A Very Brief Introduction to Proof Complexity

Noel Arteche

May 5, 2020

Abstract

This paper is a concise and brief introduction to the field of proof complexity, a subfield of complexity theory with links to logic and proof theory. The paper shortly covers the essential answers given so far to the three fundamental questions guiding research in the field: Are there always short proofs? How are proof systems related? Can I always find a short proof quickly if one exists? These three questions guide the paper in presenting some fundamental concepts and results of proof complexity, such as proof systems, the proof of Haken's exponential lower bound for the Resolution proof system, simulation, domination, the proof system hierarchy and the question of automatizability.

Contents

1 Introduction: the complexity of mathematical belief

According to philosopher Noël Carroll, in essence, human actions are rendered intelligible through narratives: "accounts of how present emerges from the past" [Car09]; how new truths come into being starting from the ones we know.

In mathematics, this is most obvious in the fundamental object of proofs, a concept that boils down to mathematical narratives, sequences of arguments that try to convince the mathematical subject of the truth of a given statement; of its qualification as *theorem.*

Proofs as mathematical objects have been studied since the early 20th century in logic through the meta-theoretical inquiries advanced by David Hilbert in proof theory. Since then, and especially propelled by the initial results of Gödel, Turing and Church on incompleteness and undecidability, proof theory has clearly set the boundaries of what can and cannot be proven.

However, in the same way that computability theory has been extended since the 70s, redrawing the limits of what can be computed by incorporating the ideas of tractability coming from complexity theory, proof theory has opened the door to *proof complexity*: a subfield of both logic and complexity theory, studying the length and complexity of proofs in given proof systems. Now, even inside the realm of what can be proven, some things may be intractable in practice.

Since proofs are narratives and some narratives may be intractably long, this opens the possibility to the existence of statements that although provable in theory, cannot be rendered intelligible in practice. After all, if proofs are narratives, whenever a proof is too long, the reader cannot get to know the end of the story. And if all possible proofs for a particular statement or sign are intractably long, then the mathematical subject will be unable to convince themselves of its truth. These ideas play already a big role in postmodern philosophy of mathematics, such as in Brian Rotman's corporeal semiotic models for mathematics, where the subject triggers mechanical processes based on mathematical language to convince themselves of the truth of certain statements. In his terms, an intractably long proof cannot be *semiotized* and thus it loses all meaning — or it never has any (see [Rot93, Art20]).

In this paper, a concise introduction to the field of proof complexity is presented through three essential topics: the existence of short proofs, the relationships between proof systems and the possibility of mechanically finding proofs. In this section, I first introduce some examples and definitions that will be useful later, as well as the questions that structure the paper. In section 2, I tackle the length of proofs and prove an early famous result in the field: Haken's theorem about the existence of exponential lower bounds in Resolution. In section 3, I present the possibility of moving to a different framework in search of better proving properties and define how proof systems are compared and related in the proof system hierarchy. Finally, section 4 gives some conclusions and briefly

touches upon the question of automatizability, which unfortunately must remain almost in the dark given the length limitations of this paper. The three main bibliographical sources for this text, for those who want to trace back definitions and proofs, are [AB09], [BS14] and [Kra05].

Now, we shall start by looking at some basic examples.

1.1 Polynomial equations

Imagine we are given a system of polynomial equations over $\mathbb{R}^n$,

$$\begin{cases} p_1(x_1, \ldots, x_n) = 0 \\ \vdots \\ p_k(x_1, \ldots, x_n) = 0 \end{cases}$$

If the system has a solution, how do we prove it? Easy! Just give the solution vector, $(x_1, \ldots, x_n) \in \mathbb{R}^n$ (for now, we will not think too much about whether coming up with that solution is difficult or not). This is a good proof, as not only is it short; it can also be checked efficiently (i.e. in polynomial-time) by evaluating the polynomials at that given point.

