PERMUTATION QUANTIFIERS IN PROPOSITIONAL LOGIC

A Propositional Proof System with Permutation Quantifiers

By
Tim Paterson, B.Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements for the Degree of

Master of Science

Department of Computing and Software

McMaster University

MASTER OF SCIENCE(2005)
(Computing and Software)

McMaster University
Hamilton, Ontario

TITLE: A Propositional Proof System with Permutation Quantifiers

AUTHOR: Tim Paterson, B.Sc.(University of Toronto)

SUPERVISOR: Professor Michael Soltys

NUMBER OF PAGES: v, 80

Abstract

Propositional proof complexity is a field of theoretical computer science which concerns itself with the lengths of formal proofs in various propositional proof systems. Frege systems are an important class of propositional proof systems. Extended Frege augments them by allowing the introduction of new variables to abbreviate formulas. Perhaps the largest open question in propositional proof complexity is whether or not Extended Frege is significantly more powerful that Frege. Several proof systems, each introducing new rules or syntax to Frege, have been developed in an attempt to shed some light on this problem.

We introduce one such system, which we call H, which allows for the quantification of transpositions of propositional variables. We show that H is sound and complete, and that H's transposition quantifiers efficiently represent any permutation.

The most important contribution is showing that a fragment of this proof system, H_1^*, is equivalent in power to Extended Frege. This is a complicated and rather technical result, and is achieved by showing that H_1^* can efficiently prove translations of the first-order logical theory $\forall P\mathbf{LA}$, a logical theory well suited for reasoning about linear algebra and properties of graphs.

Contents

Chapter 1

Introduction

A great deal of effort in modern mathematics and computer science is dedicated to the concept of a proof. In [6], Buss differentiates between the idea of a social proof and a formal proof. A social proof, he says, is one that incorporates only the amount of mathematical rigour needed to convince the professional audience of its validity. Social proofs vary greatly in appearance depending on the subject, level of complexity, and the age in which they are published. Social proofs are therefore quite difficult to characterize. On the other hand, a formal proof is a rigourous mathematical object that can be described, analyzed, manipulated, and reasoned about in its own right. A formal proof is a series of symbols (i.e. a finite string) that, when read in the correct way, demonstrates a given property, typically of a formula or expression. In the following work, we consider proof systems for first order expressions about Linear Algebra, non-k-colourable graphs, and most importantly, proof systems for propositional statements. The study of proofs, and more specifically the minimum sizes of proofs in various proof systems is called Proof Complexity.

Perhaps the largest open question in Propositional Proof Complexity is the separation between Frege systems and Extended Frege systems. No family of tautologies has ever been exhibited which requires exponential sized Frege proofs, but which has polynomial sized Extended Frege proofs. In fact, no one has ever shown a set of tautologies which requires superpolynomial sized Frege proofs at all, so it seems we are a long way from proving this separation.

Separations of propositional proof systems have their parallel in Complexity Theory as well. For example, the existence of a proof system which could efficiently prove all tautologies is equivalent to NP equalling co-NP, and therefore showing the non-existence of such a proof system would imply that $P \neq NP$ [3].

One avenue of research that has been explored in order to show the separation of Frege and Extended Frege is the introduction of various extensions to basic propositional proof systems, such as allowing for substitution of formulas for variables, allowing for quantification of variables, or allowing for permutation of variables. In Chapter 2 we introduce several such extensions and show that they are all equivalent to Extended Frege. Since Extended Frege corresponds to the complexity class P, that means that anything that can be reasoned about in polynomial time can be proved by Extended Frege. Proving these equivalences therefore tells us that those ideas (quantification, substitution etc.) capture all of polynomial time reasoning.

It is towards this goal that the major contributions of this thesis can be found. We introduce and define a new extension to Extended Frege, in which we allow for quantifiers that assert the existence (or universality) of *permutations* of variables, rather than their values. We go on to show that a fragment of this new system is equivalent to Extended Frege, giving the perhaps surprising result that a block of either existential or universal permutation quantifiers is equivalent in expressive power to introducing new variables, renaming variables, substituting formulas, or to a single block of classical quantifiers.

A classical universal quantifier is (semantically) an abbreviation for:

$$\forall x \alpha(x) \equiv \alpha(0) \wedge \alpha(1) \tag{1.1}$$

whereas a universal permutation quantifier can be thought of as abbreviating:

$$(\forall ab)\alpha(a,b) \equiv \alpha(a,b) \wedge \alpha(b,a) \tag{1.2}$$

There are analogous existential quantifiers as well:

$$\exists x \alpha(x) \equiv \alpha(0) \vee \alpha(1) \quad \text{and} \quad (\exists ab)\alpha(a,b) \equiv \alpha(a,b) \vee \alpha(b,a) \tag{1.3}$$

Where if $\alpha(a,b)$ is some formula, then $\alpha(b,a)$ is α with b replacing a and a replacing b throughout.

We present this system, which we call H to highlight its similarity to the system G (with classical quantification) presented in [8]. We give the axioms and rules of H, and show that they form a sound and complete proof system. The results in this thesis have been submitted to the Journal of Discrete Mathematics for publication [9].

1.1 Overview

In the next section we introduce some basic concepts and definitions which, while not necessarily directly related to the rest of the material are nevertheless important for breadth and understanding.

