_{\mu,d,2}([\![\pi^*]\!], q_d, 1^n) = 1$ in the standard model by the definition of $\mathsf{UB}_{\mu,d,2}$.

Recall that the equation e_d is provable in PV. By the soundness of PV, we know that e_d is true in the standard model, and thus

$$\mathsf{UB}_{\mu,d,2}([\![\pi^*]\!], q_d, 1^n) = 0$$

is true in the standard model. This leads to a contradiction with Equation (7.3). □

7.2.2 Proof of the Upper Bound

Lemma 7.6. *There is a* PV *function $R(\pi, x)$ and a constant $c \geq 1$ such that the following holds. For every $d \geq 1$, the conditional equation*

$$e_{e_d \leq \mathsf{Con}} : \mathsf{Con}_{c \cdot d}(R(\pi, x), z) \Rightarrow e_d(\pi, x),$$

where $z := 1^{|x|+|\pi|}$, admits a PV *proof of time complexity at most $O_d((|\pi| + |x|)^c)$.*

Intuitively, this lemma shows that it is feasibly provable in PV that the bounded feasibility consistency of PV implies the sentence e_d. By combining Lemma 7.3, we can conclude the upper bound for e_d:

Lemma 7.7 (upper bound for e_d)**.** *There is a constant $c \geq 1$ such that for every $d \geq 1$, e_d admits a* PV *proof of time complexity at most $O_d(\beta^{cd})$, where $\beta := \beta^{\mathsf{TC}}_{e_d, \pi_d}(\pi, x)$.*

Proof. Let $c' \geq 1$ be the constant in Lemma 7.3. By Lemma 7.3, PV proves that $\mathsf{Con}_c(\pi, z)$, and by applying the logical rule (L4), PV also proves $\mathsf{Con}_c(R(\pi, x), z)$. The proof of e_d then follows by Lemma 7.6 and applying Modus Ponens.

By the feasible deduction theorem (see Theorem 6.2), the time complexity of the proof is the summation of the time complexity of the proof of $\mathsf{Con}_{c' \cdot d}(R(\pi, x), z)$ and the time complexity of the proof of e_{e_d} in Lemma 7.6. The latter term is a fixed polynomial in $(|\pi| + |x|)^{c'd}$. The first term involves two parts:

- The time complexity contribution by Lemma 7.3, which is a fixed polynomial in $(|R(\pi,x)|+|z|)^d$. As $R(\pi,x)$ is a PV function, its output length is a fixed polynomial in $|\pi|+|x|$. Therefore, the time complexity contribution is a fixed polynomial in $(|x|+|\pi|)^{c'd}$.
- The time complexity contribution by the logical rule (L4), which is a fixed polynomial in the running time of terms in $\mathsf{Con}_d(R(\pi,x),z)$, and is therefore a fixed polynomial in $(|x|+|\pi|)^{c'd}$.

In summary, the time complexity of the proof is a fixed polynomial in $(|\pi|+|x|)^{c'd}$. Recall that by Proposition 7.4, $\beta := \beta^{\mathsf{TC}}_{e_d,\pi_d}(\pi,x)$ is a fixed polynomial in $|\pi|+|x|$. We can choose $c = O_{c'}(1)$ to be sufficiently large such that the time complexity of the proof is at most $O_d(\beta^{cd})$. □

Proof of Lemma 7.6. Let $c \geq 1$ be a constant to be determined later, and $R(\pi,x)$ be a PV function to be specified. Recall that $e_d(\pi,x)$ formalizes the statement "π is not a valid feasible PV proof of e_d". The PV proof of the equation $e_{e_d \leq \mathsf{Con}}$ is as follows.

- (*Counter-examples*). Towards a contradiction, we assume that $\mathsf{Con}_{c\cdot d}(R(\pi,x),z) = 1$ and $e_d(\pi,x)$ is not a true equation, which means that there are π^* and x^* such that $e_d(\pi^*,x^*)$ is false, i.e.,

  $$\mathsf{UB}_{\mu,d,2}(\llbracket \pi^* \rrbracket, q_d, x^*) = 1,$$

  and $\mathsf{Con}_{c\cdot d}(R(\pi^*,x^*),z^*) = 1$ is true for $z^* := 1^{|x^*|+|\pi^*|}$.

- (*Formalizing Proofs*). Note that $\mathsf{UB}_{\mu,d,2}(\llbracket \pi^* \rrbracket, q_d, x^*) = 1$ is a true PV equation with *no variable*. Therefore, by unwinding the computation of $\mathsf{UB}_{\mu,d,2}(\llbracket \pi^* \rrbracket, q_d, x^*)$, we can obtain a PV proof of the equation

  $$e_1^* : \mathsf{UB}_{\mu,d,2}(\llbracket \pi^* \rrbracket, q_d, x^*) = 1.$$

  Denote this proof by π_1^*.

- (*Proof of Contradiction*). As we know that $\mathsf{UB}_{\mu,d,2}(\llbracket \pi^* \rrbracket, q_d, x^*) = 1$, by its definition, π^* is a proof of the equation $\mathsf{UB}_{\mu,d,2}(\pi, q_d, x) = 0$ with time complexity at most $\beta^{\mathsf{TC}}_{e_d,\pi_d}(|\pi|,|x|)^d$. By applying the logical rule (L4) after π^* to substitute π with π^* and x with x^*, we can obtain a proof π_0^* that

  $$e_0^* : \mathsf{UB}_{\mu,d,2}(\llbracket \pi^* \rrbracket, q_d, x^*) = 0.$$

  Note that from e_0^* and e_1^*, we can derive $0 = 1$ and thus $s_0(z^*) = s_1(z^*)$; denote these new proof lines by π_p^*. By combining π_0^*, π_1^*, and π_p^*, we can obtain a PV proof π_{in}^* of the equation $s_0(z^*) = s_1(z^*)$.

- (*Formalizing Time Complexity Analysis*). In addition, we analyze the time complexity of the proof $\pi_{in}^* = \langle \pi_0^*, \pi_1^*, \pi_p^* \rangle$ concluding $s_0(z^*) = s_1(z^*)$.

  - π_0^* has time complexity at most $\beta^{\mathsf{TC}}_{e_d,\pi_d}(|\pi^*|,|x^*|)^d$, as mentioned above. Recall that $\beta^{\mathsf{TC}}_{e_d,\pi_d}(|\pi^*|,|x^*|)$ is a fixed polynomial in $|\pi^*|+|x^*|$, and the proof (see Proposition 7.4) can be formalized in PV.
  - π_1^* is simply unwinding the computation of the function $\mathsf{UB}_{\mu,d,2}(\llbracket \pi^* \rrbracket, q_d, x^*)$, the time complexity of π_1^* is a fixed polynomial in the time complexity of the algorithm, and such analysis can be formalized within PV.
  - It can be verified that the time complexity of π_p^* is a fixed polynomial in $|z^*|$.

  Therefore, the time complexity of the proof π_{in}^* is a fixed polynomial in $|z^*|^d$, and it can be verified such analysis can be formalized in PV.

- (*Bounded Feasibility Consistency*). Recall that by the assumption in the first step, $\mathsf{Con}_d(R(\pi^*,x^*),z^*) = 1$. Note that the constant $c \geq 1$ and the function $R(\pi,x)$ are yet to be specified.

  - We choose R such that given (π^*,x^*), it will output π_{in}^*. The computational complexity of the function will be a fixed polynomial in its input.

– As mentioned above, π^*_{in} is a valid PV proof that concludes $s_0(z^*) = s_1(z^*)$, and the time complexity of π^*_{in} is a fixed polynomial in $|z^*|^d$. By choosing c to be sufficiently large, we can prove (within PV) that $\mathsf{UB}_{\mu,c\cdot d}(\pi^*_{in}, z^*) = 1$ and $\mathsf{ValidPV}(\pi^*_{in}, [\![s_0(z^*) = s_1(z^*)]\!]) = 1$.

