Polynomial Size Proofs for the Propositional Pigeonhole Principle

An exploration into proof theory complexity

Author:
Renate Eilers
3701441

First supervisor:
Dr. Rosalie Iemhoff
Second supervisor:
Prof. Dr. Albert Visser

Utrecht University

A thesis submitted in partial fulfilment (7,5 ECTS) of
the degree of Bachelor of Science

July 9, 2014

Contents

Introduction

This thesis was submitted for the completion of the Artificial Intelligence Bachelor's programme at the Department of Philosophy, Faculty of Humanities of Utrecht University. It was written under the supervision of Dr. Rosalie Iemhoff, teacher and researcher at the Utrecht University Department of Philosophy. The research found in this paper is conducted by literary analysis.

This thesis is aimed at exploring the area of proof theory complexity, an area in mathematical logic concerned with the sizes of proofs in various proof systems. This subject is of both theoretical and practical importance to the area of artificial intelligence: proving certain properties for certain proof systems yields important implications for big questions such as whether $P = NP$ or $coNP = NP$. This gives us new insights into the effectiveness of current AI techniques, and of computational methods as a whole. Practical implications of research into proof theory complexity have to do with improving the efficiency of automated theorem proving.

The document is aimed at a public already familiar with propositional logic and basic complexity theory. It document will begin with an introduction into proof theory. Here some the basics of propositional logic can be found. It is meant to be read as a refresher rather than a comprehensive introduction. After the introduction a small number of proof systems will be showed and discussed. For each of these systems basic properties such as soundness and completeness will be proved.

The section after that will repeat some important notions in proof theory complexity. Again, this is not comprehensive and should be read only to freshen up knowledge in the area. This section discusses some of the important aspects and possible implications of proof theory complexity.

The thesis concludes by an elaborate example. It will discuss and explicate polynomial size proofs for the pigeonhole principle in two different proof systems.

1 Propositional proof theory complexity

In this section we define what we mean exactly when we talk about proofs. We will look at three proof systems and discuss some of their properties. The proof systems treated in this text are Frege systems, sequent calculus and resolution refutation. Each of these systems has different properties and is used for different purposes. Frege systems are often used in proof theory complexity. Sequent calculus is important to metatheory: the study of properties of logic systems. Resolution refutations are well suited for automated theorem proving. These three systems will be discussed in more detail later on in this section. First, however, we will cover the more basic elements of proof theory: here we will define propositional languages and what we mean by 'proof' exactly.

1.1 Propositional logic and proofs

Proof theory is the area in mathematical logic which studies mathematical proof. Let us start by explicating what we mean by the word 'proof' exactly: a proof is a means to show the correctness of a theorem. For this text, we limit the language of our theorems to propositional formulas.

A propositional language is built from the following components:

- The variables, V: $p_1, p_2, p_3, \ldots$
- The constants $\top$ for '1' or $True$ and $\bot$ for '0' or $False$
- Logical connectives, e.g.: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \oplus, \ldots$

The propositional variables $p_1, p_2, p_3, \ldots$ represent a truth value of either 1 or 0. A truth-assignment is an assignment of truth values to the propositional variables in a formula. It is a function $\tau : V \rightarrow \{0, 1\}$. The truth-assignment τ of a formula A determines the truth value of this formula. We write $\bar{\tau}(A)$ for the truth value of A under the truth assignment τ. Table 1 shows the truth values for some logical operators under their corresponding truth-assignments.

A propositional language is complete when every Boolean function can be formed by its connectives.

Table 1: The truth values of certain formulas under different truth assignments

$\tau(A)$	$\tau(B)$	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$	$A \oplus B$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	0	0	1
1	1	0	1	1	1	1	0

A propositional formula is considered a *tautology* when it yields a truth value of 1 under every truth assignment. One way to verify wether a propositional formula is a tautology would be to simply construct a truth table for it, but this is very inefficient. The method of truth tables is of exponential complexity: if

a formula consists of n distinct propositional variables, our truth table would have 2^n entries. We will address the topic of proof complexity in more detail in a later section, but for now it suffices to understand that the method of truth tables is not very efficient. Since truth tables will not suffice when the size of our formulas increase, we will have to look at different ways of constructing proofs. In [7] John Harrison gives the following description of what a proof is:

> A formal proof is a proof written in a precise artificial language that admits only a fixed repertoire of stylized steps. This formal language is usually designed so that there is a purely mechanical process by which the correctness of a proof in the language can be verified.

As stated above, the artificial language mentioned by Harrison will be limited to propositional languages for this thesis. The fixed repertoire of steps Harrison mentions are dependent on the proof system in which the proof is carried out. The following sections discusses several common proof systems that embody Harrisons description: Frege systems, sequent calculus and resolution refutations.

1.2 Frege systems

A Frege proof system is a three tuple $(\mathcal{L}, \mathcal{R}, \mathcal{A})$, where $\mathcal{L}$ can be any propositionally complete language, $\mathcal{R}$ is a set of rules and $\mathcal{A}$ is a set of axioms.

To better understand this definition, consider the following concrete system $\mathcal{F}$, where $\mathcal{L} = \{\wedge, \vee, \neg, \rightarrow\}$. Note that this language is complete. The only rule of inference that our system allows is the modus ponens. Recall that the modus ponens is the rule which allows for any formulas A and B, the formula B to be inferred from A and $A \rightarrow B$:

$$\frac{A \qquad A \rightarrow B}{B}$$

Our axioms are the following (for arbitrary A, B and C). This is the Frege system used in [3].

1. $A \rightarrow (B \rightarrow A)$
2. $(A \rightarrow B) \rightarrow (A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C)$
3. $A \rightarrow A \vee B$
4. $B \rightarrow A \vee B$
5. $(A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow (A \vee B \rightarrow C)$
6. $(A \rightarrow B) \rightarrow (A \rightarrow \neg B) \rightarrow \neg A$
7. $\neg\neg A \rightarrow A$
8. $A \wedge B \rightarrow A$
9. $A \wedge B \rightarrow B$
10. $A \rightarrow B \rightarrow A \wedge B$

The $\mathcal{F}$-proof of a formula B from Γ, where Γ is a set of formulas, is a sequence of formulas, where each formula is either an $\mathcal{F}$-axiom, a formula in Γ, or inferred from two earlier formulas in the proof by modus ponens. The final formula in the proof has to be B.

When there is a syntactic proof for B following from Γ, we write $\Gamma \vdash B$. When B is a valid semantic consequence of Γ, we write $\Gamma \models B$.

For an example of a $\mathcal{F}$-proof, see the $\mathcal{F}$-proof for $A \wedge B \vdash B \wedge A$ below:

1.	$A \wedge B$	Hypothesis 1
2.	$A \wedge B \to A$	Axiom 8
3.	$A \wedge B \to B$	Axiom 9
4.	A	Modus Ponens: 1,2
5.	B	MP: 1,3
6.	$A \to (B \to A)$	Axiom 1
7.	$B \to A$	Modus Ponens: 4, 6
8.	$B \to A \to B \wedge A$	Axiom 10
9.	$B \wedge A$	Modus Ponens: 7, 8.

For our proof system to be correct we require it to satisfy two important conditions: *soundness* and *completeness*. Recall that soundness means that every theorem that can be proved by the system is valid: if $A_1, \ldots, A_i \vdash B$, then $A_1, \ldots, A_i \models B$. Completeness is in a way the inverse of soundness. It requires that for every valid formula there is an $\mathcal{F}$-proof: if $A_1, \ldots, A_i \models B$, then $A_1, \ldots, A_i \vdash B$. We want our proof system to be sound, because it would not be worth very much if it allows us to proof invalid formulas. Completeness is also a very important property, since a system that is not able to proof every valid formula is limited.

Soundness of $\mathcal{F}$ 1. *$\mathcal{F}$ is sound: if $A_1, \ldots, A_i \vdash B$, then $A_1, \ldots, A_i \models B$.*

Proof. We will proof the soundness of $\mathcal{F}$ by induction over the number of formulas in the proof, but first we have to show the validity of our axioms and our rule, modus ponens. This is easily done by showing that all $A \in \mathcal{A}$ are in fact tautologies. This can be done by constructing truth tables for our axioms, which is very straightforward and requires a lot of space and is therefore omitted here. The validity of the modus ponens can also be verified using a truth table: $(A \wedge (A \to B)) \to B)$ is a tautology.

Now for the proof let $A_1, \ldots, A_n, B$ be an $\mathcal{F}$-proof of B, then for A_n:

- If $n = 1$, A_n is an axiom, and every axiom is a valid formula.
- If $n > 1$, A_n is either an axiom, and thus valid, or A_n is derived from two previous lines A_i and A_j and the modus ponens. By the inductive hypothesis, A_i and A_j are valid, and so A_n must be valid as well because the modus ponens preserves validity.

□

The proof of completeness of our $\mathcal{F}$-system is as given by S. Buss in [3]. We show the completeness proof can be reduced to *if B is a tautology, then $\vdash B$*, since if $A_1, \ldots, A_i \models B$, then $A_1 \to (\ldots \to (A_i \to B) \ldots)$ is a tautology. This is called the Deduction Theorem, and it is proven below. Now to prove that *if B is a tautology, then $\vdash B$* we require a number of auxiliary lemmas. These are proven below:

Lemma A 1. *$\vdash \phi \to \phi$.*

Proof.