Then, in Chapter 2 we introduce PK, a propositional proof that is equivalent to Frege, and syntactically more convenient for our purposes. Also, we identify some well known extensions to PK, and show that they are equivalent to one another.

In Chapter 3 we describe G, another PK extension which is of particular relevance. G allows for the introduction of existential and universal quantifiers of Boolean variables. We show that G is equivalent to the extensions mentioned in Chapter 2, and we extend the classical proof of completeness to include quantification.

Chapter 4 introduces a new propositional proof system H. A prototype of this system was introduced first by the author and Michael Soltys in [9], building on an idea from [13]. We show that H is sound, and complete for both quantified and unquantified formulas. We show that Extended PK can simulate H_1^*, a restricted subsystem of H. This is one direction of our main result.

The remaining chapters contain the other direction of the main result and necessary background material. We show that H_1^* can simulate Extended Frege by showing that it can efficiently prove translations of $\forall P\mathbf{LA}$. Then, we repeat results from Soltys [14] showing that $\forall P\mathbf{LA}$ can prove the soundness of the Hajós Calculus, and from Pitassi and Urquhart [10] to show that the Hajós Calculus is equivalent to Extended PK. This is a difficult, technical result, and leaves open the tantalizing question of a direct simulation.

1.2 Preliminaries

Propositional Formulas

Propositional formulas are defined inductively. The base case is a propositional variable, which can either be assigned true (1) or false (0). Then, if A and B are propositional formulas, so are $A \wedge B$, $A \vee B$, and $\neg A$. $A \wedge B$ is true if and only if both A and B are true, $A \vee B$ is true if and only if at least one of A or B is true, and $\neg A$ is true iff A is false. A *truth assignment* assigns a value to each propositional variable. Each truth assignment *satisfies* a formula iff the formula evaluates to true under the assignment.

Propositional formulas can either be valid, satisfiable, or unsatisfiable depending on whether they evaluate to true under all, some, or no truth assignments respectively. Typically we are interested in showing that a formula is valid. It is natural to think about what constitutes a formal proof of a formula's validity. Beame and Pitassi [1] point out that a propositional formula by itself could be considered a proof of its own validity. This 'proof' could be checked by checking each of its truth assignments. While possible, this is infeasible because each formula has exponentially many truth assignments. Instead we turn to the notion of a propositional proof system, as formalized in [3]. The basic idea is that a propositional proof system is a polynomial-time computable predicate P such that for each formula F

$$F \in \text{TAUT} \Leftrightarrow \exists p . P(F, p) \tag{1.4}$$

namely that there is some string p that allows us to quickly verify the validity of F.

A formula F is a *logical consequence* of a set of formulas S if every truth assignment that satisfies S also satisfies F. We write this $S \models F$. Note that any formula is a logical consequence of an unsatisfiable set of formulas.

Frege Systems

A *rule of inference* is a pair (S, B) where S is a set of formulas and B is a formula. The rule is sound if $S \models B$. S may be empty, in which case the rule is called an *axiom* or *axiom scheme*. We use the term axiom scheme to highlight the fact that any substitution instance of an axiom is valid, so that a single axiom scheme in fact corresponds to an infinite family of axioms.

A *deduction system* is a set of logical connectives and a finite set of inference rules. A *derivation* in a deduction system is a finite sequence of rule applications. If B is the result of the final rule application, then the derivation can be said to be a *proof* of B.

If B is any formula proved by a deduction system, then B must be a tautology. This can be shown by induction on proof length and the fact that $\models$ is transitive. So every deduction system is *sound*. A deduction system is *complete* if every tautology has a proof in that system. Furthermore, a deduction system is *implicationally complete* if, by adding any set of non-logical axioms Φ, there exists a valid proof of every formula C that is a logical consequence of Φ. An implicationally complete deduction system can prove any formula from an unsatisfiable set of non-logical axioms.

Frege systems were defined in [3] as implicationally complete deduction systems over a complete set of connectives. An *Extended Frege* system is a Frege system in which it is also permitted to introduce extension definitions. An extension definition is formed by taking a variable that is previously unused, and declaring it to be an abbreviation of a formula. Frege systems are all equivalent, in the sense that all Frege systems have short proofs of a set of tautologies if and only if the set has short proofs in one Frege system. Extended Frege systems are equivalent to one another in the same sense.

We will make use of these facts in order to simplify things later on. It may be that a specific formulation of a Frege system is more convenient for a given task, and we will switch between them as necessary.

Simulation and Equivalence

If A and B are two propositional proof systems, we say that B *p-simulates* A if there is a polytime function f such that if π_A is a valid A-proof, π_A, then $f(\pi_A)$ is a valid B-proof of the same formula. If A and B p-simulate one another, they are said to be *p-equivalent*.

If we can show that for every A-proof, there is a B-proof of the same formula, but we cannot give a polynomial time function from one to the other, we say that that B *weakly p-simulates* A. Unless stated otherwise, all the simulations given in this thesis are *strong* p-simulations.

Complexity Classes

There are hundreds of complexity classes[†], the most well known of which are probably P, problems which are solvable in polynomial time, and NP, problems which are verifiable in polynomial time. For our purposes, the two most important complexity classes are P/poly, and NC^1. P/poly is the class of decision problems solvable with polysize circuits. NC^1 is the class of decision problems solvable by polysize, bounded fan-in, Boolean circuits with depth $O(\log(n))$.