This leads to a contradiction with $\mathsf{Con}_{c\cdot d}(R(\pi^*, x^*), z^*) = 1$.

Moreover, we need to prove that the entire PV proof above has time complexity at most $O_d((|\pi| + |x|)^c)$ for a fixed constant c that is independent of d.

As the full formalization in PV is tedious, we only focus on three main technical steps: The definition of the function $R(\pi, x)$, and formalization of the time complexity analysis of π^*_{in} within PV (i.e. the 4-th bullet above), and the time complexity analysis of the entire proof (in meta-theory).

Definition of $R(\pi, x)$. Recall that we need to define $R(\pi, x)$ as a polynomial-time function such that $R(\pi^*, x^*)$ outputs $\pi^*_{in} = \langle \pi^*_0, \pi^*_1, \pi^*_p \rangle$. The function works as follows:

(1) (*Proof Check*). Given (π^*, x^*), the algorithm first verifies whether π^* is a valid PV proof of e_d (i.e. the function $\mathsf{ValidPV}(\pi^*, e_d)$ outputs 1). If not, the algorithm immediately halts and outputs ε. The time complexity of this step is a fixed polynomial in $|\pi^*|$.

(2) (*Counter-example Check*). The algorithm then runs $\mathsf{UB}_{\mu,d,2}([\![\pi^*]\!], q_d, x^*)$; it immediately halts and outputs ε if $\mathsf{UB}_{\mu,d,2}([\![\pi^*]\!], q_d, x^*) = 0$. The time complexity of this step is a fixed polynomial in $|\pi^*| + |x^*|$.

(3) (*Producing π^*_1*). As we have verified that $\mathsf{UB}_{\mu,d,2}([\![\pi^*]\!], q_d, x^*) = 1$, the algorithm produces the PV proof π^*_1 of the equation $\mathsf{UB}_{\mu,d,2}([\![\pi^*]\!], q_d, x^*) = 1$ with no variable. Note that the size of the proof π^*_1 is a fixed polynomial in the time complexity of $\mathsf{UB}_{\mu,d,2}([\![\pi^*]\!], q_d, x^*)$, which is in turn a fixed polynomial in $|\pi^*| + |x^*|$.

(4) (*Producing π^*_0*). Recall that π^* is a valid proof of e_d. We append two new proof lines to π^* with the logical rule (L4) to obtain a proof π^*_0 of $e_d[\pi/\pi^*, x/x^*]$. The time complexity of this step is a fixed polynomial in $|\pi^*| + |x^*|$.

(5) (*Producing π^*_p*). Note that π^*_0 is a valid proof concluding $\mathsf{UB}_{\mu,d,2}([\![\pi^*]\!], q_d, x^*) = 0$, and π^*_1 is a valid proof concluding $\mathsf{UB}_{\mu,d,2}([\![\pi^*]\!], q_d, x^*) = 1$. The algorithm produces proof lines (using logical rules) to conclude that $0 = 1$. Moreover, we can prove $s_0(x) = s_1(x)$ in PV by induction on x, and the algorithm produces these proof lines. Finally, the algorithm produces a proof lines concluding $s_0(x^*) = s_1(x^*)$ by applying the logical rule (L4) to substitute x with x^*. The time complexity of this step is a fixed polynomial in $|\pi^*| + |x^*|$.

The time complexity of the function $R(\pi, x)$ is thus a fixed polynomial in its input length (i.e. the exponent is independent of d), and the function can be formalized in PV straightforwardly.

Formalization of the time complexity analysis of π^*_{in}. Next we show that PV proves the conditional equation

$$e_{\mu,c} : \mathsf{UB}_{\mu,d,2}(\pi^*, q_d, x^*) \Rightarrow \mathsf{UB}_{\mu,c\cdot d}(\pi^*_{in}, z^*) = 1,$$

where $z^* := 1^{|x^*|+|\pi*|}$ and $\pi^*_{in} := R(\pi^*, x^*)$, and c is a sufficiently large constant to be determined later. That is, for any π^* and x^* such that the equation $e_d[\pi/\pi^*, x/x^*]$ is false, $\pi^*_{in} := R(\pi^*, x^*)$ is a valid PV proof with time complexity at most $c \cdot |z^*|^c$.

Recall that $\mathsf{UB}_{\mu,c}(\pi^*_{in}, z^*)$ outputs 1 if $f_\mu(\pi^*_{in}, z^*) \le c \cdot |z^*|^c$ and outputs 0 otherwise, where $f_\mu(\pi^*, z^*)$ is the PV function that simulates $A_\mu(\pi^*_{in}, |z^*|)$ in Theorem 5.10 for $|z^*|$ steps, outputs $A_\mu(\pi^*_{in}, |z^*|)$ if it halts, and outputs ε otherwise. By the definition of the algorithm $R(\cdot, \cdot)$, we can see that $\pi^*_{in} = \langle \pi^*_0, \pi^*_1, \pi^*_p \rangle$ is a PV proof concluding $s_0(z^*) = s_1(z^*)$ — an equation without any variable. We now analyze the time complexity contribution of each part of π_{in}.

- (π_p^*). This part of the proof is explicit: it consists of applications of the logical rules and one application of the induction rule. It can be verified that the time complexity of this part is a fixed polynomial in $|z^*|$. Moreover, assuming that f_μ is a straightforward formalization of A_μ, it can be proved in PV that the time complexity contribution of π_p^* in $f_\mu(\pi_{in}^*, z^*)$ is a fixed polynomial in $|z^*|$.

- (π_1^*). This part is to prove the equation $\mathsf{UB}_{\mu,d,2}(\pi^*, q_d, x^*) = 1$ by unwinding the computation of the function $\mathsf{UB}_{\mu,d,2}(\pi^*, q_d, x^*)$. The number of lines in the PV proof π_1^* is a fixed polynomial in $|\pi^*| + |x^*|$ (i.e. the exponent is independent of d) as the running time of the function is a fixed polynomial in its input length. Moreover, the proof only uses definition axioms of initial functions and introduced functions and the logical rules. Therefore the acquisition map of the proof is simple and can be computed in fixed polynomial time. Assuming that f_μ is a straightforward formalization, it is provable in PV that the time complexity contribution of π_1^* is a fixed polynomial in $|\pi^*| + |x^*|$, which is at most a fixed polynomial in $|z^*|$.

- (π_0^*). Note that π_0^* consists of π^* and two new proof lines using the logical rule (L4). The time complexity contribution of π^* is at most $(f_{\beta,2}(\pi_d, e_d, x))^d$ by the premise of the conditional equation $e_{\mu,c}$, which is a fixed polynomial in $|z^*|^d = (|\pi^*| + |x^*|)^d$. The time complexity of two new proof lines is negligible.

Therefore, assuming the premise of $e_{\mu,c}$, the time complexity contribution of each part is at most a fixed polynomial in $|z^*|^d$. Provided that f_μ is a straightforward simulation of A_μ, we can prove in PV that the total time complexity of π_{in}^* is a fixed polynomial in $|z^*|^d$. We can then choose $c \geq 1$ to be a sufficiently large constant such that $e_{\mu,c}$ is provable in PV.

Time complexity analysis in meta-theory. As the proof involves five steps and each step is complicated, we will only sketch the analysis of its time complexity.

- The time complexity contribution of the first and last step is a fixed polynomial in $|\pi| + |x|$. This is because the proof closely follows the computation of the terms, and all terms involved have fixed polynomial running time.