On the other hand, if the system does not have a solution, how do we prove it? It is not even possible to give all invalid points, as there are uncountably many of them. We need to think a bit further and use something like Hilbert's Nullstellensatz: find k polynomials $q_1, \ldots, q_k$ in $\mathbb{R}^n$ such that

$$\sum_{i=1}^{k} p_i \cdot q_i = 1$$

If such polynomials exist, then the system has no solution, as otherwise a solution would imply $0 = 1$. Additionally, because Hilbert's theorem ensures the existence of those polynomials, this procedure is complete. And, naturally, proofs can be efficiently verified, as it amounts to evaluating and adding polynomials.

1.2 Propositional logic

A more traditional example can be found in propositional logic. Let φ be a Boolean function $\varphi : \{0,1\}^n \to \{0,1\}$ written in CNF as a propositional formula. If $\varphi \in \mathsf{SAT}$... how do I prove it? Easy! Just give a satisfying assignment $(b_1, \ldots, b_n) \in \{0,1\}^n$. Checking the solution (evaluating the formula) can be done efficiently — after all, that's the reason $\mathsf{SAT} \in \mathsf{NP}$. On the other hand, if $\varphi \in \mathsf{UNSAT}$... what would be do? Naïvely, we could give all possible 2^n assignments, but that would take exponentially long to check. Alternatively, we could use a derivation system for propositional logic such as natural deduction, axiomatic derivations or Resolution, amongst others.

Let's have a closer look at Resolution. Suppose the formula $\varphi \in \mathsf{UNSAT}$ is written in CNF with k clauses:

$$\varphi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_k$$

Then, we define a Resolution proof as follows.

Definition 1.1 (Resolution, [AB09])**.** A *Resolution refutation* of φ is a sequence of clauses

$$C_1, \ldots, C_k, C_{k+1}, \ldots, C_s$$

such that the first $C_1, \ldots, C_k$ are the clauses in φ and for every $k < i \leq s$,

$$C_i = A \vee B$$

where $(A \vee x)$ and $(B \vee \neg x)$ are clauses that have already been derived.

The process ends when both the clauses x and $\neg x$ have been derived for some variable x (i.e. we have a contradiction). The size of the proof is the number of clauses, s.

The Resolution proof system is both sound and complete (see [AB09]), and thus it can help us with the task of producing refutations for propositional logic.

1.3 Two fundamental definitions

In the two previous examples we sketched two proof systems from different domains: Hilbert's Nullstellensatz and Resolution. We are now ready to offer a formal definition of *proof system*.

Definition 1.2 (Proof system, [Kra05])**.** Let $L \subseteq \{0,1\}^*$ be a binary language. A *proof system* for L is a binary relation $P(\tau, \rho)$ over L satisfying the following three conditions:

1. Soundness: $\exists \rho : P(\tau, \rho) \Rightarrow \tau \in L$
2. Completeness: $\tau \in L \Rightarrow \exists \rho : P(\tau, \rho)$
3. p-verifiability: $P(\tau, \rho)$ can be decided in polynomial-time

We read $P(\tau, \rho)$ as *theorem τ is proven in system P by proof ρ.*

In the two examples we discussed, we noticed that proofs could be efficiently *read*; efficiently *verified*. This is precisely what p-verifiability means in the previous definition. However, p-verifiability is defined in the length of the proof. Thus, in a p-verifiable system, a proof may still be very long, it's just that reading it

won't take much longer. Whenever we want *short* proofs, we will ask for them to be polynomially bounded in the size of the theorem they prove. If all statements have polynomial-size proofs in a given proof system, we say that this is *p-bounded*.

Definition 1.3 (p-boundedness, [Kra05])**.** A proof system P is *polynomially bounded* or *p-bounded* if and only if there exists $k \in \mathbb{N}^*$ such that for all $\tau, \rho \in \{0,1\}^*$

$$P(\tau, \rho) \Rightarrow \exists \rho' : |\rho'| \leq (|\tau| + k)^k \text{ and } P(\tau, \rho')$$

1.4 Three fundamental questions

Based on the previous definition, we can safely state that we know proof systems for both systems of polynomial equations and propositional tautologies. However, the affirmative proofs in those cases seem to be substantially easier than the negative cases. Whenever we are proving an affirmative statement (whenever a solution exists), the systems are p-bounded. On the other hand, we do not know just by looking at the definition if Hilbert's Nullstellensatz or Resolution are p-bounded too.