Frege systems, such as PK, correspond to NC^1, while Extended Frege systems, correspond to P/poly. It is strongly conjectured that these two complexity classes are not equal, but, as previously noted, it is also believed that it is a difficult task to show a separation of Frege and Extended Frege. Such a separation would be a huge achievement in propositional proof complexity.

Permutations and Symmetric Groups

Permutations are a fundamental concept in this thesis, and are also an integral component in many areas of modern algebra. For instance, in group theory, the symmetric group S_n is the group consisting of all permutations of the first n natural numbers. Symmetric groups are very important. For example, all finite groups are isomorphic to some subgroup of one of these symmetric groups, so all of finite group theory can be described in terms of permutations.

It is natural to therefore seek some connection between algebra and reasoning using permutations. We make important use of this idea by using the system **LA** as a key stepping stone in our proof that H_1^* is p-equivalent to Extended Frege. **LA** is a logical theory introduced in [12] for reasoning about Linear Algebra. We summarize it briefly in Chapter 5, and show that it translates into families of propositional tautologies.

[†] For an extensive list, see `http://qwiki.caltech.edu/wiki/Complexity_Zoo`

Chapter 2

PK

In 1936 Gerhard Gentzen introduced the propositional sequent calculus, PK. PK is known to be equivalent to any Frege system, so any results shown for one will hold for the other. We prefer PK primarily because **LA** also operates over sequents, which will simplify our task in Chapter 5.

Rather than working over formulas PK operates over *sequents*. A sequent is written as:

$$\Gamma \rightarrow \Delta \tag{2.1}$$

where Γ and Δ are sequences of propositional formulas called *cedents*. We refer to Γ and Δ respectively as the *antecedent* and the *succedent*.

A truth assignment τ satisfies a sequent if and only if it satisfies a formula in Δ or falsifies one in Γ. In this way, a sequent can be seen to represent the conjunction of all the formulas in Γ logically implying the disjunction of all the formulas in Δ. This means we can define a propositional formula $\mathcal{A}$ which is equivalent to a given sequent:

$$\mathcal{A} \overset{synt.}{=} \bigwedge \Gamma \supset \bigvee \Delta \tag{2.2}$$

Clearly, any truth assignment that satisfies $\mathcal{A}$ must satisfy the associated sequent, and vice versa.

Being able to express a sequent as a propositional formula allows us to apply various ideas from propositional logic to sequents. A sequent, S_1, is a *logical consequence* of another, S_2, if every τ that satisfies S_2 also satisfies S_1. S is *valid* if for all τ, $\tau \models S$, and S is *unsatisfiable* if for all τ, $\tau \not\models S$.

In particular, we can use this equivalence to define sequents with empty antecedents, because $\rightarrow \mathcal{A}$ is equivalent to $\Gamma \rightarrow \Delta$.

PK Proofs

The *logical axioms* of PK are $\rightarrow T$, $F \rightarrow$, or any substitution instance of $A \rightarrow A$. A PK *proof* of a sequent S is a finite sequence of sequents:

$$\pi = S_1, S_2, \ldots, S_n \tag{2.3}$$

such that $S_n = S$, and each S_i is either an axiom or follows from previous sequents by one of PK's thirteen rules of inference. There are six rules for introducing and reorganizing formulas:

Weakening

$$\frac{\Gamma \rightarrow \Delta}{\Gamma, A \rightarrow \Delta} \qquad \frac{\Gamma \rightarrow \Delta}{\Gamma \rightarrow A, \Delta} \tag{2.4}$$

Exchange

$$\frac{\Gamma_1, A, B, \Gamma_2 \rightarrow \Delta}{\Gamma_1, B, A, \Gamma_2 \rightarrow \Delta} \qquad \frac{\Gamma \rightarrow \Delta_1, A, B, \Delta_2}{\Gamma \rightarrow \Delta_1, B, A, \Delta_2} \tag{2.5}$$

Contraction

$$\frac{\Gamma_1, A, A, \Gamma_2 \rightarrow \Delta}{\Gamma_1, A, \Gamma_2 \rightarrow \Delta} \qquad \frac{\Gamma \rightarrow \Delta_1, A, A, \Delta_2}{\Gamma \rightarrow \Delta_1, A, \Delta_2} \tag{2.6}$$

There are six rules for introducing logical connectives:

$\vee$-introduction

$$\frac{\Gamma_1, A, \Gamma_2 \rightarrow \Delta \quad \Gamma_1, B, \Gamma_2 \rightarrow \Delta}{\Gamma_1, A \vee B, \Gamma_2 \rightarrow \Delta} \qquad \frac{\Gamma \rightarrow \Delta_1, A, B, \Delta_2}{\Gamma \rightarrow \Delta_1, A \vee B, \Delta_2} \tag{2.7}$$

$\wedge$-introduction

$$\frac{\Gamma_1, A, B, \Gamma_2 \rightarrow \Delta}{\Gamma_1, A \wedge B, \Gamma_2 \rightarrow \Delta} \qquad \frac{\Gamma \rightarrow \Delta_1, A, \Delta_2 \quad \Gamma \rightarrow \Delta_1, B, \Delta_2}{\Gamma \rightarrow \Delta_1, A \wedge B, \Delta_2} \tag{2.8}$$

$\neg$-introduction

$$\frac{\Gamma, A \rightarrow \Delta}{\Gamma \rightarrow \neg A, \Delta} \qquad \frac{\Gamma \rightarrow A, \Delta}{\Gamma, \neg A \rightarrow \Delta} \tag{2.9}$$

and finally, a rule for removing formulas, the **Cut** rule:

$$\frac{\Gamma, A \rightarrow \Delta \quad \Gamma \rightarrow A, \Delta}{\Gamma \rightarrow \Delta} \tag{2.10}$$

The *size* of a PK proof is the sum of the sizes of the sequents in the proof. The size of a sequent is the sum of the sizes of all the formulas in the antecedent and succedent. The size of a formula is the number of symbols (connectives and variables) it contains.