- The second and third steps are essentially the correctness of the function $R(\pi, x)$. Note that the correctness proofs of π_0^* and π_1^* closely follow from the computation of $R(\pi, x)$, and therefore the time complexity contribution of the proofs is at most a fixed polynomial in the running time of $R(\pi, x)$, which is in turn a fixed polynomial in $|\pi| + |x|$. The correctness proof of π_p^* involves an induction on the lengths of z^*, and it can be verified that its time complexity contribution is at most a fixed polynomial in $|z^*|$, which is, in turn, a fixed polynomial in $|\pi| + |x|$.

- The fourth step is sketched above. In each of π_p^*, π_1^*, and π_0^*, the time complexity of the proof tightly follows the computation of the functions involved, and it can be verified that the total running time of the functions involved is a fixed polynomial in $|\pi| + |x|$.

Therefore, the total time complexity of the proof is a fixed polynomial in $|\pi| + |x|$. The lemma follows by choosing the constant c to be sufficiently large. □

Remark 7.2. It might be counterintuitive that the PV proof of $e_{e_d \leq \mathsf{Con}}$ is of time complexity at most a fixed polynomial in its input length, as the statement formalizes properties of PV proofs with time complexity $(|x| + |\pi|)^d$.

We note that the time complexity of the proof of an equation e is not necessarily higher than the time complexity of the proofs formalized in e. For instance, the time complexity of a proof π concluding $e(x)$ is at least $|x|$, while computing the time complexity of the proof π only takes $\exp(\exp(O(|\pi|))) \cdot \mathsf{polylog}(n)$ time (see Theorem 5.10), which could be much smaller than $|x|$ if $|\pi|$ is small. Similarly, the time complexity of the equation $\mathsf{Con}_d(\pi, x)$ is a fixed polynomial in its input length despite that it formalizes consistency of proofs with feasibility up to $|x|^d$ (see Proposition 7.2).

In Lemma 7.6, the proof of $e_{e_d \le \mathsf{Con}}$ closely follows from the algorithms that generate π^*_{in} and compute its time complexity. The time complexity of the proof is a fixed polynomial time as both algorithms run in fixed polynomial time.

7.2.3 Putting Things Together

Theorem 7.1 (feasibility hierarchy theorem)**.** *There are constants $c_1, c_2 \geq 1$ such that the following holds. For any constant $d \geq 1$, there is a* PV *equation $e_d : s_d(\pi, x) = t_d(\pi, x)$ and its context π_d such that:*

1. *The time complexity of the statement e_d is at most $\beta := \beta^{\mathsf{TC}}_{e_d,\pi_d}(|\pi|, |x|) = O_d((|\pi| + |x|)^{c_1})$.*
2. *e_d admits a* PV *proof of time complexity $O_d(\beta^{c_2 \cdot d})$;*
3. *e_d is unprovable in* PV *with time complexity $o(\beta^d)$.*

Proof. The theorem follows by combining Proposition 7.4 and lemmas 7.5 and 7.7. □

Similar to the common practice of computational complexity theory, we may define *proof complexity classes* as sets of equations that can be proved with bounded time complexity. Let $s(\beta)$ be a function. We define the proof complexity class $\mathsf{EqTIME}[s(\beta)]$ as follows:

Definition 7.3. $\mathsf{EqTIME}[s(\beta)]$ is the set of pairs $(e(\vec{x}), \pi_e)$ such that the following holds.

- $e(\vec{x})$ is an equation in the context π_e.
- There is a PV proof π such that $\pi^* := \langle \pi_e, \pi \rangle$ is a valid proof of the equation e.
- Let $n_1, \ldots, n_k$ be the input length of variables $\vec{x} = (x_1, \ldots, x_k)$ in $e(\vec{x})$. There are constants $c \geq 1$, $n_0 \geq 1$ such that if $n_1, n_2, \ldots, n_k \geq n_0$,

$$\mu^{\mathsf{TC}}_{\pi^*}(n_1, \ldots, n_k) \leq c \cdot s\left(\beta^{\mathsf{TC}}_{e,\pi_e}(n_1, \ldots, n_k)\right).$$

We may drop the context π_e and say $e(\vec{x}) \in \mathsf{EqTIME}[\beta^d]$ if there is no ambiguity.

We can then derive the following proof complexity class separation from Theorem 7.1:

Corollary 7.8. *There exists a constant $c \geq 1$ such that for every $d \geq 1$, $\mathsf{EqTIME}[\beta^d] \subsetneq \mathsf{EqTIME}[\beta^{cd}]$.*

7.3 Bounded Feasibility Consistency: Another Explicit Witness

As a corollary of Lemmas 7.3, 7.5 and 7.6, we can also prove that the bounded feasibility consistency statement $\mathsf{Con}_c(\pi, x)$ is also an explicit witness of the feasibility hierarchy theorem when d is sufficiently large. This can be viewed as a counterpart of Gödel's second incompleteness theorem. Formally:

Corollary 7.9. *There are constants $c_0, c_1, c_2, d_0 \geq 1$ such that for any constant $d \geq d_0$:*

1. *The time complexity of the statement $\mathsf{Con}_{c_0 \cdot d}(\pi, x)$ is at most $\beta := \beta^{\mathsf{TC}}_{\mathsf{Con}_{c_0 \cdot d}}(|\pi|, |x|) = O_d((|\pi| + |x|)^{c_1})$.*
2. *$\mathsf{Con}_{c_0 \cdot d}(\pi, x)$ admits a* PV *proof of time complexity $O_d(\beta^{c_2 \cdot d})$.*
3. *$\mathsf{Con}_{c_0 \cdot d}(\pi, x)$ is unprovable in* PV *with time complexity $O_d(\beta^d)$.*

In particular,

$$\mathsf{Con}_{c_0 \cdot d} \in \mathsf{EqTIME}[\beta^{c_2 \cdot d}] \setminus \mathsf{EqTIME}[\beta^d].$$

Proof. Let $c_0, c_1, c_2, d_0 \geq 1$ to be determined later. Fix any $d \geq d_0$. The first bullet follows from Proposition 7.2 when $c_1 = O(1)$ is sufficiently large. The second bullet follows from Lemma 7.3 by setting $c_2 = O_{c_0}(1)$ to be sufficiently large.

Now we prove the third bullet. Suppose, towards a contradiction, that $\mathsf{Con}_{c_0 \cdot d}(\pi, x)$ is provable in PV with time complexity $O(\beta^d)$. Recall that by Lemma 7.6, there is a constant $c \geq 1$ (independent of d) such that

$$e_{e_{c_1 \cdot d} \le \mathsf{Con}} : \mathsf{Con}_{c \cdot c_1 \cdot d}(R(\pi, x), z) \Rightarrow e_{c_1 \cdot d}(\pi, x), \tag{7.4}$$

where $z := 1^{|x|+|\pi|}$, admits a PV proof of time complexity at most $O_d((|\pi| + |x|)^c)$. Moreover, the output length of $R(\pi, z)$ must be $O_d((|\pi| + |x|)^c)$, and thus we can prove in PV the equation

$$\mathsf{Con}_{c_0 \cdot d}(R(\pi, x), z := 1^{|x|+|\pi|}) \tag{7.5}$$

with time complexity at most $O((|\pi| + |x|)^{cd})$. We choose $c_0 = c \cdot c_1 \geq 1$. By the feasibility deduction theorem (see Theorem 6.2), we can prove $e_{c_1 \cdot d}$ from Equations (7.4) and (7.5) with time complexity at most

$$O_d((|\pi| + |x|)^c) + O((|\pi| + |x|)^{cd}) = o(\beta^{c_1 \cdot d}),$$

by setting $c_1 = O_c(1)$ to be sufficiently large. This leads to a contradiction with Theorem 7.1. □

8 Uniformity vs Nonuniformity in Proof Complexity

The feasible hierarchy theorem states that more theorems can be proved if we allow arithmetic proofs to be less efficient in time complexity. Given the tight quantitative connection between the time complexity of arithmetic proofs and the size of propositional proofs, it is natural to consider whether propositional proofs (i.e. non-uniform proofs) could gain surprising power compared to arithmetic proofs (i.e. uniform proofs) if they have similar levels of feasibility.