This seems to coincide with the generalized notion in mathematics that impossibility proofs are hard, both for finite and infinite domains. Some of the most important theorems in mathematics are related to impossibility proofs: $\sqrt{2} \notin \mathbb{Q}$, $\pi \notin \mathbb{Q}$, the quadrature of the circle, Fermat's Last Theorem, the Riemann Hypothesis, the $\mathsf{P} =_? \mathsf{NP}$ question, etc.

Presented with this challenge of finding impossibility proofs, and with the formal definition of proof system and p-boundedness at hand, we can now introduce the three fundamental questions that guide the research in proof complexity.

1. Is there a proof system that has *short proofs for every statement*?
2. How are proof systems *related*? Are there stronger and weaker proof systems, better and worse proving techniques for the same statements? Is there an *optimal* proof system?
3. Whenever a *short proof exists, can I (automatically) find it*? In a short time?

Interestingly, in the context of first-order logic and infinite structures, these questions are negatively answered by Gödel's and Turing's work on the Entscheidungsproblem, but, surprisingly, they are open for the weaker scenario of propositional logic.

The first and second questions are tackled in sections 2 and 3 of this paper, respectively. The third one, commonly known as *automatizability*, is very briefly mentioned in section 4.

2 On the (non-)existence of short proofs

In this section, we will turn our attention to the first one of the three questions posed above: whether there exist proof systems in which all statements have short arguments. Firstly, we take a look at the connection between computational complexity theory and proof complexity, and offer a result that will make our work easier: the Cook-Reckhow theorem. Then, we present one of the most famous results of the field, known as Haken's theorem: the exponential lower bound of Resolution.

2.1 NP, coNP and the Cook-Reckhow theorem

The question of proofs is closely related to the idea of certificates in computational complexity theory, which allows to phrase the fundamental proof complexity problems in terms of complexity classes. We could say that classic complexity classes are revisited in the light of proof complexity.

Definition 2.1 (The classes NP and coNP, revisited [Kra05])**.** We define the class NP as the class of languages that have a p-bounded proof system:

$$\mathsf{NP} = \{L \subseteq \{0,1\}^* : L \text{ has a } p\text{-bounded proof system}\}$$

As usual, the class coNP is defined as the set of the complementary NP-languages:

$$\mathsf{coNP} = \{L \subseteq \{0,1\}^* : \overline{L} \in \mathsf{NP}\}$$

NP captures proving problems with affirmative answers (those where a finite object satisfying the solution provides a proof, e.g SAT), while coNP contains the problems with negative answers, e.g. UNSAT.

Then, the question of whether short proofs always exist, can be phrased by asking whether the classes NP and coNP are the same. In this regard, the conjecture goes that this is unfortunately not the case.

Conjecture 2.1. $\mathsf{NP} \neq \mathsf{coNP}$.

Thus, according to the conjecture, in every proof system for coNP languages there are statements that can only be proven in super-polynomial size.

However, is there any interest in proving this conjecture for complexity theory as a whole? There is, regarding the fundamental $\mathsf{P} =_? \mathsf{NP}$ question. As we know, the class P is closed under complementation, so if $\mathsf{P} = \mathsf{NP}$, then $\mathsf{NP} = \mathsf{coNP}$. Then, by the contrapositive, if the previous conjecture is true and $\mathsf{NP} \neq \mathsf{coNP}$, we would be proving that $\mathsf{P} \neq \mathsf{NP}$. On the other hand, if the conjecture turns out to be false, that would not immediately settle the $\mathsf{P} =_? \mathsf{NP}$ question.

The reader may now feel that the quest for short proofs is a daunting challenge — even more that the quest for polynomial-time Turing machines. After all, proof systems are very different from one domain to another. Clearly, reasoning about Resolution seems to be completely different from reasoning about polynomial equations, and that makes general non-existence arguments very complicated. Surprisingly, this is not the case. We can forget about all possible problems and focus only on propositional tautologies. In fact, p-bounded proof systems exist for coNP-languages if and only if such a proof system exists just for propositional tautologies. This is given by the following theorem.