Where convenient, we may write $\alpha(B/x)$, by which we mean the formula $\alpha(x)$ with all instances of the variable x replaced by the formula B. If x is clear from the context, we may simply write $\alpha(B)$. In general, we cannot tell exactly what the size of $\alpha(B)$ is, but we know that it is bounded by $|\alpha(x)||B|$, which we will just write as $|\alpha||B|$.

Sample PK Proof

As an example, we present a PK proof of the tautology $A \equiv A$ where A is some formula. We do not have $\equiv$ as a connective, instead $A \equiv B$ abbreviates $A \supset B \wedge B \supset A$. Similarly, $A \supset B$ abbreviates $\neg A \vee B$.

$$\dfrac{\dfrac{\dfrac{A \to A}{\to \neg A, A}\,\neg - \mathit{right}}{\to \neg A \vee A}\,\vee - \mathit{left} \qquad \dfrac{\dfrac{\dfrac{A \to A}{\to \neg A, A}\,\neg - \mathit{right}}{\to A, \neg A}\,\mathit{exch.}}{\to A \vee \neg A}\,\vee - \mathit{left}}{\to (\neg A \vee A) \wedge (A \vee \neg A)}\,\wedge - \mathit{right}$$

where on the last line, the brackets have been inserted for readability.

Soundness and Completeness of PK

PK is both sound and complete. That is, if $\pi_{S_n} = S_1, S_2, \ldots, S_n$ is a PK proof, then S_n is a valid sequent, and likewise if S is any valid sequent, then a PK proof π_S exists.

Lemma 1 (Sequent Soundness Principle) *For each PK rule, the lower sequent is a logical consequence of the upper sequent(s).*

PROOF: The proof of this lemma, and of the following one, can be found in [4]. We show the case for $\wedge$-left. The other rules are shown to be sound in the same way. Recall, the $\wedge$-left rule is:

$$\frac{\Gamma_1, A, B, \Gamma_2 \to \Delta}{\Gamma_1, A \wedge B, \Gamma_2 \to \Delta} \tag{2.11}$$

Assume the top sequent is valid and let τ be any truth assignment. If τ satisfies some formula in Δ, or if τ falsifies some formula in Γ_1 or Γ_2, then τ must satisfy the bottom sequent as well. Otherwise, τ must falsify at least one of A or B, and therefore falsify $A \wedge B$, and in doing so satisfy the bottom sequent. □

Lemma 2 (Inversion Principle) *For each PK rule, except weakening, if the bottom sequent is valid then so are all upper sequents.*

PROOF: We can use the same argument as in the previous lemma, only in reverse. It is easy to see why we need an exception for the weakening rule, because any formula can be weakened out, including one that was crucial for preserving the sequent's validity. For example, this is a valid application of the weakening rule:

$$\frac{a \longrightarrow b}{a, b \longrightarrow b} \tag{2.12}$$

The bottom sequent is valid, the top one is not. □

Theorem 3 *PK is sound and complete.*

PROOF: Soundness follows from Lemma 1, by induction on the number of sequents in the proof and the fact that all the logical axioms ($A \longrightarrow A$) are valid.

To show completeness, we need to show that for any valid sequent S, a PK proof exists. We use induction on the number of connectives ($\vee$, $\wedge$, and $\neg$) in S. Any valid sequent with no connectives is simply two lists of atoms, and it must have one atom, a, which appears in both the antecedent and succedent (because if not, any assignment which assigns True to all atoms on the left, and False to all atoms on the right will falsify the sequent, contradicting the assumption of validity). We can use a series of weakenings and exchanges to obtain the axiom $a \longrightarrow a$.

For the induction step, we choose a formula containing a connective, and apply the appropriate connective introduction rule in reverse. The Inversion Principle guarantees that this new sequent, which has one fewer connective, is valid. □

It is interesting to note that we do not use the Cut rule in this proof of completeness. This means that the Cut rule is technically unnecessary in any proof, although it often proves to be very convenient. For example, we use it often in the proofs of equivalence below. Whether or not the Cut rule can provide a significant shortening of proofs remains an open question.

A proof that does not use the Cut rule is said to exhibit the *Subformula Property.* In such a proof, any formula that appears on any line of the proof must appear as a subformula of the conclusion.

2.1 PK Extensions

In addition to its basic rules PK allows new rules, or extensions, to be introduced easily and naturally. Several of these extensions are discussed in [8]. Since PK is complete, adding these extensions do not allow any new sequents to be proven, but it is theorized that they may significantly shorten the lengths of some PK proofs (e.g.

that they may allow for polynomial sized proofs of families of tautologies which require exponential sized proofs in basic PK). This remains an open question. However, it can be shown that the extensions polynomially simulate one another.