1.	$(\phi \to (\phi \to \phi)) \to (\phi \to (\phi \to \phi) \to \phi) \to (\psi \to \phi)$	Ax 2
2.	$\phi \to (\phi \to \phi)$	Ax 1
3.	$\phi \to (\phi \to \phi) \to \phi$	Ax 1
4.	$(\phi \to (\phi \to \phi) \to \phi) \to (\psi \to \phi)$	MP: 2,1
5.	$(\psi \to \phi)$	MP: 3,4

□

Deduction Theorem 1. $\Gamma, \phi \vdash \psi$ *if and only if* $\Gamma \vdash \phi \to \psi$

Proof. For the direction from right to left, assume that $\Gamma \vdash \phi \to \psi$. If the is the case, then we also have that $\Gamma, \phi \vdash \phi \to \psi$ and of course that $\Gamma, \phi \vdash \phi$. From here we can apply the modus ponens to ϕ and $\phi \to \psi$ to proof that $\Gamma, \phi \vdash \phi$

For the other direction, assume that $\Gamma, \phi \vdash \psi$. This means that there is a proof $A_1, \ldots, A_n$ of ψ from $\Gamma \cup \{\phi\}$, where A_n is ψ. We prove by induction over i that for every i: $\Gamma \vdash \phi \to A_i$.

When $i = 1$, we separate two cases:

Case 1 A_1 is either an axiom, or $A_1 \in \Gamma$. By axiom one, we have that $A_1 \to (\phi \to A_1)$. By applying the modus ponens to this and A_1 we can derive $\phi \to A_1$.

Case 2 $A_1 = \phi$. In this case we have to proof that $\phi \vdash \phi$. For this proof, see the auxiliary lemma *a* above.

Now for the induction step we can assume that for every $k < i$, $\Gamma \vdash \phi \to A_k$. Again we seperate two cases:

Case 1 A_i is an axiom or $A_i \in \Gamma \cup \{\phi\}$. This step is identical to proving the base case by simply replacing B_1 by B_i.

Case 2 A_i follows by the modus ponens from A_j, A_m, where $j < m < i$. A_m must then be of the form $A_j \to A_i$ and so by the inductive assumption we have that $\Gamma \vdash \phi \to (A_j \to A_i)$ and also $\Gamma \vdash \phi \to A_j$. From here we can proof $\phi \to A_i$:

1.	$(\phi \to A_j) \to (\phi \to (A_j \to A_i)) \to (\psi \to A_i)$	Ax 2
2.	$(\phi \to A_j)$	Given
3.	$(\phi \to (A_j \to A_i)) \to (\psi \to A_i)$	MP: 2,1
4.	$\phi \to (A_j \to A_i)$	Given
5.	$\phi \to A_i$	MP: 4,3

From both cases follows that $\Gamma \vdash \phi \to A_i$, and so this concludes our proof of the deduction theorem. □

Lemma B 1. $\phi \to \psi \vdash \neg\psi \to \neg\phi$.

Proof. By the duduction theorem it suffices to show that $\phi \to \psi, \neg\psi \vdash \neg\phi$.

1.	$\neg\psi \to (\phi \to \neg\psi)$	Ax 1
2.	$\neg\psi$	Given
3.	$\phi \to \neg\psi$	MP: 2,1
4.	$(\phi \to \psi) \to (\phi \to \neg\psi) \to \neg\phi$	Ax 6
5.	$\phi \to \neg\psi$	MP: 3,4
6.	$\neg\phi$	MP: 5,4

□

Lemma C 1. $\phi, \neg\phi \vdash \psi$.

Proof.

1.	$\phi \to (\neg\psi \to \phi)$	Ax 1
2.	$\phi \vdash \neg\psi \to \phi$	Deduction Theorem, 1
3.	$\phi \vdash \neg\phi \to \neg\neg\psi$	Lemma B, 2
4.	$\phi, \neg\phi \vdash \neg\neg\psi$	Deduction Theorem, 3
5.	$\neg\neg\psi \to \psi$	Ax 7
6.	ψ	MP: 4,5

□

Lemma D 1. $\neg\phi \vdash \phi \to \psi$.

Lemma D follows directly from Lemma B and the Deduction Theorem.

Lemma E 1. $\psi \vdash \phi \to \psi$.

Proof. $\psi \to (\phi \to \psi)$ is Axiom 1 with ψ substitued for A and ϕ substituted for B. If we apply the Deduction Theorem to this, we get $\psi \vdash \phi \to \psi$.

□

Lemma F 1. $\phi, \neg\psi \vdash \neg(\phi \to \psi)$.

Proof. We are going to proof $\phi \vdash \neg\psi \to \neg(\phi \to \psi)$, which is to Lemma F equal by the Deduction Theorem. By Lemma B and the Deduction Theorem this in turn equals $\phi, \phi \to \psi$. This follows from the modus ponens directly.

□

Lemma G 1. $\phi, \psi \vdash \phi \wedge \psi$.

Proof.

1.	$\phi \to \psi \to (\phi \wedge \psi)$	Ax 10
2.	ϕ	Given
3.	$\psi \to (\phi \wedge \psi)$	MP: 2,1
4.	ψ	Given
5.	$\phi \wedge \psi$	MP: 4,3

□

Lemma H 1. $\neg\phi \vdash \neg(\phi \wedge \psi)$ *and* $\neg\psi \vdash \neg(\phi \wedge \psi)$

Proof. By the Deduction Theorem it suffices to show $\vdash \neg\phi \to \neg(\phi \wedge \psi)$. By lemma B, this equals $(\phi \wedge \psi) \to \phi$, which is Axiom 8 and thus clearly true. The second proof is done by the same steps, except that $(\phi \wedge \psi) \to \psi$ reduces to Axiom 9.

□

Lemma I 1. $\neg\phi, \neg\psi \vdash \neg(\phi \vee \psi)$.

Proof. By the Deduction Theorem it suffices to proof $\neg\phi \vdash \neg\psi \to \neg(\phi \vee \psi)$. By Lemma B, this equals $\neg\phi \vdash (\phi \vee \psi) \to \psi$.

1.	$\neg\phi \vdash \phi \to \psi$	Lemma D
2.	$\neg\phi$	Given
3.	$\phi \to \psi$	MP: 2,1
4.	$\psi \to \psi$	Lemma A
5.	$(\phi \to \psi) \to (\psi \to \psi) \to (\phi \vee \phi) \to \psi)$	Ax 5
6.	$(\psi \to \psi) \to (\phi \vee \phi) \to \psi)$	MP: 3,5
7.	$(\phi \vee \phi) \to \psi$	MP: 3,6

□

Lemma 1.1. *If* $\Gamma, \phi \vdash \psi$ *and* $\Gamma, \neg\phi \vdash \psi$*, then* $\Gamma \vdash \psi$

Proof. The Deduction Theorem and lemma c combines the hypothesis $\Gamma, \phi \vdash \psi$ into $\Gamma \vdash \neg\psi \to \neg\phi$ and $\Gamma, \neg\phi \vdash \psi$ into$\Gamma \vdash \neg\psi \to \neg\neg\phi$. We apply these results with two modus ponens steps to the sixth axiom: $(\neg\psi \to \neg\phi) \to (\neg\psi \to \neg\neg\phi) \to \neg\neg\phi$ to yield $\neg\neg\phi$. Combining this result with Axiom 7 and a modus ponens step gives us $\Gamma \vdash \psi$. □

Lemma 1.2. *Let the formula* A *involve only propositional variables from* $p_1, \ldots, p_n$*. For* $1 \leq i \leq n$*, suppose that* B_i *is either* p_i *or* $\neg p_i$*. Then, either*

$$B_1, \ldots, B_n \vdash A \text{ or } B_1, \ldots, B_n \vdash \neg A$$

Proof. The proof is by induction over A. We separate five distinct cases:

Case 1 A is atomic. In this base case A is simply p_i and since B_i is either p_i or $\neg p_i$ we have that $B_i \vdash A$ or $B_i \vdash \neg A$.

Case 2 $A = \neg A_1$. This gives us two possibilities, depending on the truth value of A:

1. If $\overline{\tau}(A) = 1$ then by the induction hypothesis $B_1, \ldots, B_n \vdash \neg A_1$. Combining this with our seventh axiom, $\neg\neg A \vdash A$, we get $B_1, \ldots, B_n \vdash A$.
2. If $\overline{\tau}(A) = 0$, then $B_1, \ldots, B_n \vdash A_1$. Since $A_1 = \neg A$, this implies that $B_1, \ldots, B_n \vdash \neg A$.

Case 3 A is a formula $A_1 \vee A_2$.

a. If $\overline{\tau}(A) = 1$, then the induction hypothesis states that either $B_1, \ldots, B_n \vdash A_1$ or $B_1, \ldots, B_n \vdash A_2$. If the first is the case, we can use Axiom 3 to get $B_1, \ldots, B_n \vdash A_1 \vee A_2$. In the latter case, we can use Axiom 4 to get $B_1, \ldots, B_n \vdash A$, and thus $B_1, \ldots, B_n \vdash A$.

b. If $\overline{\tau}(A) = 0$, then by the induction hypothesis we get that $B_1, \ldots, B_n \vdash \neg A_1$ and $B_1, \ldots, B_n \vdash \neg A_2$. From this Lemma I implies $B_1, \ldots, B_n \vdash \neg A$.

Case 4 A is a formula $A_1 \wedge A_2$.