Theorem 2.1 (Cook-Reckhow, [BS14])**.** *Let* TAUT *denote the language of propositional tautologies. Then* NP = coNP *if and only if a p-bounded proof system exists for* TAUT*.*

Proof. SAT is NP-complete and UNSAT is the complementary of SAT, so UNSAT is coNP-complete. And because for any Boolean formula,

$$\varphi \in \mathsf{TAUT} \Leftrightarrow \neg\varphi \in \mathsf{UNSAT}$$

TAUT is also coNP-complete. Now, if NP = coNP, then TAUT $\in$ NP and thus it has a p-bounded proof system. On the other hand, if TAUT has a p-bounded proof system, since TAUT is coNP-complete, all other coNP languages will have a p-bounded proof system, so NP = coNP.

□

2.2 Exponential lower bounds in Resolution

Inspired by the Cook-Reckhow theorem, we now turn our attention to proof systems like Resolution. We may ask if the Resolution proof system we looked at earlier is p-bounded. Based on the NP $\neq$ coNP conjecture, we can hypothesize that the answer will be negative. Indeed, we can show that there exist propositional tautologies whose proofs in Resolution are all exponential-size.

The two main techniques used to show this are known as bottleneck methods and feasible interpolation theorems. Though interpolation results are more modern and sometimes more intuitive, they depend on some notions coming from circuit complexity, which are out of the scope of this text. Therefore, for the sake for concision, we will focus on showing an exponential lower bound through the first technique, a result known as Haken's theorem, which proves that the Pigeonhole Principle needs exponential-size proofs in Resolution.

2.2.1 The Pigeonhole Principle

Simple, almost trivial theorems, are sometimes hard to prove when we change the framework imposed by the proof system. Proof complexity theorists use these

hard-to-prove theorems to provoke combinatorial explosions in certain proof systems.

The result we will use is a simple combinatorial property of sets, underlying many relevant theorems across mathematics, known as the *Pigeonhole Principle*: if we have m pigeons and n holes, and $m > n$, then there must be a hole with more than one pigeon.

Theorem 2.2 (The Pigeonhole Principle)**.** *Let $m, n \in \mathbb{N}^*$. If $m > n$, then there exists no bijection from the set $\{1, \ldots, m\}$ to the set $\{1, \ldots, n\}$.*

Proof. Straightforward induction over $\mathbb{N}^*$. □

Most undergraduate textbooks would say that the straightforward induction is left as an exercise to the reader. But what happened if the proof system we worked with did not have induction? What if we tried to prove the Pigeonhole Principle in a system like Resolution? Very likely, the system would get lost in the locality of the formulae and it would need exponentially many steps to prove the tautology — it would assign some pigeons, see that it does not work, start over, and so on.

In order to study the Pigeonhole Principle under Resolution, we first need to encode the theorem into propositional logic formulae written in Conjunctive Normal Form (CNF). We will work with the formula PHP_n^m stating that "there exists a bijection from $[m]$ to $[n]$". Clearly, if $m > n$, PHP_n^m is always false: $PHP_n^m \in \mathsf{UNSAT}$ and $\neg PHP_n^m \in \mathsf{TAUT}$.

Our formulae will be written in terms of variables $p_{i,j}$. Semantically, $p_{i,j}$ is true if and only if pigeon i is assigned to hole j.

These variables are arranged in two types of clauses:

1. Each pigeon is assigned to some hole

$$(p_{i,1} \vee \cdots \vee p_{i,n})$$

2. The k-th hole doesn't get both the i-th and j-th pigeons

$$(\neg p_{i,k} \vee \neg p_{j,k})$$

Thus, the complete Pigeonhole Formulae have the following form:

$$PHP_n^m = \bigwedge_{i \leq m} (p_{i,1} \vee \cdots \vee p_{i,n}) \wedge \bigwedge_{\substack{i,j \leq m \\ k \leq n}} (\neg p_{i,k} \vee \neg p_{j,k})$$