Renaming PK

Renaming PK adds a rule which renames all free instances of some variables:

$$\frac{\Gamma(x_1,\ldots,x_n) \to \Delta(x_1,\ldots,x_n)}{\Gamma(y_1,\ldots,y_n) \to \Delta(y_1,\ldots,y_n)} \tag{2.13}$$

Renaming PK gets its real power when the renaming rule is used to rename a variable to another existing variable. Essentially, this takes two different variables and equates them. It is important to note that the y_i are meta-variables, and that they could actually represent some or all of x_i.

A related system, Permutation PK, adds the restriction that the renaming be bijective. This means that the y_i must be a permutation of the x_i. It is not known whether Permutation PK is equivalent to PK, equivalent to Extended PK, or sits somewhere between. Determining which of these cases is true is an important open question in propositional proof complexity.

True-False PK

True-False, or TF PK, adds two new rules which replace variables with either 0 or 1 throughout the sequent. These new rules are:

$$\frac{\Gamma(x_1,\ldots,x_n) \to \Delta(x_1,\ldots,x_n)}{\Gamma(c_1,\ldots,c_n) \to \Delta(c_1,\ldots,c_n)} \tag{2.14}$$

where for $i = 1 \ldots n$, $c_i \in \{0, 1, x_i\}$

Substitution PK

Substitution PK goes further than either Renaming or TF PK by allowing all instances of a variable to be replaced by an arbitrary formula A. It does so with a single new rule:

$$\frac{\Gamma(x_1,\ldots,x_n) \to \Delta(x_1,\ldots,x_n)}{\Gamma(A_1,\ldots,A_n) \to \Delta(A_1,\ldots,A_n)} \tag{2.15}$$

Extended PK

Extended PK allows new variables to be introduced and declared to be equivalent to any formula. We are free to choose any variable that has not occurred up to this point in the proof, although these new variables cannot appear in the sequent we are attempting to prove.

$$\to a \equiv A$$

This rule is usually used where A is a subformula of some sequent we are working with in the proof, and a is used to shorten that sequent.

2.2 p-Equivalence of PK Extensions

It is not known that any of the PK extensions significantly shorten PK proofs, but we can show that they are all p-equivalent to one another[1]. Our task is made simpler in this case by the fact that each of the PK extensions contain the basic PK rules. This means that many of the steps in one proof can be carried out identically in the simulating proof system. We need only show that the special rules introduced in each extension can be simulated efficiently in the others. Since Frege and PK are equivalent systems, that means that Substitution Frege and Substitution PK are equivalent, as are Extended Frege and Extended PK etc. For convenience, we use them interchangably where it will not cause problems.

We show the equivalence of the four extensions to PK by showing that Renaming PK p-simulates TF PK, that TF PK p-simulates Substitution PK, that Substitution PK p-simulates Extended PK, and that Extended PK p-simulates Renaming PK.

Lemma 4 *Renaming PK p-simulates TF PK.*

[1] Except for Permutation PK

PROOF: This proof is due to Samuel Buss (see [2]). We also use the fact that tautologies without variables (that is, containing only connectives, and 0 and 1) have short PK proofs (see Lemma 28 in the Appendix).

Assume first that π_{TF} is a TF PK proof of a sequent $S(x_1, \ldots, x_n)$. We must show that we can efficiently construct a Renaming PK proof π_R of the same sequent. We know that there are short PK proofs of:

$$\rightarrow \mathcal{A}(1, \ldots, 1) \quad \text{and} \quad \rightarrow \mathcal{A}(0, \ldots, 0) \tag{2.16}$$

where $\mathcal{A}$ is the formula equivalent to $S(2.2)$. Define Z to be the following propositional formula:

$$Z = (x_1 \vee x_2 \vee \ldots \vee x_n) \wedge \neg(x_1 \wedge x_2 \wedge \ldots \wedge x_n) \tag{2.17}$$

which asserts that the variables x_i are not all true and not all false.

To build π_R, we first build a proof of the sequent $Z \rightarrow \mathcal{A}(\vec{x})$. Begin by taking each line in π_{TF}:

$$\rightarrow \alpha(\vec{x}) \tag{2.18}$$

and replacing it with:

$$Z \rightarrow \alpha(\vec{x}) \tag{2.19}$$

We then need to take steps to recreate a valid proof. If α is derived in π_{TF} by a basic rule of PK then we can merely weaken to introduce Z. If, however, α is the result of introducing 1 in place of some variable:

$$\frac{\rightarrow \alpha(x_1, \ldots, x_i, \ldots x_n)}{\rightarrow \alpha(x_1, \ldots, 1, \ldots x_n)} \tag{2.20}$$

then we have some more work to do. Assume that we have a proof of the top sequent. Then, from that, we derive the following $n - 1$ sequents:

$$\begin{array}{rl} Z(x_i/x_1) \rightarrow & \alpha(x_1,\ldots,x_1,\ldots,x_n) \\ Z(x_i/x_2) \rightarrow & \alpha(x_1,\ldots,x_2,\ldots,x_n) \\ \vdots & \vdots \\ Z(x_i/x_n) \rightarrow & \alpha(x_1,\ldots,x_n,\ldots,x_n) \end{array} \tag{2.21}$$

where $Z(x_i/x_j)$ denotes Z with all instances of x_i replaced with x_j. Lemma 27 shows that polynomial sized proofs exist for sequents of the form $x_i, \alpha(x_i) \rightarrow \alpha(1)$. Therefore, derive the following $n-1$ sequents:

$$\begin{array}{rcl} x_1, \alpha(x_1,\ldots,x_1,\ldots,x_n) & \rightarrow & \alpha(x_1,\ldots,1,\ldots,x_n) \\ x_2, \alpha(x_1,\ldots,x_2,\ldots,x_n) & \rightarrow & \alpha(x_1,\ldots,1,\ldots,x_n) \\ \vdots & \rightarrow & \vdots \\ x_n, \alpha(x_1,\ldots,x_n,\ldots,x_n) & \rightarrow & \alpha(x_1,\ldots,1,\ldots,x_n) \end{array} \tag{2.22}$$