8.1 Conjectures of Uniformity vs Nonuniformity Lower Bound

To compare the strength of arithmetic theories and propositional proof systems, we consider the provability of a sequence of uniformly generated propositional formulas. Formally:

Definition 8.1. A family $\varphi = \{\varphi_n\}_{n \in \mathbb{N}}$ of propositional formulas with n variables is said to be *uniform*, if there is a PV function Φ (called its *generator*) such that for any $n \in \mathbb{N}$, $\Phi(1^n)$ outputs the encoding of φ_n.

For simplicity, we always assume that the n-variable propositional formula φ_n is described by a binary string of length at least n, and we use $|\varphi_n|$ to denote the length of the description of φ_n.

A concrete example of uniform families of propositional formulas is the formulas $[e]^n_{\mathsf{Cook}}$ obtained from the propositional translation of a PV equation $e(x)$ with one variable. To be more specific, let $e_d(\pi, x)$ be the equation in Theorem 7.1, and consider the univariant version $e'_d(\tau) := e_d(\pi/\mathsf{Left}(\tau), x/\mathsf{Right}(\tau))$, where $\mathsf{Left}(\tau)$ and $\mathsf{Right}(\tau)$ outputs the left and right component of a pair. The propositional translation of $e'_d(\tau)$ is a uniform family of formulas $\varphi_n := [e'_d(\tau)]^n_{\mathsf{Cook}}$, where φ_n is a tautology if and only if $\forall \tau \in \{0,1\}^n$ $e'_d(\tau)$ is true.

Definition 8.2 (EF proofs)**.** Let $s(m)$ be a function. A family $\varphi = \{\varphi_n\}_{n \in \mathbb{N}}$ of propositional formulas is said to admit EF proofs of size s if there is an $n_0 \geq 1$ such that for every $n \geq n_1$, there is an EF proof of φ_n of size $s(|\varphi_n|)$. The set of all families of propositional formulas that admits proofs of size $O(s(m))$ is denoted by $\mathsf{EF\text{-}SIZE}[s(m)]$.

Definition 8.3 (PV proofs)**.** A uniform family $\varphi = \{\varphi_n\}_{n \in \mathbb{N}}$ of propositional formulas with generator Φ is said to be provable in PV if PV_1 proves

$$\forall n \in \mathsf{Log}\ \forall x \in \{0,1\}^n\ \mathsf{Val}(\Phi, x) = 1, \tag{8.1}$$

where $\mathsf{Val}(\Phi, x)$ is the PV function formalizing that $\varphi_{|x|}(x) = 1$, $\varphi_{|x|} := \Phi(1^{|x|})$. That is, PV proves that φ_n is a tautology for any n. The set of all families of uniform propositional formulas that admit PV proofs is denoted by FPV.

Conjecture 8.1. $\mathsf{FPV} \not\subseteq \mathsf{EF\text{-}SIZE}[m^k]$ for any $k \geq 1$.

Conjecture 8.1 can be viewed as the proof complexity counterpart of the circuit lower bound conjecture $\mathsf{P} \not\subseteq \mathsf{SIZE}[n^k]$ for any k. We also note that the conjecture is *stronger* than Cook's conjecture that EF is not p-bounded (see, e.g., [Coo75]): the latter conjecture only requires the existence of a family $\varphi = \{\varphi_n\}_{n \in \mathbb{N}}$ of *tautologies* such that for infinitely many n, φ_n does not have $|\varphi_n|^k$ size EF proof, while the former conjecture further requires that φ is a PV-provable family of tautologies.

8.2 Lower Bounds against Uniform Propositional Proofs

Next, we introduce a notion of "uniform propositional proofs" as a proof complexity counterpart of uniform circuits, e.g., P-uniform $\mathsf{SIZE}[n^k]$. Formally:

Definition 8.4 (PV-uniform EF proofs)**.** Let $s(m)$ be a PV function. A uniform family $\varphi = \{\varphi_n\}_{n\in\mathbb{N}}$ of propositional formulas with generator Φ is said to admit PV-uniform EF proofs of size s such that PV_1 proves

$$\forall n \in \mathsf{Log}, n \geq n_0\ \exists \pi\ (\mathsf{EFSize}(\pi) \leq c \cdot s(|\Phi(1^n)|) \wedge \mathsf{IsEFProof}(\Phi(1^n), \pi) = 1) \tag{8.2}$$

for some constants $c, n_0 \geq 1$. The set of all uniform families of propositional formulas that admits PV-uniform proofs of size $s(m)$ is denoted by PV-uniform EF-SIZE$[s(m)]$.

Equivalently, by the witnessing theorem of PV_1, we can define that Φ admit PV-uniform EF proofs of size s if there is a PV function GenProof such that PV proves

$$\forall n \in \mathsf{Log}, n \geq n_0\ (\mathsf{EFSize}(\mathsf{GenProof}(1^n)) \leq s(|\Phi(1^n)|) \wedge \mathsf{IsEFProof}(\Phi(1^n), \mathsf{GenProof}(1^n)) = 1) \tag{8.3}$$

for some constant $n_0 \geq 1$.

One reason to introduce the notion of uniform propositional proofs is that the following theorem, which is implicit in [Coo75], provides an exact characterization of FPV by formulas provable with uniform propositional proofs.

Theorem 8.1 (implicit in [Coo75])**.** FPV $=$ PV-uniform EF-SIZE$[m^{O(1)}]$.

Proof Sketch. (PV-uniform EF-SIZE$[m^{O(1)}] \subseteq$ FPV). Recall that the soundness of EF is provable in PV (see Theorem 3.2). If PV proves that φ_n admits an EF proof, PV also proves with the soundness of EF that φ_n is a tautology.

(FPV $\subseteq$ PV-uniform EF-SIZE$[m^{O(1)}]$). Let $\varphi = \{\varphi_n\}_{n\in\mathbb{N}}$ be a uniform family of formulas that are PV-provably tautologies. Let P be the propositional proof system that accepts only φ_n for $n \geq 1$, then PV proves that P is a sound propositional proof system. By the main theorem of [Coo75] (see also Theorem 3.3), PV proves that EF p-simulate the propositional proof system P. In particular, this means that φ_n (which admits a linear-size proof in P) also admits a polynomial-size proof in EF. This implies that $\varphi \in$ PV-uniform EF-SIZE$[m^{O(1)}]$. □

We are now ready to introduce the following conjecture that FPV, or equivalently, PV-uniform polynomial size EF proofs, is strictly stronger than PV-uniform EF proofs of fixed polynomial size.

Conjecture 8.2. FPV $\not\subseteq$ PV-uniform EF-SIZE$[m^k]$ for any $k \geq 1$.

This conjecture is not directly related to the main results of the paper. Nevertheless, we argue that the feasibility hierarchy theorem (see Theorem 7.1) provides strong evidence for Conjecture 8.2. Suppose the conjecture is false for some constant $k \geq 1$. Fix any constant $d \gg k$, and let $e(x)$ be a PV-provable equation such that $e(x) \notin \mathsf{EqTIME}[\beta^d]$. Let $\varphi_n = [e]^n_{\mathsf{Cook}}$ be the propositional translation of φ_n. We know that φ_n is *unprovable* in PV with time complexity $O(\beta^d)$, but it is provable in PV that φ_n admits EF proofs of size $O(\beta^k) \ll \beta^d$. This indicates that for any statement with high proof complexity in terms of time complexity, not only does it have short EF proofs, but the proofs are also uniformly constructible and provably correct.