1. Suppose that $\overline{\tau}(A) = 1$. By the induction hypothesis both $B_1, \ldots, B_n \vdash A_1$ and $B_1, \ldots, B_n \vdash A_2$. From this Lemma G implies $B_1, \ldots, B_n \vdash A$

2. $\overline{\tau}(A) = 0$ Then $B_1, \ldots, B_n \vdash \neg A_1$ or $B_1, \ldots, B_n \vdash \neg A_2$ or both. This together with Lemma H implies that $B_1, \ldots, B_n \vdash A$.

Case 5 A is a formula $A_1 \to A_2$. This gives us three possible options:

a. $\overline{\tau}(A_1) = 0$. Then $\overline{\tau}(A_1 \to A_2) = 1$ and by the induction hypothesis $B_1, \ldots, B_n \vdash \neg A_1$. Combine this with Lemma D to get $B_1, \ldots, B_n \vdash A_1 \to A_2$, which equals $B_1, \ldots, B_n \vdash A$.

b. $\overline{\tau}(A_2) = 1$. By the induction hypothesis $B_1, \ldots, B_n \vdash A_2$. Combining this with Lemma E we get $B_1, \ldots, B_n \vdash A_1 \to A_2$, which equals $B_1, \ldots, B_n \vdash A$.

c. $\overline{\tau}(A_1) = 1$ and $\overline{\tau}(A_2) = 0$. Then $\overline{\tau}(A_1 \to A_2) = 0$ and by the induction hypothesis $B_1, \ldots, B_n \vdash A_1$ and $B_1, \ldots, B_n \vdash \neg A_2$. Together with Lemma F this gives us $B_1, \ldots, B_n \vdash \neg(A_1 \to A_2)$, and since $A_1 \to A_2 = A$, $B_1, \ldots, B_n \vdash \neg A$.

□

Now that we have finished our bootstrapping, we can proof the completeness theorem.

Completeness Theorem 1. *If A is a tautology, $\vdash A$.*

Proof. Suppose A is a tautology containing the variables $p_1, \ldots, p_n$, and for every $B_i \in \{B_1, \ldots, B_n$, B_i is either p_i or $\neg p_i$. By Lemma 1.2 we have that $B_1, \ldots, B_n \vdash A$ or $B_1, \ldots, B_n \vdash \neg A$. A is a tautology, and so we know the first option must be the case. We also know that since A is a tautology, for any possible value assignment of our B_i, we have that $B_1, \ldots, B_n \vdash A$. We will show by induction over i that $B_1, \ldots, B_i \vdash A$, where $0 \leq i \leq n$.

For the base case $k = n$, and we know this to be true by Lemma 1.2

For the inductions step, see that $B_1, \ldots, B_k \vdash A$ follows from $B_1, \ldots, B_k$, $B_{k+1} \vdash A$ and $B_1, \ldots, B_k, \neg B_{k+1} \vdash A$ by Lemma 1.1. Since A is a tautology, it is for any truth value of B_{k+1}. When $k = 0$, we have $\vdash A$, proving the completeness theorem. □

Before moving on to sequent calculus something remains to be said about the $\mathcal{F}$-systems. Later on in this document we are going to deal with a variation on the $\mathcal{F}$-system described in this section, named *extended Frege* ($e\mathcal{F}$). This system has an additional rule which is called the *extension rule*, permitting the abbreviation of long formulas. This can be used to reduce the size of proofs, which will be discussed in section 1.5. An $e\mathcal{F}$-proof of B is a sequence of formulas $A_1, \ldots, A_n, B$ where for all i, A_i either follows from the rules described in the above system, or A_i is a formula of the form $p_i \equiv A_j$, where p_i doesn't occur in $A_1, \ldots, A_{i-1}$ nor in A_n. An application of the extension rule can be seen as a defitinion, where p_i is the name and A_j is the definition. After its declaration the definition can be used in any but the last step of the proof. This saves us from writing the full definition. If the abbreviated formula is very long, using the extension rule can significantly reduce the number of symbols in a proof.

We will see a good example of the power of the extension rule later on in the polynomial size of the $e\mathcal{F}$-proof in section 2.1.

1.3 Sequent calculus

Another popular proof system is the propositional sequent calculus (PK). It is considered a rather aesthetic system which usually allows for more intuitive proofs than for instance resolution refutation or Frege proofs.

Rather than formulas, propositional sequent calculus works with *sequents*. A sequent consists of two *cedents*: the *antecedent* and the *succedent*, which are finite sets of formulas. The two are separated by an arrow (not the be confused by the implication symbol):

$$A_1, \ldots, A_n \longrightarrow B_1, \ldots, B_m$$

where $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$ are propositional formulas. A sequent can be interpreted as a conjuction of all formulas apearing in the antecedent implying the disjunction of the formulas appearing in the succedent:

$$\bigwedge_{i=1}^{n} A_i \rightarrow \bigvee_{j=1}^{m} B_j$$

An empty antecedent always has the truth value 1 and an empty succedent always has a truth value 0.

A sequent calculus proof consists of a rooted tree. The root is called the *endsequent*, and it is the sequent that is proved by the proof. Save for leaves, every node in the tree must be a sequent that follows from a set of rules (specified below) and one or two upper sequents. At leaf nodes we find axioms, which must take on the shape of $A \longrightarrow A$, where A can be any formula.

PK has two different types of rules: *weak* inference rules and *strong* inference rules:

Weak inference rules

$$\text{Exchange left } \frac{\Gamma, A, B, \Pi \longrightarrow \Delta}{\Gamma, B, A, \Pi \longrightarrow \Delta} \qquad \text{Exchange right } \frac{\Gamma \longrightarrow \Delta, A, B, \Pi}{\Gamma \longrightarrow \Delta, B, A, \Pi}$$

$$\text{Contraction left } \frac{A, A, \Gamma \longrightarrow \Delta}{A, \Gamma \longrightarrow \Delta} \qquad \text{Contraction right } \frac{\Gamma \longrightarrow \Delta, A, A}{\Gamma \longrightarrow \Delta, A}$$

$$\text{Weakening left } \frac{\Gamma \longrightarrow \Delta}{A, \Gamma \longrightarrow \Delta} \qquad \text{Weakening right } \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, A}$$

The strong inference rules consist of the *cut rule* and several propostional rules.

The cut rule

$$\frac{\Gamma \longrightarrow \Delta, A \qquad A, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}$$

The propositional rules

$$\neg left\ \frac{\Gamma \longrightarrow \Delta}{\neg A, \Gamma \longrightarrow \Delta} \qquad \neg right\ \frac{A, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \neg A,}$$

$$\wedge left\ \frac{A, B, \Gamma \longrightarrow \Delta}{A \wedge B, \Gamma \longrightarrow \Delta} \qquad \wedge right\ \frac{\Gamma \longrightarrow \Delta, A \qquad \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \wedge B,}$$

$$\vee left\ \frac{A, \Gamma \longrightarrow \Delta \qquad B, \Gamma \longrightarrow \Delta}{A \vee B, \Gamma \longrightarrow \Delta} \qquad \vee right\ \frac{\Gamma \longrightarrow \Delta, A, B}{\Gamma \longrightarrow \Delta, A \vee B}$$

$$\rightarrow left\ \frac{\Gamma \longrightarrow \Delta, A \qquad B, \Gamma \longrightarrow \Delta}{A \rightarrow B, \Gamma \longrightarrow \Delta} \qquad \rightarrow right\ \frac{A, \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \rightarrow B}$$

If there is a PK-proof for the sequent $\Gamma \longrightarrow \Delta$, we write $PK \vdash \Gamma \longrightarrow \Delta$.
As an axample, a PK-proof for $A \wedge B \longrightarrow B \wedge A$ is given:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B \longrightarrow B}\,\text{Axiom}}{A, B \longrightarrow B}\,\text{Weakening left}}{A \wedge B \longrightarrow B}\,\wedge left \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A \longrightarrow A}\,\text{Axiom}}{B, A \longrightarrow A}\,\text{Weakening left}}{A, B \longrightarrow A}\,\text{Exchange left}}{A \wedge B \longrightarrow A}\,\wedge left}{A \wedge B \longrightarrow B \wedge A}\,\wedge right$$

Now what remains is to proof that the PK-system is both sound and complete.

Soundness Theorem 1. *PK is sound: if* $PK \vdash A \longrightarrow B$ *then* $A \models B$.

The only axioms PK allows for are $A \longrightarrow A$. Since Lemma A from the previous section shows us that $A \vdash A$ in a sound system we know that $A \models A$.

The soundness of the PK-rules is proven by showing for every PK-rule that if its upper sequents are valid, then so are the lower sequents. Since the number of rules is large and all can be proven in a similar manner, we will only demonstrate a few: the weakening right and the $\vee$ left rule.

For the weakening, right rule, let $\Gamma \longrightarrow A$ be a valid upper sequent, so $\Gamma \models A$. since $A \models A \vee B$ (we know this because it is an axiom of our sound Frege system), we can derive that $\Gamma \models A \vee B$, and so $\Gamma \longrightarrow A \vee B$ is a valid as well.

Now for the $\vee$ *left* rule, let $A, \Gamma \longrightarrow \Delta$ and $B, \Gamma \longrightarrow \Delta$ be two valid sequents. Since $A, \Gamma \models \Delta$ and $B, \Gamma \models \Delta$ it is the case that $A \vee B, \Gamma \models \Delta$, and so $A \vee B, \Gamma \longrightarrow \Delta$ is valid as well.

The proofs of other rules are analogue to the ones above.