From these groups of sequents, we can derive $Z^{-i} \rightarrow \alpha(1)$ where Z^{-i} is Z but with the variable x_i removed:

$$Z^{-i} \equiv (x_1 \vee \ldots x_{i-1} \vee x_{i+1} \vee \ldots x_n) \wedge \neg(x_1 \wedge \ldots x_{i-1} \wedge x_{x+1} \wedge \ldots x_n) \tag{2.23}$$

because for $i \neq j$, $Z(x_i/x_j) \equiv Z^{-i}$ (semantically, though not syntactically). We can also produce a short proof of $Z, \neg Z^{-i} \rightarrow \alpha(x_1,\ldots,1,\ldots x_n)$ (see [2]). We can then use this and $Z^{-i} \rightarrow \alpha(1)$ to derive a valid proof of the sequent:

$$Z \rightarrow \alpha(x_1,\ldots,1,\ldots,x_n) \tag{2.24}$$

The case for 0-introduction is handled similarly.

After this is complete, we have a valid proof of the sequent:

$$Z \rightarrow \mathcal{A} \tag{2.25}$$

Lemma 27 shows that short proofs exist for the following two sequents:

$$(x_1 \wedge x_2 \wedge \ldots \wedge x_n), \mathcal{A}(1, 1, \ldots, 1) \to \mathcal{A}(\bar{x}) \tag{2.26}$$

and

$$\neg(x_1 \vee x_2 \vee \ldots \vee x_n), \mathcal{A}(0, 0, \ldots, 0) \to \mathcal{A}(\bar{x}) \tag{2.27}$$

Then from these we use $\wedge$ introduction to obtain:

$$\neg Z, \mathcal{A}(1, 1, \ldots, 1), \mathcal{A}(0, 0, \ldots, 0) \to \mathcal{A}(\bar{x}) \tag{2.28}$$

We use (2.16) to cut out the two variable-free tautologies on the left, and use (2.25) to cut Z and get a proof of $\to \mathcal{A}$. We can trivially obtain a proof of $\Gamma \to \Delta$ from $\to \mathcal{A}$. □

Lemma 5 *TF PK p-simulates Substitution PK.*

Proof: This can also be found in [2]. Given a Substitution PK proof π_{Sub}, we can build an equivalent TF PK proof as follows. All applications of the basic rules remain unchanged in π_{TF}, and whenever we substitute a variable x with a formula B, we use the following construction.

$$\dfrac{\dfrac{\dfrac{\dfrac{\to \alpha(x)}{\to \alpha(1)}\,1-\mathit{replac.}}{B \to \alpha(1)}\,\mathit{weak.}}{B \to \alpha(1), \alpha(B)}\,\mathit{weak.} \qquad \begin{array}{c}\vdots\,\pi_1\\ B, \alpha(1) \to \alpha(B)\end{array}}{B \to \alpha(B)}\,\mathit{cut}$$

And then separately derive:

$$\dfrac{\dfrac{\dfrac{\dfrac{\to \alpha(x)}{\to \alpha(0)}\,1-\mathit{replac.}}{\neg B \to \alpha(0)}\,\mathit{weak.}}{\neg B \to \alpha(0), \alpha(B)}\,\mathit{weak.} \qquad \begin{array}{c}\vdots\,\pi_2\\ \neg B, \alpha(0) \to \alpha(B)\end{array}}{\neg B \to \alpha(B)}\,\mathit{cut}$$

which we then combine with $\vee$-left:

$$\dfrac{B \rightarrow \alpha(B) \quad \neg B \rightarrow \alpha(B)}{B \vee \neg B \rightarrow \alpha(B)} \vee - left$$

We can easily derive $\rightarrow B \vee \neg B$ and cut it out.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{B \rightarrow B}{\rightarrow \neg B, B} \neg - right}{\rightarrow B, \neg B} exch.}{\rightarrow B \vee \neg B} \vee - right}{\rightarrow \alpha(B), B \vee \neg B} weak.}{\rightarrow B \vee \neg B, \alpha(B)} exch.$$

to get:

$$\rightarrow \alpha(B) \tag{2.29}$$

This is clearly polynomial in size, provided π_1 and π_2 have polynomial sized proofs. This is shown in Lemma 27 in the Appendix. □

Lemma 6 *Substitution PK p-simulates Extended PK.*

Proof: This is given in [8]. We repeat the proof here. Note that we can assume that the extensions occur in the first lines of any Extended PK proof, because of the restriction that the variables introduced by the extensions cannot appear beforehand. This allows us to 'push' them up without altering the proof.