2.2.2 Haken's theorem

Based on the previous propositional formulae, in 1985, Haken managed to prove an exponential lower bound for the Resolution proof system. His original argument (see [Hak85]), based on bottleneck counting, is somewhat cumbersome. We present here a refined version by Beame and Pitassi (see [BP96]). The idea behind the proof is simple: we convert a Resolution refutation of PHP^n_{n-1} of length s into a *monotonized* refutation, also of length s. This new refutation is no longer valid, but for *critical assignments* — some special assignments — it still derives a contradiction. Based on this we can derive a contradiction on the length of the monotonized refutation being subexponential, but since the original refutation has the same length as the monotonized one, the original refutation must also be exponentially long.

Before going into the complete proof, we present the two fundamental ingredients: *monotonized refutations* and *critical assignments*.

Suppose a Resolution refutation ρ of PHP^n_{n-1} is of the form:

$$\rho = C_1, \ldots, C_k, C_{k+1}, \ldots, C_s$$

For each clause C, we get a clause C' that is *monotone* (it doesn't have negative literals) by replacing each literal $\neg p_{i,j}$ by $\bigvee_{k \neq i} p_{k,j}$. We get a *monotonized refutation*:

$$\rho' = C'_1, \ldots, C'_k, C'_{k+1}, \ldots, C'_s$$

Naturally, the monotonized refutation is no longer a valid refutation, because it does not derive a contradiction for all possible assignments. However, it still works fine for a subset of assignments, known as *critical assignments*: an assignment α where $n-1$ of the n pigeons are successfully assigned to different holes, and the i-th pigeon is left unassigned. We call that α an *i-critical* assignment.

It turns out that if we take a Resolution refutation of PHP^n_{n-1} and turn it into a monotone one, we can prove that there is always a *huge clause*: a clause with at least $2n^2/9$ variables.

Lemma 2.1 (Existence of huge clauses, [AB09])**.** *Every monotonized Resolution refutation of PHP^n_{n-1} must contain a clause with at least $2n^2/9$ variables.*

Proof. For each clause C in the monotonized refutation, let

$$witness(C) = \{i : \text{there is an } i\text{-critical } \alpha \text{ such that } C(\alpha) = 0\}$$

Let $t = |witness(C)|$ and fix an $i \in witness(C)$ and build an i-critical α that falsifies C. For each $j \notin witness(C)$, obtain an assignment α' by mapping pigeon i to whatever hole l pigeon j was mapped in α and leave j unassigned. This is

a j-critical assignment, but since $j \notin witness(C)$ we get that α' must satisfy C. And because C is monotone, that means variable $p_{i,l}$ is in C.

Repeat the previous process for each one of the $n-t$ values of $j \notin witness(C)$ with the same α. We get that there are at least $n-t$ distinct variables in C. Repeating the process for each $i \in witness(C)$, we conclude that C has at least $t(n-t)$ variables.

Note that since every clause C is derived from previous clauses $X = (A \vee x)$ and $Y = (B \vee \neg x)$, if a critical assignment falsifies C it also falsifies one of X or Y. As a result,

$$|witness(C)| \leq |witness(X)| + |witness(Y)|$$

Thus if C is the first clause in the derivation such that $|witness(C)| > n/3$, then

$$\frac{n}{3} < |witness(C)| < \frac{n}{3} + \frac{n}{3} = \frac{2n}{3}$$

Note that such a C exists: the initial clause has $|witness(C_1)| = 0$ and the final clause (the contradiction) has $|witness(C_s)| = n$. Then if C_s is derived from some clauses A and B, then it must be the case that $|witness(A)| + |witness(B)| = n$, and then at least one of them qualifies as a clause whose witness number is bigger than $n/3$.

Under the condition $n/3 < t < 2n/3$ the following inequality holds:

$$t(n-t) > \frac{2n^2}{9}$$

Then, because C has at least $t(n-t)$ variables it also has at least $2n^2/9$ and C is huge.

□

Using the notion of monotonized refutations and the lemma we just proved, we can get to Haken's original result.