Following the intuition, we further conjecture that the propositional translation of the univariant version $e_d(\tau) := e_d(\mathsf{Left}(\tau), \mathsf{Right}(\tau))$ of the statement in Theorem 7.1 or the bounded feasibility hierarchy statements (see Corollary 7.9) is an explicit witness of the separation in Conjecture 8.2.

8.3 Conditional Unprovability of Proof Complexity Upper Bounds

Conjecture 8.2 is an interesting conjecture in its own right. In this section, we further show that it implies strong unprovability results in PV, namely PV cannot prove NP = coNP.

First, we define what we mean by proving NP = coNP in PV_1. Recall that UB_M^{k,m_0} is a PV function formalizing that M is a sound, complete, and p-bounded propositional proof system. More formally:

Definition 3.1. Let M be a PV function and $k, m_0 \geq 1$ be integers. We define UB_M^{k,n_0} be conjunction of two sentences Sound_M^{k,m_0} and $\mathsf{Complete}_M^{k,m_0}$, where

$$\begin{aligned}\mathsf{Sound}_M^{k,m_0} := \forall n, m \in \mathsf{Log},\ m \geq m_0, \forall \text{ formula } \varphi \in \{0,1\}^m \text{ with } n \text{ variables such that:}\\ \forall \text{ assignment } x \in \{0,1\}^n : \varphi(x) = 1 \text{ or } \forall \text{ witness } \pi \text{ of size at most } m^k : M(\varphi, \pi) = 0;\\ \mathsf{Complete}_M^{k,m_0} := \forall n, m \in \mathsf{Log},\ m \geq m_0, \forall \text{ formula } \varphi \in \{0,1\}^m \text{ with } n \text{ variables such that:}\\ \exists \text{ assignment } x \in \{0,1\}^n : \varphi(x) = 0 \text{ or } \exists \text{ witness } \pi \text{ of size at most } m^k : M(\varphi, \pi) = 1.\end{aligned}$$

Definition 8.5. We say $\mathsf{PV}_1 \vdash$ "NP = coNP" if there is a PV function M and integers $k, m_0 \geq 1$ such that $\mathsf{PV}_1 \vdash \mathsf{UB}_M^{k,m_0}$; and we say $\mathsf{PV}_1 \nvdash$ "NP = coNP" if $\mathsf{PV}_1 \vdash$ "NP = coNP" is false.

We note that the constants k, m_0 are quantified outside of the theory. This makes the unprovability results weaker, as the provability of NP = coNP when k, m_0 quantified within the theory is implied, but may not imply the provability when they are quantified outside of the theory.

Theorem 8.2. $\mathsf{PV}_1 \nvdash$ "NP = coNP" *under Conjecture 8.2.*

The proof of Theorem 8.2 consists of two steps. First, we show that $\mathsf{PV}_1 \vdash$ "NP = coNP", i.e., there is a polynomially bounded propositional proof system, if and only if PV proves that EF is a polynomially bounded proof system. We then show that the latter statement is false under Conjecture 8.2.

Let $V_{\mathsf{EF}}(\varphi, \pi)$ be a straightforward PV function that verifies whether π is an EF proof of φ: it outputs 1 (resp. 0) if π is (resp. is not) a valid EF proof of the formula φ.

Lemma 8.3. $\mathsf{PV}_1 \vdash$ "NP = coNP" *if and only if there are integers* $k, m_0 \geq 1$ *such that* $\mathsf{PV}_1 \vdash \mathsf{UB}_{V_{\mathsf{EF}}}^{k,m_0}$.

Proof. The (*if*) direction is trivial, so it suffices to prove the (*only if*) direction.

Suppose that $\mathsf{PV}_1 \vdash$ "NP = coNP", we know by the definition that there is a PV function M and integers $k, m_0 \geq 1$ such that $\mathsf{PV}_1 \vdash \mathsf{UB}_M^{k,m_0} = \mathsf{Sound}_M^{k,m_0} \wedge \mathsf{Complete}_M^{k,m_0}$, i.e., PV_1 proves that M is a sound and complete proof system for Taut with proof length m^k. Our goal is to show that PV_1 proves $\mathsf{UB}_{V_{\mathsf{EF}}}^{k_{\mathsf{EF}},m_{\mathsf{EF}}}$ for some $k_{\mathsf{EF}}, m_{\mathsf{EF}} \geq 1$, i.e., EF is a sound and p-bounded proof system for Taut; note that the soundness of EF is known to be provable in PV_1 (see Theorem 3.2), it suffices to prove that EF is p-bounded.

Let $k_{\mathsf{EF}}, m_{\mathsf{EF}}$ be parameters to be determined later. We argue in PV_1 that $\mathsf{Complete}_{V_{\mathsf{EF}}}^{k_{\mathsf{EF}},m_{\mathsf{EF}}}$. Let $n, m \in \mathsf{Log}$, $m \geq m_{\mathsf{EF}}$, $\varphi \in \{0,1\}^m$ be the encoding of a variable with n variables, our goal is to either find a falsifying assignment of φ or find an EF proof of φ of size $m^{k_{\mathsf{EF}}}$. By $\mathsf{Complete}_M^{k_0,m_0}$, i.e. the completeness of M, we can either obtain a falsifying assignment of φ or find an M-proof of φ of size m^k. In the former case, we can directly conclude the proof using the falsifying assignment.

Now we consider the latter case. Let π_M be the M-proof of φ of size m^k. As $\mathsf{PV}_1 \vdash \mathsf{Sound}_M^{k,m_0}$, we know by Theorem 3.3 that EF p-simulates M. As π_M is an M-proof of φ, we further know that there are constants k', m_0' such that if $m \geq m_0'$, there is an EF proof π_{EF} of φ of size $|\pi_M|^k \leq m^{k \cdot k'}$. This concludes the proof if we set $m_{\mathsf{EF}} := \max\{m_0, m_0'\}$ and $k_{\mathsf{EF}} = k \cdot k'$. □

Lemma 8.4. *Under Conjecture 8.2,* $\mathsf{PV}_1 \nvdash \mathsf{Complete}_{V_{\mathsf{EF}}}^{k,m_0}$ *for any constants* $k, m_0 \geq 1$.

Proof. Fix arbitrary constants $k, m_0 \geq 1$. Suppose, towards a contradiction, that $\mathsf{PV}_1 \vdash \mathsf{Complete}^{k,m_0}_{V_{\mathsf{EF}}}$ and Conjecture 8.2 is true, i.e., $\mathsf{FPV} \not\subseteq \mathsf{PV\text{-}uniform\ EF\text{-}SIZE}[m^k]$ for some $k \geq 1$. Then there is a uniform family of formulas $\varphi = \{\varphi_n\}_{n\in\mathbb{N}}$ generated by a PV-function Φ such that $\varphi \in \mathsf{FPV} \setminus \mathsf{PV\text{-}uniform\ EF\text{-}SIZE}[m^k]$. Note that $\varphi \in \mathsf{FPV}$ implies that it is provable in PV_1 that for any $n \in \mathsf{Log}$, φ_n is a tautology. We will show that $\varphi \in \mathsf{PV\text{-}uniform\ EF\text{-}SIZE}[m^k]$, which leads to a contradiction.

We argue in PV. We will show that for any $n \geq m_0$, φ_n admits an EF proof of size at most $|\varphi_n|^k$. Fix any $n \in \mathsf{Log}$ such that $n \geq m_0$ and let $m := |\varphi_n| \geq m_0$. Since $\mathsf{PV} \vdash \mathsf{Complete}^{k,m_0}_{V_{\mathsf{EF}}}$, we know that either there is a falsifying assignment of φ_n, or there is an EF proof π_{EF} of φ_n of size at most m^k. The former case leads to a contradiction as φ_n is a tautology and is provable in PV_1. Subsequently, we can obtain an EF proof π_{EF} of φ_n of size m^k, which completes the proof. □

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ABM23] Albert Atserias, Sam Buss, and Moritz Müller. On the consistency of circuit lower bounds for non-deterministic time. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1257–1270, 2023.