Let π_E be some Extended PK proof of $S = \Gamma \rightarrow \Delta$. Assume that $q_1 \equiv \Psi_1, \ldots, q_r \equiv \Psi_r$ are all the extension definitions in π_E and that they are all introduced in the first r steps of the proof

Then consider the sequent:

$$q_r \equiv \Psi_r, q_{r-1} \equiv \Psi_{r-1}, \ldots q_1 \equiv \Psi_1, \Gamma \rightarrow \Delta \tag{2.30}$$

which is valid, because the soundness of PK's rules means that $\Gamma \longrightarrow \Delta$ is a logical consequence of the r extension definitions. Lemma 4.4.10 in [8] shows that a proof of (2.30) exists with size $O(m^2)$ (where m is the size of (2.30)). Therefore, we apply the substitution rule to derive:

$$\dfrac{\begin{array}{c}\vdots\,\pi\\ q_r \equiv \Psi_r, q_{r-1} \equiv \Psi_{r-1}, \dots q_1 \equiv \Psi_1, \Gamma \longrightarrow \Delta\end{array}}{\Psi_r \equiv \Psi_r, q_{r-1} \equiv \Psi_{r-1}, \dots q_1 \equiv \Psi_1, \Gamma \longrightarrow \Delta}$$

Next, separately prove the sequent $\longrightarrow \Psi_r \equiv \Psi_r$ and cut out the first equivalence. After r such steps, we have a Substitution PK proof of $\Gamma \longrightarrow \Delta$. □

Lemma 7 *Extended PK p-simulates Renaming PK*

Proof: This is shown by demonstrating that Extended PK can simulate Substitution PK (of which Renaming is a special case). The details of the proof are rather technical and unenlightening, so we merely outline a high-level argument. The full proof can be found in [8].

This follows from two ideas. Firstly, if a system can be proven to be sound in polynomial time, then Extended PK can simulate it, and secondly, Dowd's proof that the soundness of the Substitution rule can be proven in polynomial time [5].

We prove the soundness of Substitution PK as follows. Given a well-formed Substitution PK proof of a sequent S (a proof can be verified to be well-formed efficiently by checking that each line follows from the previous lines by a rule), assume that some truth assignment exists that falsifies the end sequent. Then we can construct a truth assignment that falsifies an axiom (which is a contradiction).

For any rule with a single top sequent, except Substitution, the Inversion Principle guarantees that the truth assignment that falsifies the bottom sequent will falsify the top sequent.

For $\wedge$-right, if $\tau \not\models \Gamma \longrightarrow A \wedge B, \Delta$ then we know that either $\tau \not\models A$ or $\tau \not\models B$ (or both), and therefore that τ falsifies at least one of the two top sequents. Choose the branch that is falsified, and continue. Similarly, if $\tau \not\models \Gamma, A \vee B \longrightarrow \Delta$ then τ must either satisfy A or B or both, and so falsify at least one of the top sequents. In either case, we choose the branch that is falsified and continue, without altering τ.

Lastly, for the Substitution rule, if the formula Θ replaced a variable a in the original proof, then to construct a falsifying assignment τ' for the top sequent, we let

$$\tau'(x) = \begin{cases} \tau(x) & \text{if } x \neq a \\ \tau(\Theta) & \text{if } x = a \end{cases} \tag{2.31}$$

where $\tau(\Theta)$ is the truth value of Θ under τ. Once we reach the top of the proof, we will have a truth assignment that falsifies the conclusion and also falsifies an axiom. Such a truth assignment cannot exist and so the conclusion of the proof must be valid.

Since this is a polynomial time process, Extended PK must be able to simulate Substitution PK. For a direct simulation see [8]. □

This notion of an indirect simulation by using one system to certify the correctness of a proof in another system is an important one. It is often quite difficult to show simulations directly, but by reasoning more abstractly about the capabilities of a given proof system, we may be able to provide an indirect simulation. We will use this concept later with LA, H, and the Hajós Calculus.

We are now able to conclude:

Theorem 8 *The four extensions to PK, namely Renaming, Substitution, TF, and Extended PK are p-equivalent.*

Proof: Follows immediately from the previous four lemmas. □

Chapter 3

G: PK with variable quantification

Krajíček [8] defines a different sort of extension to PK in the form of quantifiers. There are two quantifiers, each with a left and right introduction rule. This new system, G, is generally used to prove sequents which are quantifier-free; the quantifiers are used in an attempt to shorten proofs. Whether or not G can actually produce significantly smaller proofs remains an open question.

Syntax and Semantics

$\forall$ introduction

$$\frac{\alpha(B), \Gamma \to \Delta}{\forall x \alpha(x), \Gamma \to \Delta} \quad \frac{\Gamma \to \Delta, \alpha(p)}{\Gamma \to \Delta, \forall x \alpha(x)} \tag{3.1}$$

$\exists$ introduction

$$\frac{\alpha(p), \Gamma \to \Delta}{\exists x \alpha(x), \Gamma \to \Delta} \quad \frac{\Gamma \to \Delta, \alpha(B)}{\Gamma \to \Delta, \exists x \alpha(x)} \tag{3.2}$$

where B is any formula, and under the restriction that the atom p does not occur in the bottom sequent for $\forall$-left and $\exists$-right.

The need for the restriction on the $\forall$ right and $\exists$ left case can be illustrated with a pair of counterexamples.

$$\frac{p, \neg p \rightarrow a \vee \neg a}{\exists x x, \neg p \rightarrow a \vee \neg a} \qquad \frac{a \vee \neg a \rightarrow p, \neg p}{a \vee \neg a \rightarrow \forall x x, \neg p} \tag{3.3}$$

In each case, the upper sequent is valid, but a truth assignment exists which does not satisfy the lower sequent.