Theorem 2.3 (Haken's theorem, [AB09])**.** *For any $n \geq 2$, every refutation of PHP^n_{n-1} in the Resolution proof system has size at least $2^{n/20}$.*

Proof. Let ρ be an arbitrary Resolution refutation of PHP^n_{n-1} of size s. We convert it into a monotonized refutation ρ', also of length s.

We know (from the lemma) that for every monotonized refutation of PHP^n_{n-1}, there is one *huge* clause: a clause C' containing at least $2n^2/9$ variables.

We say a clause is *large* if contains at least $n^2/10$ variables. Let ℓ denote the number of large clauses. Because $n^2/10 < 2n^2/9$, the lemma gives us that there exists at least one large clause: $\ell \geq 1$.

The averaging argument can be applied to get that there is some variable $p_{i,j}$ occurring in 1/10-th of the large clauses. If we set $p_{i,j} = 1$ and $p_{i,j'} = 0$ for all $j' \neq j$, we get that all the clauses where $p_{i,j}$ shows up are now made true and can be removed.

After removing the clauses made true, we are left with at most $9/10\ell$ large clauses and a sequence of clauses that constitutes a monotonized refutation of PHP_{n-2}^{n-1}.

If we repeat this process $t = \log_{10/9} \ell$ times, we end up with a monotonized refutation of PHP_{n-t-1}^{n-t} that has lost all of the original large clauses.

Now assume for a contradiction that the number of clauses s in the refutation ρ' is not exponential: $s < 2^{n/20}$. Then the number of large clauses is not exponential either: $\ell < 2^{n/20}$. Therefore:

$$\log_{10/9} \ell < \log_{10/9} 2^{n/20} \Rightarrow t < n \cdot \frac{\log_{10/9} 2}{20} \approx n \cdot 0.32 < \frac{n}{3} \Rightarrow t < \frac{n}{3}$$

The refutation of PHP_{n-t-1}^{n-t} has no clauses larger than $n^2/10$. But by the lemma it does have a huge clause for its level i.e. a huge clause of $2(n-t)^2/9$ variables. But since $t < n/3$, that gives $2(n-t)^2/9 > n^2/10$, so the refutation has a large clause, which is a contradiction.

Thus the monotonized refutation is exponentially long. But the length of the monotonized refutation is the same as the original refutation, so every Resolution refutation of PHP_{n-1}^{n} is exponentially long.

□

3 The bigger picture: the proof system hierarchy

In the light of Haken's theorem, we may feel annoyed by the perspective of a simple proof system like Resolution producing intractably long proofs in some cases. We may then try to look to other derivation systems in propositional logic.

After all, many different calculus exist for propositional tautologies: truths tables, axiomatic derivation systems or Frege systems, Gentzen's sequent calculus, natural deduction, the Davis-Putnam procedure, etc. Some of these can easily be discarded, like truth tables, because they always produce exponential-size proofs; or the Davis-Putnam procedure, for being a rudimentary version of Resolution. We might then decide to look even further. If we change the domain of the problem, we might get a completely different, much powerful proof system. Will this be the case of the polynomial equations presented at the beginning?

3.1 Simulation: from logic to polynomial equations

It turns out that SAT can be solved in the continuum, by translating every formula into a system of polynomial equations.

Consider a 3SAT instance, $\varphi : \{0,1\}^n \to \{0,1\}$ expressed in CNF as $\varphi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_k$.

We can translate the 3CNF formula to a system of polynomial equations. For a clause like $C = (x \vee \neg y \vee z)$, add the equations:

$$\begin{cases} (1-x)y(1-z) = 0 \\ x(x-1) = 0 \\ y(y-1) = 0 \\ z(z-1) = 0 \end{cases}$$

We get a system of $k + n$ polynomial equations.

How can we solve this? Well, if $\varphi \in \mathsf{SAT}$, we can solve the system numerically (via Gauss-Newton, Levenberg-Marquardt, etc.). On the other hand, if $\varphi \in \mathsf{UNSAT}$, we could use Hilbert's Nullstellensatz.