Proving the completeness of PK requires an additional theorem:

The inversion theorem 1. *Let I be any propositional inference that is not a weakening inference. If I's lower sequent is valid, then so are all of I's upper sequents.*

The inversion theorem is in a way soundness the other way around. It is proved in a similar manner: by showing for every rule save for the weakening rules that if the lower sequent is valid, all upper sequents must be valid. Since the proof is so much alike to the soundness' proof we will not elaborate on it here.

Completeness Theorem 1. *PK is complete: if* $PK \vdash A \longrightarrow B$*, then* $A \models B$*.*

Proof. For the proof of completeness of PK, assume that $\Gamma \models \Delta$. We use induction over the number of logical connectives, m, in the sequent to proof that there is a PK-proof for $\Gamma \longrightarrow \Delta$.

If $m = 0$ Γ and Δ contain only atoms. Since $\Gamma \longrightarrow \Delta$ is valid, there must be some atom P in both Γ and Δ. $\Gamma \longrightarrow \Delta$ can then be derived from $P \longrightarrow P$ through weakening and exchange steps.

If $m > 0$, $\Gamma \longrightarrow \Delta$ contains at least one logical connective. $\Gamma \longrightarrow \Delta$ can then be derived by its upper sequents by a $\wedge-, \vee-$ or $\neg-$introduction on either the left or right side, depending on whether the connective is in Γ or in Δ, and through exchanges. Each upper sequent of these rules have at least one connective less than $\Gamma \longrightarrow \Delta$, and by the inversion theorem they are valid. This concludes the induction step and thereby our proof for the completeness of PK. □

1.4 Resolution

Proof by *resolution refutation* is a technique that is very suitable for computerized proof search. Since automated proof search usually begin with the desired formula A and works up from there, Frege systems with the modus ponens or sequent calculus with the cut rule give us infinitely many options for the previous step. To understand why resolution refutations are so suitable for automated proof search it is necessary to know how resolution refutations work.

Resolution works with *clauses*. A clause in its turn is a set of *literals*, and a literal is either a propositional variable p or its negation $\neg p$. A positive literal is an unnegated variable and a negative literal is a negated variable. A clause is considered to be a disjunction of its literals so that for example the clause $C = \{p_1, \ldots, p_1\}$ would evaluate to 1 if at least one of the variables $p_1, \ldots, p_1$ has the truth value 1. The empty clause evaluates to 0.

In a resolution refutation the aim is to show for a set of clauses that the empty set can be derived by a finite number of applications of the *resolution rule*. The resolution rule can be applied when there two clauses C and D such that $x \in C$ and $\neg x \in D$:

$$\frac{C \qquad D}{C \setminus \{x\} \cup D \setminus \{\neg x\}}$$

$C \setminus \{x\} \cup D \setminus \{\neg x\}$ is called the *resolvent* of C and D.

The resolution rule is sound, since any truth assignment that satisfies C and D would also satisfy the resolvent: if both C and D evaluate to 1, then either 1. $\overline{\tau}(x) = 1$, or 2. $\overline{\tau}(x) = 0$.

In the first case, $\overline{\tau}(\neg x) = 0$, and therefore one of the other literals in D must evaluate to 1. In this case, the union of the remaining literals of D and C would still evaluate to 1.

In the latter case, $\overline{\tau}(x) = 0$. Since C evaluates to 1, this means that for some other literal in C, q, $\overline{\tau}(q) = 1$. Then the union of the remainder of C and D would evaluate to 1.

Since the resolution rule is sound and the empty set is unsatisfiable, no number of applications of the resolution rule to a set of clauses Γ that is satisfiable under the same truth assignment could yield the empty set. This means that if we can produce the empty set from a set of clauses, these clauses are not satisfiable, and this is exactly how resolution refutations work: if we want to proof for a formula A that it is valid, we proof that the negation $\neg A$ is not satisfiable by producing the empty set from it in a number of resolution steps. As an example, a resolution refutation for $A \wedge B \rightarrow B \wedge A$is given below:

To proof $A \wedge B \rightarrow B \wedge A$, we need to show that the negation of this formula, $\neg(A \wedge B \rightarrow B \wedge A)$ yields the empty set in a finite number of resolution steps. First, we need to rewrite our negated formula to CNF:

$$\begin{gathered} \neg(A \wedge B \rightarrow B \wedge A) \\ \Rightarrow \\ A \wedge B \wedge \neg(B \wedge A) \\ \Rightarrow \\ A \wedge B \wedge (\neg B \vee \neg A) \end{gathered}$$

Now that we have rewritten our negated formula to CNF, we can use all of its clauses A, B and $\neg B \vee \neg A$ to derive the empty set:

$$\dfrac{A \qquad \dfrac{B \qquad \neg B \vee \neg A}{\neg A}}{\{\}}$$

We have already shown that the resolution rule preserves validity. We shall now continue to proof completeness for resolution.

Completeness Theorem for Resolution 1. *If Γ is an unsatisfiable set of clauses there is a resolution refutation of Γ.*

Proof. Assume that Γ is an unsatisfiable set of clauses. The proof is given by induction on m, the number of distinct variables in Γ.

The base case for $m = 0$ is trivial, since if there are no variables in Γ, Γ must be the empty set.

For the induction step, consider a variable p in Γ, and let Γ' be the set containing all clauses C in Γ which contain neither p nor $\neg p$ and all clauses that are the resolvent of the clauses C and D with respect to p. Now, p does not appear in Γ', so the number of variables in Γ' is smaller than the number of

variables in Γ. Now what remains is to show that if Γ is unsatisfiable, then so is Γ'. We do this by proving that if Γ' is satisfiable, then so is Γ.

Assume that Γ' is satisfiable. This means that there is a truth assignment τ, that satisfies Γ'. This τ would satisfy all clauses in Γ except for those containing $\neg p(p)$. Suppose that there is some clause $C_{\neg p}(C_p) \in \Gamma$ that is not satisfied by τ. Then, τ must satisfy every $C_p(C_{\neg p})$ in Γ by a literal that is *not* p. This can be seen by noting that τ satisfies the resolvent of $C_{\neg p}$ and C_p, and since $C_{\neg p}(C_p)$ is unsatisfiable, some literal in $C_p(C_{\neg p})$ must be true or the resolvent of C_p and $C_{\neg p}$ wouldn't be satisfiable. Since $C_p(C_{\neg p})$ is satisfiable independent of the truth assignment of p, we can make a new truth assignment τ', which is the same as τ except for that is assigns 0(1) to p.

By constructing τ' Γ' we have proven that if Γ' is satisfiable, so is Γ. Now by applying modus tollens to this result, we get that if Γ is not satisfiable, then neither is Γ', □

1.5 Proof complexity

Complexity concerns the inherent difficulty of problems. There are different types of complexity, and the complexity of a problem depends on the model of computation used for computing the problem. In this text, we consider the (non-)deterministic Turing machine as our main model of computation.

We will distinguish mostly between the asymptotic complexity classes P, NP and $coNP$. Remember that the class of P entails all computations that would run in polynomial time on a deterministic Turing machine. That is to say, there is a polynomial p such that if the size of this problem is relational to n, it can be solved in $p(n)$ steps or less.

The complexity class NP entails the computations that run in polynomial time on a non-deterministic Turing machine, which is the same as saying that a possible solution could be verified in polynomial time by a deterministic Turing machine. A problem is called NP-complete if every other problem in the class of NP can be reduced to this problem in polynomial time. A common example of an NP(-complete!) problem is deciding whether a given formula is an element of SAT, the set of satisfiable formulas: $SAT = \{\phi \mid \phi \text{ is satisfiable } \}$.

Another important complexity class is $coNP$. Note that $coNP$ is sometimes erroneously defined as the complement of NP. $coNP$ is the set of problems of which the complement is in NP. Formally: coNP $=\{x \mid \overline{x} \in NP\}$. Where a correct solution for a problem in NP can be quickly verified, an incorrect solution for a problem in the class of $coNP$ can be quickly refuted. As with NP-completeness, a problem is called $coNP$-complete if every problem in $coNP$ reduces to it in polynomial time. As SAT is in NP, $\overline{SAT}$ is in $coNP$. $\overline{SAT}$ is the the set of unsatisfiable formulas. The complexity class of $coNP$ is important to proof theory, since it holds $TAUT$, which is the language of tautologies. $TAUT$ is in $coNP$ since checking if $\phi \in TAUT$ is the same problem as checking whether $\neg\phi \in \overline{SAT}$. $TAUT$ is $coNP$-complete.

Although a lot of research has been done and is ongoing, the questions of whether $P = NP$ and $NP = coNP$ are still open. But although these are the

biggest open questions in complexity theory, $P \neq NP$ is widely believed to be true.

An open problem that proof theory concerns itself with the question of whether all tautologies have polynomial size proofs: is there a polynomial p such that if $\phi \in TAUT$, is there a proof Π of ϕ such that the length of Π is less than $p(\phi)$? Note that we are looking for a proof Π that is relational to some polynomial rather than the other way around. A common mistake is to ask whether there is a polynomial p such that the size of a proof Π of ϕ is relational to $p(\Pi)$. There is always such a polynomial: simply define p to be the size of the proof.

Before moving on to what we mean by the length of a proof, let's consider the implications if the answer to the above problem would be 'yes'. If there is indeed some proof of polynomial size for every tautology, a non-deterministic Turing machine could guess this proof. A deterministic Turing machine could then verify this proof in deterministic polynomial time. Since $TAUT$ is $coNP$-compete, this implies that $coNP = NP$.