[Ajt83] Miklós Ajtai. $\sum^1_1$-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of mathematics*, pages 781–793, 2004.

[AT24] Albert Atserias and Iddo Tzameret. Feasibly constructive proof of schwartz-zippel lemma and the complexity of finding hitting sets. *arXiv preprint arXiv:2411.07966*, 2024.

[BKO20] Jan Bydzovsky, Jan Krajíček, and Igor C. Oliveira. Consistency of circuit lower bounds with bounded theories. *Logical Methods in Computer Science*, 16(2), 2020.

[BM20] Jan Bydzovsky and Moritz Müller. Polynomial time ultrapowers and the consistency of circuit lower bounds. *Arch. Math. Log.*, 59(1-2):127–147, 2020.

[BN21] Sam Buss and Jakob Nordström. Proof complexity and SAT solving. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 233–350. IOS Press, 2021.

[BPI22] Douglas Bridges, Erik Palmgren, and Hajime Ishihara. Constructive Mathematics. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2022 edition, 2022.

[Bus86] Samuel R. Buss. *Bounded Arithmetic*. Bibliopolis, 1986.

[Bus87] Samuel R. Buss. Polynomial size proofs of the propositional pigeonhole principle. *J. Symb. Log.*, 52(4):916–927, 1987.

[Bus94] Samuel R Buss. On gödel's theorems on lengths of proofs i: Number of lines and speedup for arithmetics. *The Journal of Symbolic Logic*, 59(3):737–756, 1994.

[Bus95a] Samuel R Buss. On gödel's theorems on lengths of proofs ii: Lower bounds for recognizing k symbol provability. In *Feasible mathematics II*, pages 57–90. Springer, 1995.

[Bus95b] Samuel R. Buss. Relating the bounded arithmetic and polynomial time hierarchies. *Annals of Pure and Applied Logic*, 75(1-2):67–77, 1995.

[Bus99] Samuel R. Buss. Bounded arithmetic, proof complexity and two papers of parikh. *Ann. Pure Appl. Log.*, 96(1-3):43–55, 1999.

[Bus15] Sam Buss. Quasipolynomial size proofs of the propositional pigeonhole principle. *Theoretical Computer Science*, 576:77–84, 2015.

[CFK+15] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.

[CG25] Marco Carmosino and Stefan Grosser. Student-teacher constructive separations and (un) provability in bounded arithmetic: Witnessing the gap. In *Symposium on Theory of Computing* (STOC), *to appear*, 2025.

[CKK+25] Marco Carmosino, Valentine Kabanets, Antonina Kolokolova, Igor C. Oliveira, and Dimitrios Tsintsilidas. Provability of the circuit size hierarchy and its consequences. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, January 7-10, 2025, Columbia University, New York, NY, USA*, volume 325 of *LIPIcs*, pages 30:1–30:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[CKKO21] Marco Carmosino, Valentine Kabanets, Antonina Kolokolova, and Igor C. Oliveira. Learn-uniform circuit lower bounds and provability in bounded arithmetic. In *Symposium on Foundations of Computer Science* (FOCS), 2021.

[CLO24a] Lijie Chen, Jiatu Li, and Igor C Oliveira. On the unprovability of circuit size bounds in intuitionistic S^1_2. *arXiv preprint arXiv:2404.11841*, 2024.

[CLO24b] Lijie Chen, Jiatu Li, and Igor C Oliveira. Reverse mathematics of complexity lower bounds. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 505–527. IEEE, 2024.

[CLRS22] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[CN10] Stephen A. Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. Cambridge University Press, 2010.

[Cob65] Alan Cobham. The intrinsic computational difficulty of functions. *Proc. Logic, Methodology and Philosophy of Science*, pages 24–30, 1965.

[Coo75] Stephen A. Cook. Feasibly constructive proofs and the propositional calculus (preliminary version). In *Symposium on Theory of Computing* (STOC), pages 83–97, 1975.

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979.

[CT06] Stephen A. Cook and Neil Thapen. The strength of replacement in weak arithmetic. *ACM Trans. Comput. Log.*, 7(4):749–764, 2006.

[CU93] Stephen Cook and Alasdair Urquhart. Functional interpretations of feasibly constructive arithmetic. *Annals of Pure and Applied Logic*, 63(2):103–200, 1993.

[DM22] Damir D. Dzhafarov and Carl Mummert. *Reverse mathematics: problems, reductions, and proofs*. Springer Nature, 2022.

[dRGR22] Susanna F. de Rezende, Mika Göös, and Robert Robere. Guest column: Proofs, circuits, and communication. *SIGACT News*, 53(1):59–82, 2022.

[FGHK23] Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. Improving 3n circuit complexity lower bounds. *Comput. Complex.*, 32(2):13, 2023.

[GHJ+24] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Separations in proof complexity and TFNP. *J. ACM*, 71(4):26:1–26:45, 2024.

[Göd31] Kurt Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für mathematik und physik*, 38:173–198, 1931.

[Göd36] Kurt Gödel. Über die länge von beweisen. *Ergebnisse eines mathematischen Kolloquiums*, 7:23–24, 1936.

[Gol08] Oded Goldreich. Computational complexity: a conceptual perspective. *ACM Sigact News*, 39(3):35–39, 2008.

[Hak85] Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985.

[HS65] Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[HS66] Fred C Hennie and Richard Edwin Stearns. Two-tape simulation of multitape turing machines. *Journal of the ACM (JACM)*, 13(4):533–546, 1966.

[HS08] Grzegorz Herman and Michael Soltys. A polytime proof of correctness of the rabin-miller algorithm from fermat's little theorem. *arXiv preprint arXiv:0811.3959*, 2008.

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *Symposium on Theory of Computing* (STOC), pages 1076–1089. ACM, 2023.

[Jeř05] Emil Jeřábek. *Weak pigeonhole principle and randomized computation*. PhD thesis, Charles University in Prague, 2005.

[Jeř06] Emil Jeřábek. The strength of sharply bounded induction. *Mathematical Logic Quarterly*, 52(6):613–624, 2006.

[Jeř07a] Emil Jeřábek. Approximate counting in bounded arithmetic. *Journal of Symbolic Logic*, 72(3):959–993, 2007.

[Jeř07b] Emil Jeřábek. On independence of variants of the weak pigeonhole principle. *J. Log. Comput.*, 17(3):587–604, 2007.

[Jeř23] Emil Jeřábek. Elementary analytic functions in VTC0. *Ann. Pure Appl. Log.*, 174(6):103269, 2023.

[JJ22] Abhishek Jain and Zhengzhong Jin. Indistinguishability obfuscation via mathematical proofs of equivalence. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1023–1034, 2022.

[JJMP25] Abhishek Jain, Zhengzhong Jin, Surya Mathialagan, and Omer Paneth. On succinct obfuscation via propositional proofs. *FOCS 2025 (to appear)*, 2025.

[JKLM25] Zhengzhong Jin, Yael Tauman Kalai, Alex Lombardi, and Surya Mathialagan. Universal snargs for NP from proofs of correctness. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 933–943. ACM, 2025.

[JKLV24] Zhengzhong Jin, Yael Kalai, Alex Lombardi, and Vinod Vaikuntanathan. Snargs under lwe via propositional proofs. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1750–1757, 2024.

[JLS24] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. *Commun. ACM*, 67(3):97–105, 2024.

[Kha22] Erfan Khaniki. Nisan-wigderson generators in proof complexity: New lower bounds. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 17:1–17:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Knu97] Donald Ervin Knuth. *The art of computer programming*, volume 3. Pearson Education, 1997.

[KO17] Jan Krajíček and Igor C. Oliveira. Unprovability of circuit upper bounds in Cook's theory PV. *Logical Methods in Computer Science*, 13(1), 2017.