It is clear then that Hilbert's Nullstellensatz can be used to solve propositional tautologies. One might even adventure that, with some more work, a proof in Resolution could be converted into Hilbert's Nullstellensatz. This would allow us to say that Hilbert's Nullstellensatz *simulates* Resolution. This is the first relationship we define between proof systems.

Definition 3.1 (Simulation, [BS14])**.** A proof system P *simulates* another system Q whenever:

1. P and Q can prove the same language L, i.e. for every $\tau \in L$

$$\exists \rho_1 : P(\tau, \rho_1) \Leftrightarrow \exists \rho_2 : Q(\tau, \rho_2)$$

2. Proofs in Q can be transformed into proofs in P via a polynomial-time computable function f:

$$Q(\tau, \rho) \Leftrightarrow P(\tau, f(\rho))$$

3.2 Domination, optimality and the proof system hierarchy

Simulation is a way to relate proof systems and compare them. If Hilbert's Nullstellensatz turned out to be able to prove the Pigeonhole Formulae in polynomial size, we might even say that this system is somehow stronger. Unfortunately, this is not the case: the Pigeonhole Formulae also need exponential-size proofs in this system as well as in some derived ones, like the polynomial calculus.

However, there exists a system in which this does not happen: Frege systems. In Frege systems, the Pigeonhole Principle is proven in polynomial size. Besides, existing lower bound techniques such as interpolation are likely not to work (under *reasonable* complexity assumpions like "RSA is secure").

Naturally, we ask, can Frege systems simulate Resolution? And Hilbert's Nullstellensatz? Sometimes, finding the exact transformation between proof systems is tiresome and not particularly useful. In order to define a more convenient notion of comparison between proof systems we define a weaker notion of *simulation* known as *domination*, where the proofs don't have to be converted in polynomial time; it's enough that their size is polynomially bounded, even if we don't know how to efficiently transform ones into others.

Definition 3.2 (Domination, [BS14])**.** A proof system P *dominates* another system Q if there is a polynomial $f : \mathbb{N} \to \mathbb{N}$ such that for any string $\tau \in L$,

$$\exists \rho : Q(\tau, \rho) \Leftrightarrow \exists \rho' : |\rho'| \leq f(|\rho|) \text{ and } P(\tau, \rho')$$

Using this new notion, we can relate all the existing proof systems into a hierarchy, known as the *proof system hierarchy*. A depiction of the hierarchy with some known relationships can be appreciated in Figure 1.

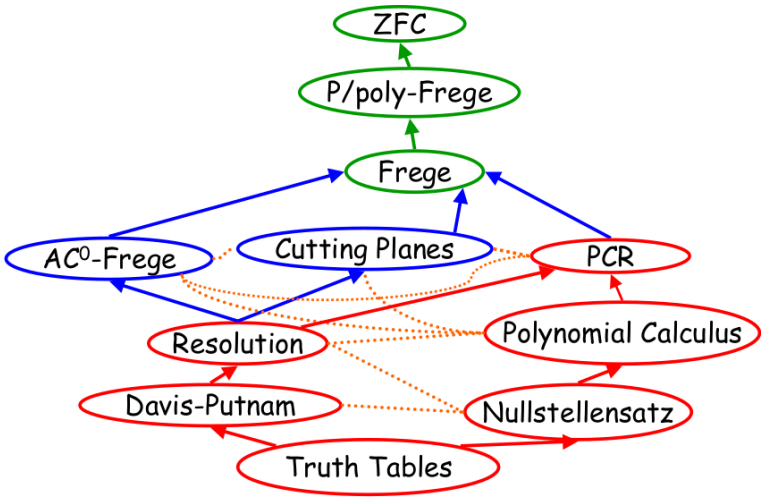

Figure 1: The proof system hierarchy (as depicted in [BS14], no legend provided).

Looking at the hierarchy, it is easy to see that Frege systems are some of the most powerful proof systems known to date, only dominated by variants of it or by the Zermelo-Fraenkel with Axiom of Choice set-theoretic axioms.

A natural question is whether there is a proof system sitting in the top of this hierarchy; a system capable of dominating any other proof system. This leads to the notion of *optimality*.