There are three different methods of defining the complexity of a proof [4]:

1. The number of lines (steps) in a proof $\Pi = (\phi_1, \ldots, \phi_m)$ is m;
2. The symbol length of a proof, n, is $|\Pi| = \sum_{i=1}^{m} |\phi_i|$;
3. The depth d of a proof is the maximum AND/OR-depth of a formula in the proof.

The AND/OR depth of a formula ϕ can be calculated by rewriting ϕ over the complete basis $\{\wedge, \vee, \neg\}$ and bringing the negations onto the variables. Finding the maximum number of alterations between $\vee$ and $\wedge$ then gives us the AND/OR depth. For the remainder of this text, when we discussing 'the size of proof Π' we mean the symbol length n of Π.

Note that the Big O-notation is usually implied when considering proof complexity. This means that we leave out constants and lower factors, mentioning only the largest growing factor in the function. Formally, it is said that a function $f(n)$ belongs to the complexity class $O(g(n))$ if there exist $c, m > 0$ such that $g(n) > c \cdot f(n)$ for every $n > n'$. For example, $n^4 + 5n2 + 1000n = O(n^4)$ and $800n^{33} + 2^n = O(2^n)$.

Different proof systems can have different bounds on proof size. Resolution refutations have been shown to have proofs of exponential size for certain problems where Frege systems are of polynomial size ([6],[2]). A proof system f is called *super* if there is a polynomial p such that every tautology has an f-proof that is bounded in size by p. It is not currently known whether a super proof system exists. If it does, $NP = coNP$ for the reason stated above. There is still ongoing research on the question of whether (extended) Frege systems are super.

Another topic in proof theory complexity is whether there are *optimal* proof systems. A proof system is called *optimal* when it can p-simulate every other

proof system. We say that proof system ε p-simulates ε' of there is a polynomial p such that for every tautology ϕ and ε-proof Π of ϕ there is and ε'-proof Π' of ϕ and $|\Pi'| \leq p(|\Pi|)$.

Frege systems can p-simulate sequent calculi and resolution refutations. Resolution refutations can not p-simulate Frege systems or sequent calculi.

If Frege systems are optimal propositional logic would be just as powerful as other systems can be on propositional tautologies.

It is not known if Frege-systems can p-simulate $e\mathcal{F}$-systems. There are problems for which $e\mathcal{F}$-proofs are thought to be in a lower complexity class than $\mathcal{F}$-proofs. Examples are the Fisherman Inequality and the Odd-Town Theorem [1]. One problem that was long thought to be polynomial time solvable in $e\mathcal{F}$ but not in $\mathcal{F}$ is the propositional pigeonhole principle. However, in [2] Samuel Buss has proved that there is a polynomial size proof for the propositional pigeonhole principle, challenging the notion that $e\mathcal{F}$ is properly higher than Frege systems. This proof and its analysis will be the subject of the next section of this thesis. The $e\mathcal{F}$-proof will be shown first, along with an explication of why this type of proof would not yield a polynomial size $\mathcal{F}$-proof. This will be followed by an explanation and complexity analysis of Buss' proof.

As an example for the analysis of proof theory complexity, we shall analyse both the $e\mathcal{F}$-proof and the $\mathcal{F}$-proof given by Buss in the next section.

2 The pigeonhole principle

The pigeonhole principle states that if $n+1$ pigeons are put into n holes, at least one hole would contain two or more pigeons. Formally, it says if f is an injective function from $n+1$ to n, f is not 1-1. When a function is 1-1, this means that every element in the domain is mapped to exactly one element in the co-domain and vice versa. When considering the pigeons this notion seems intuitively true, but proving it in propositional logic is not as straightforward as might be expected.

First, we need to convert the pigeonhole principle into a propositional formula. To do so, we use the variables p_{ik} for $0 \leq i \leq n, 0 \leq k \leq (n-1)$. We assign p_{ik} to be 1 if pigeon i is in hole k (which is to say, $f(i) = k$), and we assign it to be 0 otherwise ($f(i) \neq k$).

To express that every pigeon is in some hole, we write

$$\bigwedge_{i=0}^{n} \bigvee_{k=0}^{n-1} p_{ik}.$$

The symbol $\bigwedge$ expresses a series of conjunctions, and the symbol $\bigvee$ expresses a series of disjunctions. Note that an empty string of conjuctions has a truth value of 1, and that an empty string of disjunctions has a truth value of 0.

To express that some hole contains at least two pigeons, we write

$$\bigvee_{k=0}^{n-1} \bigvee_{0 \leq i < j \leq n} (p_{ik} \wedge p_{jk})$$

Putting these together, we get the formula PHP_n:

$$\bigwedge_{i=0}^{n} \bigvee_{k=0}^{n-1} p_{ik} \rightarrow \bigvee_{k=0}^{n-1} \bigvee_{0 \leq i < j \leq n} (p_{ik} \wedge p_{jk})$$

The full formulas for $0 \leq n \leq 2$ are written out below:

$$\begin{aligned}
PHP_0 = \quad & 0 \rightarrow 0 \\
PHP_1 = \quad & p_{0,0} \wedge p_{1,0} \rightarrow p_{0,0} \wedge p_{1,0} \\
PHP_2 = \quad & (p_{0,0} \vee p_{0,1}) \wedge (p_{1,0} \vee p_{1,1}) \wedge (p_{2,0} \vee p_{2,1}) \rightarrow \Big((p_{0,0} \wedge \\
& p_{1,0}) \vee (p_{0,1} \wedge p_{1,1}) \Big) \vee \Big((p_{0,0} \wedge p_{2,0}) \vee (p_{0,1} \wedge p_{2,1}) \Big) \vee \\
& \Big((p_{1,0} \wedge p_{2,0}) \vee (p_{1,1} \wedge p_{2,1}) \Big)
\end{aligned}$$

Note that the formula PHP_n is relational to $O(n^3)$ in size: on the left hand side of the implication there is one variable for every combination of i and k. There are n i's and $n-1$ k's, so this gives us $n \cdot (n-1) = O(n^2)$ variables. On the right hand side there are variables for every combination of i, j and k. This gives us $(n-1) \cdot (n-1) \cdot n = O(n^3)$ symbols. $O(n^2) + O(n^3) = O(n^3)$.

The most obvious way to proof PHP_n would be to show that every truth assignment that satisfies the left hand side of the implication in PHP_n (that is, every assignment of the n pigeons to the $n-1$ holes) would imply the right hand side of PHP_n. This proof may be straightforward, but it grows very large: there are $(n-1)^n$ possible assignments of pigeons to holes, which leaves us with a proof that is exponential in size.

By stepping away from this intuitive but large proof and working with properties specific to 1-1 functions, it has been managed to construct a proofs of PHP_n that are polynomial in size. We shall consider the $e\mathcal{F}$-proof as given by Cook and Reckhow in [5], and explain why this method will not give a polynomial size proof in $\mathcal{F}$. Then we shall continue to explain Buss' method in [2], which gives a polynomial size $\mathcal{F}$-proof for PHP_n.

2.1 A polynomial size proof of PHP_n in $e\mathcal{F}$

In this section we will explicate the polynomial size proof in $e\mathcal{F}$ that Cook and Reckhow give in [5]. Before getting into the technical details of this proof we will first consider the underlying concepts.

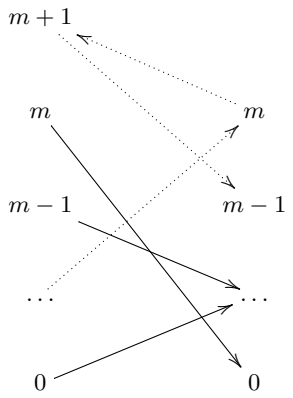
2.1.1 Outline of the proof

Cook and Reckhow proof PHP_n by assuming the counterfactual: that f *is* 1-1. Then they show that this leads to a contradiction. From this contradiction it can be concluded that the assumption must be wrong. This is called a proof by contradiction. Here follows an outline of their method:

We are going to define a sequence of functions, $f_n, \ldots, f_1$, such that f_n is f, a function from $n+1$ to n, and for all other $m < n$, f_m is a function from $m+1$ to m. f_m is defined in terms of f_{m+1} as follows:

$$f_m(i) = \begin{cases} f_{m+1}(i) & \text{if } f_{m+1}(i) < m \\ f_{m+1}(m+1) & \text{otherwise} \end{cases}$$

See the diagram below for clarification. The solid lines signify the case where $f_{m+1}(i) < m$. The dotted lines signify the other case.



$\{0, \ldots, m\}$ $\{0, \ldots, m-1\}$

Cook and Reckhow continue to proof the implication

$$f_m \text{ is 1-1} \rightarrow f_{m-1} \text{ is 1-1}$$

They do this by assuming that there *is* in fact a collission in f_{m-1}. This means that there is $i < j \in \{0, \ldots, m\}$ and a $k \in \{0, \ldots, m-1\}$ such that $f_m(i) = k$ and $f_m(j) = k$. If both arise from the case where $f_{m-1} < m$, then $f_{m+1}(i) = k$ and $f_{m+1}(j) = k$, but this contradicts our assumption that f_{m+1} is 1-1. If both arise from the second case then $f_{m+1}(i) = m$ and $f_{m+1}(j) = m$. This also contradicts our assumption. Then let's consider the last option, where one value is derived from the first case and the other is derived from the second. This means that $f_{m+1}(i) = k$, $f_{m+1}(j) = m$ and $f_{m+1}(m) = k$. This would mean that f_m collides at i and m, thus contradicting that f_{m+1} is 1-1.