[KP88] Jan Krajícek and Pavel Pudlák. The number of proof lines and the size of proofs in first order logic. *Arch. Math. Log.*, 27(1):69–84, 1988.

[KP89] Jan Krajícek and Pavel Pudlák. Propositional provability and models of weak arithmetic. In Egon Börger, Hans Kleine Büning, and Michael M. Richter, editors, *CSL '89, 3rd Workshop on Computer Science Logic, Kaiserslautern, Germany, October 2-6, 1989, Proceedings*, volume 440 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 1989.

[KP90] Jan Krajíček and Pavel Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Math. Log. Q.*, 36(1):29–46, 1990.

[KP98] Jan Krajíček and Pavel Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. *Inf. Comput.*, 140(1):82–94, 1998.

[Kra95] Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1995.

[Kra11a] Jan Krajícek. *Forcing with Random Variables and Proof Complexity*, volume 382 of *London Mathematical Society lecture note series*. Cambridge University Press, 2011.

[Kra11b] Jan Krajíček. On the proof complexity of the Nisan-Wigderson generator based on a hard NP $\cap$ coNP function. *Journal of Mathematical Logic*, 11(1), 2011.

[Kra19] Jan Krajíček. *Proof Complexity*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2019.

[Kra24] Jan Krajíček. On the existence of strong proof complexity generators. *Bull. Symb. Log.*, 30(1):20–40, 2024.

[KT90] Jan Krajíček and Gaisi Takeuti. On bounded Σ_1^1 polynomial induction. In *Feasible Mathematics: A Mathematical Sciences Institute Workshop, Ithaca, New York, June 1989*, pages 259–280. Springer, 1990.

[LC11] Dai Tri Man Le and Stephen A. Cook. Formalizing randomized matching algorithms. *Log. Methods Comput. Sci.*, 8(3), 2011.

[LLR24] Jiawei Li, Yuhao Li, and Hanlin Ren. Metamathematics of resolution lower bounds: A TFNP perspective. *CoRR*, abs/2411.15515, 2024.

[LO23] Jiatu Li and Igor C. Oliveira. Unprovability of strong complexity lower bounds in bounded arithmetic. In *Symposium on Theory of Computing* (STOC), 2023.

[LY22] Jiatu Li and Tianqi Yang. 3.1n - $o(n)$ circuit lower bounds for explicit functions. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1180–1193. ACM, 2022.

[Lê14] Dai Tri Man Lê. *Bounded Arithmetic and Formalizing Probabilistic Proofs*. PhD thesis, University of Toronto, 2014.

[MDS25] Yaohua Ma, Chenxin Dai, and Elaine Shi. Quasi-linear indistinguishability obfuscation via mathematical proofs of equivalence and applications. *Cryptology ePrint Archive*, 2025.

[MP20] Moritz Müller and Ján Pich. Feasibly constructive proofs of succinct weak circuit lower bounds. *Annals of Pure and Applied Logic*, 171(2), 2020.

[Nau08] Pavel Naumov. On meta complexity of propositional formulas and propositional proofs. *Arch. Math. Log.*, 47(1):35–52, 2008.

[Nel14] Edward Nelson. *Predicative arithmetic*, volume 32. Princeton University Press, 2014.

[Ngu08] Phuong Nguyen. *Bounded reverse mathematics*. PhD thesis, University of Toronto, 2008.

[Oja04] Kerry Ojakian. *Combinatorics in Bounded Arithmetic*. PhD thesis, Carnegie Mellon University, 2004.

[Oli25] Igor C. Oliveira. Sigact news complexity theory column 124 meta-mathematics of computational complexity theory. *ACM SIGACT News*, 2025.

[Par71] Rohit Parikh. Existence and feasibility in arithmetic. *Journal of Symbolic Logic*, 36(3):494–508, 1971.

[Pic14] Ján Pich. *Complexity Theory in Feasible Mathematics*. PhD thesis, Charles University in Prague, 2014.

[Pic15a] Ján Pich. Circuit lower bounds in bounded arithmetics. *Annals of Pure and Applied Logic*, 166(1):29–45, 2015.

[Pic15b] Ján Pich. Logical strength of complexity theory and a formalization of the PCP theorem in bounded arithmetic. *Logical Methods in Computer Science*, 11(2), 2015.

[PS21] Ján Pich and Rahul Santhanam. Strong co-nondeterministic lower bounds for NP cannot be proved feasibly. In *Symposium on Theory of Computing* (STOC), pages 223–233, 2021.

[Pud86] Pavel Pudlák. On the length of proofs of finitistic consistency statements in first order theories. In *Studies in Logic and the Foundations of Mathematics*, volume 120, pages 165–196. Elsevier, 1986.

[Pud96] Pavel Pudlák. On the lengths of proofs of consistency: a survey of results. In *Collegium logicum*, pages 65–86. Springer, 1996.

[PW85] Jeff Paris and Alex Wilkie. Counting problems in bounded arithmetic. In *Methods in Mathematical Logic: Proceedings of the 6th Latin American Symposium on Mathematical Logic held in Caracas, Venezuela August 1–6, 1983*, pages 317–340. Springer, 1985.

[Raz95] Alexander A Razborov. Bounded arithmetic and lower bounds in boolean complexity. In *Feasible Mathematics II*, pages 344–386. Springer, 1995.

[Raz15] Alexander A Razborov. Pseudorandom generators hard for k-dnf resolution and polynomial calculus resolution. *Annals of Mathematics*, pages 415–472, 2015.

[Sim09] Stephen George Simpson. *Subsystems of second order arithmetic*, volume 1. Cambridge University Press, 2009.

[Sko23] Thoralf Skolem. *Begründung der elementaren Arithmetik durch die rekurrierende Denkweise ohne Anwendung scheinbarer Veränderlichen mit unendlichem Ausdehnungsbereich*. Dybusach, 1923.

[Sti20] John Stillwell. *Reverse mathematics*. Springer, 2020.

[SW14] Rahul Santhanam and Ryan Williams. On uniformity and circuit lower bounds. *Computational Complexity*, 23(2):177–205, 2014.

[SW21] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *SIAM J. Comput.*, 50(3):857–908, 2021.

[Tha05] Neil Thapen. Structures interpretable in models of bounded arithmetic. *Ann. Pure Appl. Log.*, 136(3):247–266, 2005.

[Tur37] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, s2-42(1):230–265, 1937.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987.

[Wil18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the international congress of mathematicians: Rio de janeiro 2018*, pages 3447–3487. World Scientific, 2018.

[WP87] Alex J Wilkie and Jeff B Paris. On the scheme of induction for bounded arithmetic formulas. *Annals of pure and applied logic*, 35:261–302, 1987.

A Cook's Theory PV

A.1 Formal Definition of PV

We first revisit the formalization of PV in [Coo75]. The functions and proofs of PV are defined by simultaneous induction, where each function or proof is associated with a number called its *order*. We first define the base case that introduces functions of order 0 and proofs of order 0:

- ε be a constant symbol.
- $s_0(x), s_1(x), \circ(x,y), \mathsf{TR}(x), \mathsf{ITR}(x,y)$ are function symbols of order 0.
- A *term* of order i is defined by compositions of order-i functions, the constant symbol, and variables, e.g., $\mathsf{ITR}(s_0(s_1(x)), s_1(y))$ is a term of order 0, where x, y are variables.
- An *equation* of order i is of form $s = t$, where s, t are terms of order i.

The definition axioms of the function symbols are proofs of order 0:

$$x \circ \varepsilon = x, \quad x \circ s_i(y) = s_i(x \circ y) \qquad i \in \{0,1\} \tag{A.1}$$

$$\mathsf{TR}(\varepsilon) = \varepsilon, \quad \mathsf{TR}(s_i(x)) = x \qquad i \in \{0,1\} \tag{A.2}$$

$$\mathsf{ITR}(x,\varepsilon) = x, \quad \mathsf{ITR}(x, s_i(y)) = \mathsf{TR}(\mathsf{ITR}(x,y)) \qquad i \in \{0,1\} \tag{A.3}$$

Function introduction rules. For every $i \geq 1$, a function of order i can be introduced according to one of the following two rules.