Definition 3.3 (Optimality)**.** A proof system P is *optimal* if it dominates all other proof systems. Besides, it is *strongly* optimal if it also simulates all other proof systems.

Although optimal proof systems may exist, we do not think any of them can be p-bounded.

Conjecture 3.1. *There exists no optimal p-bounded proof system for* TAUT.

This conjecture draws a path towards proving the NP $\neq$ coNP conjecture presented earlier. First, find an optimal proof system for TAUT; second, show that it is not p-bounded by proving an exponential lower bound on it. What we did with Resolution is just the second step. Unfortunately, since Resolution is dominated by other systems, it is not optimal, so the NP $=_?$ coNP question remains open.

4 Conclusion and the quest for automatizability

In this paper we have offered a concise review of some of the main issues of proof complexity, as well as a more technical detail into one of the most yearned results: exponential lower bound theorems.

Looking back at the three questions posed at the beginning, we can briefly summarise what we discussed.

1. Is there a proof system that has *short proofs for every statement*?
   Complexity theorists conjecture the answer is no, and thus NP $\neq$ coNP. We have proven that, for instance, the Resolution proof systems doesn't always have short proofs.

2. How are proof systems *related*?
   We have defined the notions of *simulation*, *domination* and *optimality* and have seen how the currently known relations between proof systems are expressed in the proof system hierarchy.

There was, however, a third question that we did not answer: whenever a *short proof exists, can I (automatically) find it*? In a short time? Let us give some brief remarks on this question, a more practical matter known as *automatizability*.

Definition 4.1 (Automatizability)**.** A proof system P is said to be *automatizable* if it is possible to find a proof in time polynomial in the size of the smallest one.

Note that, in general, proof systems like Resolution are non-deterministic, so it is not trivial to see that a proof will be found shortly, let alone to produce the shortest one. The quest for automatizability is a recurring theme amongst researchers working on SAT-solvers or QBF-solvers. After all, these are built on top of proofs systems like Resolution or Q-Resolution, extended with heuristics that help in finding the proofs more quickly. Naturally, we would like these systems to be automatizable, so that if, say, a linear-size proof exists for a formula, we can find it in linear-time when running the algorithm implementing this system.

Besides, the question of automatizability brings us closes to the P $=_?$ NP question. If we found a p-bounded proof system for a language like TAUT we

would only collapse $\mathsf{NP} = \mathsf{coNP}$. If, additionally, this system was automatizable, we could find the proofs in time polynomial in the size of the smallest one. But since all statements have polynomial-size proofs, we could find polynomial-size proofs in polynomial time, which would finally imply $\mathsf{P} = \mathsf{NP}$.

References

[Aar16] Scott Aaronson. P vs. NP. In *Open problems in mathematics*, pages 1–122. Springer, 2016.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[Art20] Noel Arteche. *God, Infinity and Language: Semiotic Perspectives on the Philosophy of Mathematics*. 2020.

[BP96] Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 274–282. IEEE, 1996.

[BS14] Paul Beame and Ashish Sabharwal. *Proof Complexity Lecture Notes*. 2014.

[Car09] Noël Carroll. *On Criticism*. Routledge, 2009.

[Hak85] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[Hey20] Jan Heylen. *Intermediate Logic*. Compiled from the Open Logic Text, 2020.

[Kra05] Jan Krajíček. Proof complexity. In *European congress of mathematics (ECM)*, pages 221–231. Stockholm, Sweden, Zurich: European Mathematical Society, 2005.

[Kra19] Jan Krajíček. *Proof Complexity*, volume 170. Cambridge University Press, 2019.

[Pat19] Panos Patrinos. *Optimization - Course Notes*. KU Leuven, 2019.

[Rot93] Brian Rotman. *Ad Infinitum: The Ghost in Turing's Machine - An Essay in Corporeal Semiotics*. Stanford University Press, 1993.

[Ste12] Jacqueline Stedall. *The History of Mathematics: A Very Short Introduction*. Oxford University Press, 2012.

[Tor98] Roberto Torretti. *El paraíso de Cantor: la tradición conjuntista en la filosofía matemática*. Editorial Universitaria, 1998.