Once the validity of f_m is 1-1 $\rightarrow f_{m-1}$ is 1-1 is secured, Cook and Reckhow continue to show that it follows from a number of modus ponens applications that

$$f_n \text{ is 1-1} \rightarrow f_1 \text{ is 1-1}$$

Which is the same as saying $\neg PHP_n \rightarrow \neg PHP_1$. But since $PHP_1 = p_{0,0} \wedge p_{1,0} \rightarrow p_{0,0} \wedge p_{1,0}$ clearly true, we can conclude from a reductio ad absurdum that PHP_n must be true.

This concludes the conceptual proof for PHP_n. The next section shows us how it translate into $e\mathcal{F}$.

2.1.2 Technical details of the proof

Representing the formulas in propositional logic

For translating the above proof sketch into $e\mathcal{F}$, we introduce new variables to simulate the formulas f_n to f_1. The variables are defined below (remember that p_{ik} expresses $f(i) = k$):

$$q_{ik}^n \equiv p_{ik}$$

And for $m < n$

$$q_{ik}^m \equiv q_{ik}^{m+1} \vee (q_{im}^{m+1} \wedge q_{mk}^{m+1})$$

q_{ik}^m is now defined so that it is true if $f_m(i) = k$ and false otherwise: the left hand side of the above conjunction signifies the case where $f_m(i) < m$ and $f_{m+1}(i)$ goes to k. The right hand side is the case where $f_{m+1}(i)$ goes to m.

Now we define the formula A_k to be

$$\bigwedge_{i=0}^{k} \bigvee_{j=0}^{k-1} q_{ij}^k \rightarrow \bigvee_{j=0}^{k-1} \bigvee_{0 \leq i < m \leq k} (q_{ij}^k \wedge q_{mj}^k)$$

Proving f_m is 1-1 $\rightarrow$ f_{m-1} is 1-1

By defining A_k like this it is a propositional representation of f_k. It is clear that $\neg PHP_n \rightarrow \neg A_n$, since A_n is the same as PHP_n because q_{ik}^n directly corresponds with p_{ik}. If we can show that for all $m < n$, the proof for $\neg A_{m+1} \rightarrow \neg A_m$ can be done in in polynomial size, we are done. Cook and Reckhow show that the proof can be done in $O(n^3)$ lines. Their proof is given below.

Proof. Write A_k as $L_k \rightarrow R_k$, so that

$$L_k = \bigwedge_{i=0}^{k} \bigvee_{j=0}^{k-1} q_{ij}^k$$

and

$$R_k = \bigvee_{j=0}^{k-1} \bigvee_{0 \leq i < m \leq k} (q_{ij}^k \wedge q_{mj}^k)$$

To prove $\neg A_{k+1} \rightarrow \neg A_k$, we show that the counterfactual, $\neg A_{k+1} \wedge A_k$, leads to a contradiction. The negation of A_{k+1} can be written as $L_{k+1} \wedge \neg R_{k+1}$.

So we assume L_{k+1}, $\neg R_{k+1}$ and A_k. We distinguish between two cases:

Case 1 L_k is true, so then R_k must be true as well for A_k to be true, and so there are two pigeons $i, j \leq k$ and one hole $h < k$ such that q_{ih}^k and q_{jh}^k. We can rewrite q_{ih}^k to $q_{ih}^{k+1} \vee (q_{ik}^{k+1} \wedge q_{k+1,h}^{k+1})$ and q_{jh}^k to $q_{jh}^{k+1} \vee (q_{jk}^{k+1} \wedge q_{k+1h}^{k+1})$. The conjunction of these two formulas: $(q_{ik}^{k+1} \wedge q_{jk}^{k+1}) \vee (q_{ik}^{k+1} \wedge q_{jk}^{k+1} \wedge q_{k+1h}^{k+1}) \vee (q_{ik}^{k+1} \wedge q_{k+1h}^{k+1} \wedge q_{jh}^{k+1}) \vee (q_{ik}^{k+1} \wedge q_{k+1h}^{k+1} \wedge q_{jk}^{k+1} \wedge q_{k+1h}^{k+1})$.

But every disjunction in the formula above contradicts $\neg R_{k+1}$, which states that no two pigeons are in the same hole. And so we arrive at a contradiction in only a small number of steps.

Case 2 L_k is not true. Then there is some pigeon $i \leq k$ such that for all $j < k$ we have that $\neg q_{ij}^k$. This implies that for all $h < k$ $\neg q_{ih}^{k+1}$. Since L_{k+1} is true, we know that for some hole $l < k+1$ q_{il}^{k+1}. But this hole can not be between 0 and $k-1$. This leaves us with q_{ik}^{k+1}. Because of $\neg q_{ij}^k$ and the definition of q_{ij}^k we also know that for all $h < k$, $\neg q_{ik}^{k+1} \vee \neg q_{k+1h}^{k+1}$. Since we already established that q_{ik}^{k+1} is true, this implies that for all $h < k$, $\neg q_{k+1h}^{k+1}$. Since L_{k+1} is true, there must be at least one $m \in \{0, \ldots, k\}$ such that q_{k+1m}^{k+1}, which can be thought of as there being at least one hole that pigeon $k+1$ is assigned to. The only hole not excluded by the previous negation is k, and this leaves us with q_{k+1k}^{k+1} But we now have q_{k+1k}^{k+1} and q_{ik}^{k+1} and this contradicts $\neg R_{k+1}$ which states that for no $r, s \in \{0, \ldots, k+1\}$ such that $r \neq s$ and no $t \in \{0, \ldots, k\}$ $q_{rt}^{k+1} \wedge q_{st}^{k+1}$. This can be thought of as 'there are no two pigeons are in the same hole'.

Both the above cases lead to a contradiction, proving by contradiction that $\neg A_{k+1} \rightarrow \neg A_k$. □

Now the complexity analysis of this proof: for the first case we have to show for every combination of two metaphorical pigeons and one metaphorical hole that placing these pigeons in the hole contradicts $\neg R_{k+1}$. There are $k \cdot (k-1) \cdot (k-1) < k^3 \leq n^3$ possible combinations. Deriving the contradiction can be done in a constant number of lines, so the total proof size of this case is within $O(n^3)$.

For the second case we have to show for every pigeon $i \leq k$ that if it is not in some hole, we will get a contradiction. This means that we have to show it $k \leq n$ times. Showing that $\neg q_{ij}^k$ implies $\neg q_{ij}^{k+1}$ and $\neg q_{ik}^{k+1} \vee \neg q_{k+1j}^{k+1}$ for all $j < k$ can be done in a constant number of steps, since it is simply rewriting q_{ij}^k. We can show that this implies q_{ik}^{k+1} and q_{k+1k}^{k+1} in $k+1 \leq n$ steps. This is seen by noting that we have $a \vee b \vee \ldots \vee k$ and $\neg a \wedge \neg b \wedge \ldots \wedge \neg(k-1)$ and scoring out the variables in the disjunction and the negated variables in the conjunction from front to back until only k remains. This has to be done no more than $k+1 \leq n$ times. We have to do it once for q_{ik}^{k+1} and once for q_{k+1k}^{k+1}. Deriving a contradiction with $\neg R_{k+1}$ from this is done in a constant number of steps. This case has a number of $O(n^2)$ steps in total. Adding everything up, we get a total number of lines of $O(n^3) + O(n^2) = O(n^3)$ steps.

Completing the proof

To complete the proof of PHP_n, apply the above proof of $\neg A_k \rightarrow \neg A_{k-1}$ to every formula in the sequence $f_n, \ldots, f_1$. Each formula in this proof has size $O(n^3)$. This bound is acquired by realizing that the largest formula we might

encounter is A_n, the bounds of which we calculate in a manner identical to the one we used to analyse the size of PHP_n. Multiplying these gives us a total size of $O(n^3) \cdot O(n^3) = O(n^6)$ per proof. Since we have to proof n instances to get to $\neg A_1$ from $\neg A_n$ we get $n \cdot O(n^6) = O(n^7)$. Proving A_1 takes constant time, so the total size of our proof is $O(n^7)$.

2.2 A polynomial size proof of PHP_n in $\mathcal{F}$

The $e\mathcal{F}$-proof of PHP_n given in the previous section can be transformed into a $\mathcal{F}$-proof by replacing every extension variable q_{ik}^m in the proof by the propositional formula Q_{ik}^m that it abbreviates, but this would increase the size of our formulas dramatically: q_{ik}^{m-5} is defined in terms of q_{ik}^{m-4}, which is defined in terms of q_{ik}^{m-3}, etcetera. Our definition for Q_{ik}^m is very similar to that of q_{ik}^m and it should evoke the same intuitions:

$$Q_{ik}^n = p_{ik}$$

and for all $m < n$

$$Q_{ik}^m = Q_{ik}^{m+1} \vee (Q_{im}^{m+1} \wedge Q_{mk}^{m+1})$$

So for Q_{ik}^m we have to go into recursion for Q_{ik}^{m+1}, for Q_{im}^{m+1} and for Q_{mk}^m, which are also defined recursively. Each step of the recursion triples the size of our formula. This gives us a formula size of $O(3^n)$ for Q_{00}^1 and Q_{10}^1, inflating the total proof size to $O(n^4 \cdot 3^n)$, which is superpolynomial.

This shows us the dramatic effect that using the extension rule can have on proof size. For this reason, it is thought that $e\mathcal{F}$ is more efficient than $\mathcal{F}$. PHP_n has long been an example for this believe, but in the following section we shall see the propositional pigeonhole principle does not seperate the two systems. This does not mean that $e\mathcal{F}$ and $\mathcal{F}$ are equally efficient: there are still open problems that have polynomial size $e\mathcal{F}$-proofs whereas no polynomial size $\mathcal{F}$-proofs are known.