- (*Composition*). If t is an order-$(i-1)$ term with variables $\vec{x}$, $f_t^{(0)}$ can be introduced as an order-i function with the definition axiom $f_t^{(0)}(\vec{x}) = t$.

- (*Recursion*). If $g(\vec{x})$, $h_0(\vec{x},y,z)$, $h_1(\vec{x},y,z)$, $k_0(\vec{x},y)$, $k_1(\vec{x},y)$ are order-$(i-1)$ functions, and there are order-$(i-1)$ proofs π_i of the equation: [8]

$$\mathsf{ITR}(h_i(\vec{x},y,z), z \circ k_i(\vec{x},y)) = \varepsilon \tag{A.4}$$

  for $i \in \{0,1\}$, then $f^{(1)}_{\Pi}$ (where $\Pi := (g, h_0, h_1, k_0, k_1, \pi_0, \pi_1)$) may be introduced as an order-i function with the definition axioms:

$$f^{(1)}_{\Pi}(\vec{x}, 0) = g(\vec{x}) \tag{A.5}$$

$$f^{(1)}_{\Pi}(\vec{x}, s_i(y)) = h_i(\vec{x}, y, f^{(1)}_{\Pi}(\vec{x}, y)) \;\; i \in \{0,1\} \tag{A.6}$$

Moreover, any function of order $i-1$ is also an order-i function.

Deduction rules. For every $i \geq 1$, an order-i proof of an order-i equation $s = t$ is a sequence of order-i equations $(e_1, e_2, \ldots, e_\ell)$ for some $\ell \in \mathbb{N}$, where $e_\ell =$ "$s = t$" and the equations are introduced one by one following the rules below:

- (*Logical Rules*). The logical rules for equations follow:
  - (L1): If $s = t$ has been introduced, one may introduce $t = s$.
  - (L2): If $s = t$, $t = u$ have been introduced, one may introduce $s = u$.
  - (L3): If $s_1 = t_1, \ldots, s_n = t_n$ has been introduced and $f(x_1, \ldots, x_n)$ is an order-i function symbol with n variables, one may introduce the equation $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$.
  - (L4): If $s = t$ has been introduced, v is an order-i term, and x is an variable, one may introduce $s[x/v] = t[x/v]$, where $s[x/v]$ denote the term obtained from s by substituting all occurrences of x by v.
- (*Definition Axioms*). A definition axiom of an order-i function may be introduced without premise.
- (*Structural Induction*). If $g(\vec{x})$, $h_0(\vec{x},y,z)$, $h_1(\vec{x},y,z)$ are order i functions, and $f_1(\vec{x},y)$, $f_2(\vec{x},y)$ are two functions satisfying that equations

$$f_1(\vec{x},\varepsilon) = g(\vec{x}), \quad f_1(\vec{x}, s_i(y)) = h(\vec{x}, y, f_1(\vec{x},y)),\ i \in \{0,1\} \tag{A.7}$$

$$f_2(\vec{x},\varepsilon) = g(\vec{x}), \quad f_2(\vec{x}, s_i(y)) = h(\vec{x}, y, f_2(\vec{x},y)),\ i \in \{0,1\} \tag{A.8}$$

  have all been introduced, then one may introduce $f_1(\vec{x},y) = f_2(\vec{x},y)$.

This completes the formal definition of PV.

A.2 Representation of PV Equations and Proofs

The standard formalization of the theory PV is sometimes inconvenient as the name of the introduced functions consists of the information about how it is introduced, and therefore a PV proof may contain a function symbol whose name consists of another PV proof. In this paper, we prefer the following formalization of PV proofs and equations.

Definition A.1. A PV proof π is a sequence of proof lines $e_1, \ldots, e_m$, $m \geq 0$, where each line is either an *equation* or a proof line for *function introduction*. The validity of π is defined inductively as follows:

- The empty proof π is valid.
- Suppose that $\pi : e_1, \ldots, e_m$ is valid, and e is an equation such that e can be derived by PV rules or axioms, and all premises have been available in π, then $e_1, \ldots, e_m, e$ is a valid PV proof.

[8] Informally, eq. (A.4) means that: $|h_i(\vec{x},y,z)| \leq |z \circ k_i(\vec{x},y)|$

- Suppose that $\pi : e_1, \ldots, e_m$ is valid, f is a fresh function symbol, and t be a term such that each function symbol in t is either an initial function or has been introduced in π,

  $$e_1, \ldots, e_m, (\mathsf{Composition}, f, t)$$

  is a valid PV proof, and f is introduced in the proof. The function symbol f denotes the PV function introduced by the composition rule using the term t.
- Suppose that $\pi : e_1, \ldots, e_m$ is valid, f is a fresh function symbol, and g, h_0, h_1, k_0, k_1 are either initial functions or introduced function symbols in π. In addition, the equations

  $$e'_i : \mathsf{ITR}(h_i(\vec{x}, y, z), z \circ k_i(\vec{x}, y)) = \varepsilon$$

  for $i \in \{0, 1\}$ have been available in π. Then

  $$e_1, \ldots, e_m, (\mathsf{Recursion}, f, g, h_0, h_1, k_0, k_1)$$

  is a valid PV proof, and f is introduced in the proof. The function symbol f denotes the PV function introduced by limited recursion from $(g, h_0, h_1, k_0, k_1, e'_0, e'_1)$.

In this definitions, the names of the function symbols no longer consist of the information about how it is introduced. Such information is now recorded in the proof lines for function introduction. Similarly, we define equations in the context of a proof as follows:

Definition A.2. An equation e is said to be a valid PV equation in the context of a proof π_e if every function symbol f in e has been introduced in π_e.

A.3 Bounding Values

Let $t(\vec{x})$ be a term in PV with k variables $\vec{x} = (x_1, \ldots, x_k)$. The *bounding value* of the term t is a function $\ell_t(n_1, \ldots, n_k)$ provides an upper bound of the output length of t when $|x_1| \leq n_1, \ldots, |x_k| \leq n_k$. The bounding value of a term is recursively defined on the definition of t:

- (*Initial Functions*). $\ell_\varepsilon = 0$, $\ell_x(n) = n$, $\ell_{s_\sigma}(n) = n + 1$ for $\sigma \in \{0, 1\}$, $\ell_{\mathsf{TR}}(n) = n - 1$, $\ell_{\mathsf{ITR}}(n, m) = n$, $\ell_\circ(n, m) = n + m$.
- (*Functions via Composition*). Let t be a function symbol introduced by composition using a term t', we define $\ell_t(\vec{n}) = \ell_{t'}(\vec{n})$.
- (*Functions via Limited Recursion*). Let $t(\vec{x})$ be a function symbol introduced by limited recursion from (g, h_0, h_1, k_0, k_1), where x_k is the induction variable. We define

  $$\ell_t(\vec{n}) := \ell_g(n_1, \ldots, n_{k-1}) + n_k \cdot (\ell_{k_0}(\vec{n}) + \ell_{k_1}(\vec{n})).$$

- (*Composition*). Suppose that t is the composition of a function symbol $f(y_1, \ldots, y_j)$ and terms $s_1(\vec{x}), \ldots, s_j(\vec{x})$, i.e., $t(\vec{x}) \equiv f(s_1(\vec{x}), \ldots, s_j(\vec{x}))$. We define

  $$\ell_t(\vec{n}) := \ell_f(\ell_{s_1}(\vec{n}), \ldots, \ell_{s_j}(\vec{n})).$$

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il