Similar to the proof in $e\mathcal{F}$, the $\mathcal{F}$-proof is also structured around deriving a contradiction from the counterfactual, but it is done in a different manner. We continue with the outline and follow with the technical details later.

2.2.1 Outline of the proof

Again, we assume that f is 1-1. Let M^ℓ be the number of pigeons in all holes up to and including ℓ. Formally:

$$M^\ell = \left\| \left\{ i \in \{0, \ldots, n\} : \bigvee_{0 \leq k \leq \ell} p_{ik} \right\} \right\|$$

Since f is 1-1, $M^0 \leq 1$ or else there would be a collision at hole $k = 0$. From this reasoning we get the following inequalities:

$$
\begin{array}{c}
M^0 \leq 1 \\
M^1 \leq M^0 + 1 \leq 2 \\
M^2 \leq M^1 + 1 \leq 3 \\
\vdots \\
M^{n-1} \leq M^{n-2} + 1 \leq n
\end{array}
$$

However, since f is total on the domain $\{0, \ldots, n\}$, we know that $M^{n-1} = n + 1$. This is where we reach a contradiction, and from this and a reductio ad absurdum we conclude that PHP_n must be the case.

2.2.2 Technical details of the proof

To work this proof out in $\mathcal{F}$, we need to define propositional formulas that can simulate the numeric quantities and arithmetic operations above.

Defining numeric quantities in propositional logic

We write the numbers M^ℓ in binary using $(a+1)$-bit numbers, where $a = \lfloor log_2(n+1) \rfloor$. We choose this value for a, because the largest value that $^\ell$ can be is $n+1$ in the case of M^{n-1}. If a lower value needs to be expressed this can be done using leading zeroes. For example, if $n = 16$ then a would be 4. We would write 3 as 00011 and 17 as 10001.

The numbers are represented by a sequence of formulas $\hat{m}^\ell = m_a^\ell, \ldots, m_0^\ell$, where each m_i^ℓ represents whether the i-th bit is 0 or 1: 0 if m_i^ℓ is $False$ and 1 if it is $True$.

The numerical quantities $\hat{m}^\ell$ are defined by formulas $Count^\ell$ which consist of a bits so that there is one formula for every bit of every $\hat{m}^\ell$. Each $Count_i^\ell$, $0 \leq i \leq a$, has $n+1$ free variables: $x_0, \ldots, x_n$. Each of these variables x_i is going to be substituted by $f(i) \leq \ell$. The number of true variables equals M^ℓ, so we need to find a way to count them. The first approach to this is using the 'standard' method for binary addition, but it turns out that this gives us a superpolynomial size proof. Still, it is an instructive example and for this reason we will work it out below.

Addition in propositional logic

Our first try for counting the number of true free variables in $Count^\ell$ is by using formula Add_i:

$$
Add_i(\hat{y}, \hat{z}) := \begin{cases} y_0 \oplus z_0 & \text{if i} = 0 \\ y_0 \oplus z_0 \oplus Carry_i & \text{otherwise} \end{cases}
$$

$Carry_i$ will evaluate to true if the addition of previous bits carries 1.

$$
Carry_i = \bigvee_{j<i} \left(y_j \wedge z_j \wedge \bigwedge_{j<k<i} (y_k \oplus z_k) \right)
$$

$Carry_i$ is of $O(\log^2 n)$ size: j can take op to $a - 1 \approx \log n$ distinct values, and inside the big disjunct k can take up to $a - 2 \approx \log n$ values. This gives an upper boundary of $\log n \cdot \log n = \log^2 n$.

Now we let A^ℓ_{ij} be the number of x_k that are $True$ is the jth segment of size 2^i of $Count^\ell$. Formally:

$$A^\ell_{ij} := \|\{x_k : x_k = True \wedge j \cdot 2^i \leq k < (j+1) \cdot 2^i\}\|$$

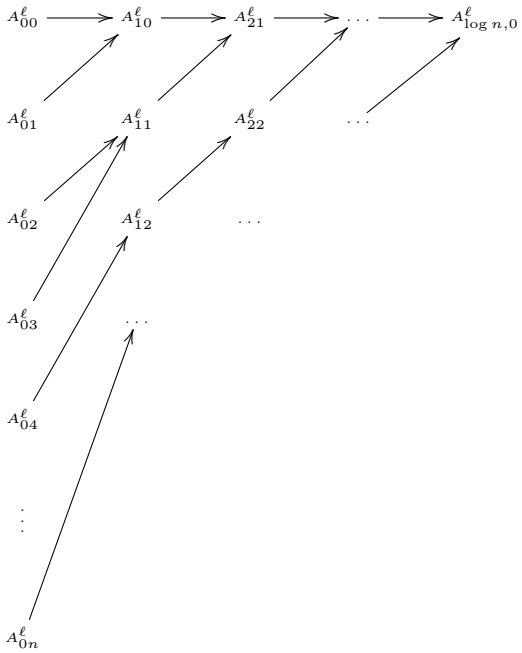
We count these values using a divide and conquer technique. For $i = 0$, this is very straightforward:

$$A^\ell_{0j} = \begin{cases} True & \text{if } x_j = \text{True} \\ False & \text{otherwise} \end{cases}$$

If $i > 0$, we calculate the values inductively:

$$A^\ell_{ij} = A^\ell_{i-1,2j} + A^\ell_{i-1,2j+1}$$

A diagram might clarify the induction steps:



We express these A^ℓ_{ij} by propositional formulas in a manner similar to the quantities M_ℓ: we use a-bit sequences $\hat{a}^\ell_{ij} = {}_a a^\ell_{ij}, \ldots, {}_0 a^{ij}$.

For $i = 0$, ${}_0 a^\ell_{ij}$ is x_j. For all $b > 0$, ${}_b a^\ell + ij$ is $False$. For $i > 0$ we let

$$a^{ij}_b = Add_b(\hat{a}^\ell_{i-1,2j}, \hat{a}^\ell_{i-1,2j+1})$$

$Count_i^\ell$ is then given by ${}_ia^\ell_{\log n,0}$: this is the sum of all true x_k in $Count^\ell$ in section 0 of size $\log n$. Since the our number consist of exactly $\log n$ bits, this equals the total sum of all true x_k in $Count^\ell$. As explained with the analysis of $Carry_i$, each ${}_ba^\ell_{ij}$ appears $O(\log^2 n)$ times in ${}_{b'}a^\ell_{i+1,\frac{j}{2}}$ for all $b' > b$. Since we need to recurse up to $i = \log n$, every x_j will appear $O(\log^2 n)^{\log n}$ times. If we want to express $O(\log^2 n)^{\log n}$ in terms of n, we get the following:

$$\begin{gathered} n^{f(n)} = O(\log^2 n)^{\log n} \\ \Rightarrow \\ \log(n^{f(n)}) = \log(O(\log^2 n)^{\log n}) \\ \Rightarrow \\ f(n) \cdot \log n = \log n \cdot \log(O(\log^2 n)) \\ \Rightarrow \\ f(n) = \log(O(\log^2 n)) \end{gathered}$$

So $O(\log^2 n)^{\log n}$ equals $n^{\log(O(\log^2 n))}$, which is superpolynomial. Since we are looking for a polynomial size proof, this will not suffice. Fortunately, there is a way to simulate the addition operations in constants time. This is done by using carry-save addition.

The idea of this is that the sum of three numbers n_0, n_1, n_2 can be expressed as the sum of two numbers m_0, m_1. For every bit b, m_{0b} is the sum of $(n_{0b}+n_{1b}+n_{2b}) \mod 2$ and m_{1b} is 1 if $n_{0b}+n_{1b}+n_{2b} > 0$ and 0 otherwise. m_{1b} expresses is 1 if there would be a carry for regular addition. (For now) we will omit further details of carry-save addition and assume that the resulting formulas are at most $k^{\log n} = n^k$. $\hat{a}^\ell_{ij}$ is then obtained by applying the original Add_b once to every bit of the m_0 and m_1. $\hat{a}^\ell_{ij}$ gives us the propositional representations of $Count^i$ and are polynomial in size.

Representing inequalities in propositional logic

Now that numeric and quantities have been defined, we need a way to represent inequalities in propositional logic before we can continue the proof. We define the following formulas:

$$\begin{gathered} EQ_i(\hat{y}, \hat{z}) = \bigwedge_{0 \le j \le i} (y_j \leftrightarrow z_j) \\ LESS_i(\hat{y}, \hat{z}) = \bigvee_{0 \le j \le i} \left(\neg y_j \wedge z_j \wedge \bigwedge_{j < k \le i} (y_k \leftrightarrow z_k) \right) \\ LEQ_i(\hat{y}, \hat{z}) = EQ_i(\hat{y}, \hat{z}) \vee LESS(\hat{y}, \hat{z}) \end{gathered}$$

The validity of EQ_i is very easy to verify: two numbers are of course equal if all of their bits are equal. Since i is at most $\log n$, EQ_i consists of at most $\log n$ conjunctions, sizing up to $O(\log n)$.

To verify $LESS_i$, see that $\hat{z}$ is larger than $\hat{y}$ if, starting with the leftmost, most significant bit and moving to the right the bits of $\hat{z}$ and $\hat{y}$ are equal at least

until there is one bit b where $z_b = True$ and $y_b = False$. $LESS_i$ can consist of up to $\log n$ disjunctions, each of which can consist of up to $\log n$ conjunctions, giving a total size of $O(\log^2 n)$.

The validity of LEQ_i follows from that of EQ_i and $LESS_i$. It's size is $O(\log n) + O(\log^2 n) = O(\log^2 n)$.

For practical purposes we write usually write $LEQ(\hat{a}, \hat{b})$, $LESS(\hat{a}, \hat{b})$ and $EQ(\hat{a}, \hat{b})$ rather writing out that the (in)equality is true for every bit i of $\hat{a}$ and $\hat{b}$.

Proving that $\neg PHP_n$ implies $M^0 \leq 1$

Now that we have all the arithmetic operations and ways to represent numeric quantities in propositions, we can begin our proof of PHP_n. As explained in the beginning of this section, we assume the counterfactual and then derive through a number of inequalities at a contradiction.

We begin by showing that

$$\neg PHP_n \rightarrow LEQ(Count^0, 1)$$

Which is the same as saying that if PHP_n is 1-1, $M^0 \leq 1$, because the way in which we defined $Count^\ell$ makes sure it equals M^ℓ.

Proof. Since we assume $\neg PHP_n$ we know that $\bigwedge_{0 \leq i < j \leq n} \neg p_{i0} \vee \neg p_{j0}$. We distinguish between two cases:

Case 1 For some i, p_{i0} is $True$. By the formula above we have that for all other j, $\neg p_{j0}$. The free variables in $Count^0$ will be assigned accordingly: $x_i = 1$ and for all other j, $x_j = 0$. When we add the number of true x_k to get $Count^0$, the summation will of course amount to 1, and so we have that $EQ(Count^0, 1)$, and so we have $LEQ(Count^0, 1)$ by the definition of LEQ.

Case 2 For all i, $\neg p_{i0}$. As a result, for all j $x_j = 0$, and so $EQ(Count^0, 0)$. Since $LEQ(0, 1)$ we can derive $LEQ(Count^0, 1)$.

$LEQ(Count^0, 1)$ derives from both possible cases, and thus we have proved that $\neg PHP_n \rightarrow LEQ(Count^0, 1)$. □

This proof is of polynomial size. For the first case we have to show that for any i, if p_{i0} we can derive $LEQ(Count^0, 1)$. This means that we have to proof the following n times: all other j are $\neg pj0$, this adds $O(n)$ variables to our proof, and the resulting summation of all x_k will be 1. The addition step is done in constant time save for the final application of Add_i, giving us a total of $O(\log^2 n)$ for this step. Proving the inequalities can also requires an $O(\log^2 n)$-length proof. Doing each of these steps n times gives us a total proof size of $O(n^2 + n \cdot \log^2 nn \cdot n \cdot \log^2 n = O(n^2)$.

The second case requires us to show that if for no i, p_i, then all the inequalities are valid, so this proof is of $O(\log^2 n)$ size.

Proving that $\neg PHP_n$ implies $M^\ell \leq M^{\ell-1} + 1$

The next step of our proof is to show that for every $m \in \{0, \ldots, n-1\}$:

$$\neg PHP_n \rightarrow LEQ(Count^m, Add(Count^{m-1}, 1))$$

This is the same as saying that if PHP_n is 1-1, then $M^\ell \leq M^{\ell-1} + 1$. The proof is similar in structure to the proof of $\neg PHP_n \rightarrow LEQ(Count^0, 1)$, so it will be reviewed only briefly.

Proof. We use the fact that by $\neg PHP_n$, no two pigeons are put in the same hole again, and again we distinguish between two cases.

In the first case, one pigeon i is placed in hole m: p_{im}. This implies that for all other pigeons j, $\neg p_{jm}$, and so only one x_k that was not true in $Count^{m-1}$, namely x_i, can be true in $Count^m$. From this we can show that $EQ(Count^m, Add(Count^{m-1}, 1))$ and so $LEQ(Count^m, Add(Count^{m-1}, 1))$.

For the second case no pigeon is placed in hole m, so for all i: $\neg p_{im}$. In this case $EQ(Count^m, Count^{m-1})$ and since obviously
$LEQ(Count^{m-1}, Add(Count^{m-1}, 1))$ we can show that $LEQ(Count^m, Add(Count^{m-1}, 1))$.

This concludes the proof for $\neg PHP_n \rightarrow LEQ(Count^m, Add(Count^{m-1}, 1))$. □

The complexity analysis is very similar to that of $\neg PHP_n \rightarrow LEQ(Count^0, 1)$ as well, and so it follows that this step of the proof is of polynomial size.

Proving that $\neg PHP_n$ implies $M^{n-1} \leq n$

Our next step is to show that from the reasoning above, it follows that

$$\neg PHP_n \rightarrow LEQ(Count^{n-1}, n)$$

Proof. To see this, we begin to proof that $\neg PHP_n \rightarrow LEQ(Count^1, 2)$:

We have that $\neg PHP_n \rightarrow LEQ(Count^1, Add(Count^0, 1))$. Since we also know that $\neg PHP_n \rightarrow LEQ(Count^0, 1)$ and we can show that $EQ(Add(1, 1), 2)$, we can derive that $LEQ(Count^1, 2)$, implying that $\neg PHP_n \rightarrow LEQ(Count^1, 2)$.

We have to repeat the above procedure n times, first to show that $\neg PHP_n \rightarrow LEQ(Count^2, 3)$ and so on, until we reach $\neg PHP_n \rightarrow LEQ(Count^{n-1}, n)$. □

Each step in the above proof consists of a number of additions and inequalities. Since all of these are of $O(\log^2 n)$ size, this total size of this step is $O(n \cdot \log^2 n)$.

Proving that $\neg PHP_n$ implies $M^{n-1} = n + 1$ We can show that the total number of pigeons in all $n-1$ holes must equal $n+1$. In propositional logic:

$$\neg PHP_n \rightarrow EQ(Count^{n-1}, n + 1)$$

We do this by showing that the counterfactual allows us to derive a contradiction.

Proof. Assume the counterfactual, so that we have $\neg PHP_n$ and also $\neg EQ(Count^{n-1}, n+1)$. Since $Count^{n-1}$ is the summation of the number of true x_k and there are only $n+1$ of these, we know that it can never exceed this number. So let us assume that $Count^{n-1}$ is less than $n+1$. In this case there must be some $i \in \{0, \ldots, n\}$ such that $\neg x_i$. From our definition of x_i it follows that for this i, $\neg \bigvee_{j=0}^{n-1} p_{ij}$, but this directly contradicts $\neg PHP_n$. Hence we can derive that $\neg PHP_n \rightarrow EQ(Count^{n-1}, n+1)$. □

The size of this proof is polynomial: for every i we have to show that $\neg x_i \rightarrow \neg\neg PHP_n$, which is done in constant time and results in an $O(n)$ size proof.

Deriving the contradiction

To derive at a contradiction, $\neg PHP_n \rightarrow LEQ(Count^{n-1}, n)$ and $\neg PHP_n \rightarrow EQ(Count^{n-1}, n+1)$ can be combined into:

$$\neg PHP_n \rightarrow LEQ(n+1, n)$$

But clearly $\neg LEQ(n+1, n)$: it is not the case that for all bits b $n+1_b$ and n_b are equal, nor is there some bit i such that $\neg n+1_i \wedge n_i \wedge \bigwedge_{i<j\leq \log n} n+1_j \leftrightarrow n_j$. We have now derived a contradiction from $\neg PHP_n$, and with a reductio ad absurdum we get PHP_n. This is exactly what we wanted to prove.

Now that we have shown that every step of the proof is of polynomial size, it follows that combining these steps also yields a proof of polynomial size, and this tells us that the polynomial pigeonhole principle does not seperate $\mathcal{F}$ from $e\mathcal{F}$.

Conclusion

We have looked at some of the basics notions of proof theory complexity. The possible implications for questions such as whether there are proof systems in which all tautologies have proofs bounded in size by some polynomial demonstrate the importance of doing research in this area: it could play a major role in determining whether $NP = coNP$. As an example of proof theory complexity we have analysed two proofs for the propositional pigeonhole principle in the two systems $\mathcal{F}$ and $e\mathcal{F}$. While the latter is generally considered to be properly higher than the first, we have shown how PHP_n, which was long thought to separate the two systems, has a polynomial size proof in both systems. For future research it would be interesting to look into other problems currently separating $e\mathcal{F}$ and $\mathcal{F}$. Perhaps it is possible to construct polynomial size $\mathcal{F}$-proofs for these problems, or, more likely, to proof that these problems can not have polynomial size $\mathcal{F}$-proofs.

References

[1] M. L. Bonet, S. R. buss, and T. Pitassi, *Are there hard examples for frege systems?*, (1995), pp. 31–56.

[2] S. R. Buss, *Polynomial size proofs of the propositional pigeonhole principle*, The Journal of Symbolic Logic, 52 (1987).

[3] S. R. Buss, *An Introduction to Proof Theory*, Elsevier North-Holland, 1998.

[4] ——, *Propositional proof complexity: An introduction*, Computational Logic, (1999), pp. 127–178.

[5] S. A. Cook and R. A. Reckhow, *The relative efficiency of propositional proof systems*, The Journal of Symbolic Logic, 44 (1979), pp. 36–50.

[6] A. Haken, *The intractability of resolution*, Theoretical Computer Science, 39 (1985), pp. 13–27.

[7] J. Harrison, *Formal proof- theory and practice*, Notices of the AMS, 55 (2008), pp. 1395–1406.