Sequents with quantifiers behave exactly like quantifier-free sequents, with quantifiers being interpreted in the expected manner. Explicitly, $\exists x \alpha(x)$ is true if some value of x satisfies $\alpha(x)$, and $\forall x \alpha(x)$ is true if all possible values of x satisfy $\alpha(x)$. This allows us to note that each quantifier has a semantically equivalent formula:

$$\forall x \alpha(x) \equiv \alpha(0) \wedge \alpha(1) \qquad \exists x \alpha(x) \equiv \alpha(0) \vee \alpha(1)$$

The true value of quantification becomes apparent when we consider the case in which α contains additional quantifiers. Since the equivalent formulas are more than twice as long as the quantified formulas, nested quantification provides an exponential decrease in formula (and therefore sequent) length.

Soundness and Completeness of G

Lemma 9 *G is sound and complete.*

PROOF: The soundness of most of G's rules was shown in Lemma 3. We need only show the soundness of the four new rules, which we do in the same way as before. Take the $\forall$-left rule:

$$\frac{\alpha(B), \Gamma \rightarrow \Delta}{\forall x \alpha(x), \Gamma \rightarrow \Delta} \tag{3.4}$$

Assume the top sequent is valid, and let τ be any truth assignment. If τ falsifies any formula in Γ or satisfies any formula in Δ, then τ satisfies the bottom sequent also. Otherwise, τ must, by the validity of the top sequent, falsify $\alpha(B)$, in which case τ also falsifies $\forall x \alpha(x)$ and so satisfies the bottom sequent. Similar arguments show the soundness of the other three new rules.

The completeness of G follows from the completeness of PK, because any correct PK-proof is a G-proof. A more interesting question is whether G can prove all sequents containing quantifiers. □

Extended Completeness of G

By extended completeness, we mean that G is complete for valid sequents containing variable quantifiers. If A is a formula, let the number of quantifiers in A be denoted by $|A|_Q$, and similarly for sequents.

Theorem 10 *If S is a valid sequent, possibly containing variable quantifiers, then G can prove S.*

PROOF: Let S be a valid sequent (possibly with formulas containing quantifiers). We show that a proof of S must exist by induction on the maximum number of quantifiers in any formula in S.

We show the cases for connectives and quantifiers occurring on the left. The right-hand cases proceed analogously.

The base case is when the sequent is quantifier free. That is, $|S|_Q = \max_{\alpha \in S} |\alpha|_Q = 0$ In this case, the sequent is a classical tautology, and a proof of S exists by the completeness of PK.

Then, for the induction step, let $\max_{\alpha \in S} |\alpha|_Q = k+1$. Then, for each α_i in S in turn, if $|\alpha_i|_Q = k+1$ perform the following steps.

If the outermost connective in α_i is not a quantifier, apply the appropriate connective introduction rule in reverse. This produces one sequent (for $\neg$-left or $\wedge$-left) which will have at most $k+1$ quantifiers in each formula. For $\vee - left$, we obtain two sequents as follows:

$$\frac{\Gamma, \beta \rightarrow \Delta \quad \Gamma, \gamma \rightarrow \Delta}{\Gamma, \beta \vee \gamma \rightarrow \Delta}$$

where $\alpha = \beta \vee \gamma$. Continue working with both sequents until the outermost connective in the current formula is a quantifier.

Once there is a quantifier as the outermost connective in the current formula, the sequent must look like this:

$$\Gamma, \exists x \alpha(x) \rightarrow \Delta \quad \text{or} \quad \Gamma, \forall x \alpha(x) \rightarrow \Delta \tag{3.5}$$

For the $\exists$-left case, we proceed as follows:

$$\dfrac{\begin{array}{c}\pi\\ \vdots\\ \Gamma, \alpha(a) \rightarrow \Delta\end{array}}{\Gamma, \exists x \alpha(x) \rightarrow \Delta}\ \exists - left$$

Where we choose a to be some variable that does not appear in $\Gamma, \alpha(x)$, or Δ.

Similarly, for the $\forall$-left case:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\pi\\ \vdots\\ \Gamma, \alpha(a), \alpha(\neg a) \rightarrow \Delta\end{array}}{\Gamma, \alpha(a), \forall x \alpha(x) \rightarrow \Delta}\ \forall - left}{\Gamma, \forall x \alpha(x), \forall x \alpha(x) \rightarrow \Delta}\ \forall - left}{\Gamma, \forall x \alpha(x) \rightarrow \Delta}\ contr.$$

where here we have a free choice of a.

We repeat this for each other formula in Γ and Δ, removing a quantifier from each formula which had $k+1$ quantifiers to begin with. After this process, each formula has at most k quantifiers, and a proof exists by the induction hypothesis. □

G_i and G_i^*

Krajíček defines a special subset of G based on the number of alternations of quantifiers when each formula is in prenex normal form. A formula is in prenex normal form if it is written:

$$Q_1 x_1 Q_2 x_2 ... Q_n x_n \alpha(x_1, x_2, ...x_n) \tag{3.6}$$

where each Q_i is either $\forall$ or $\exists$, and α is quantifier-free.

G_i is the set of all sequents where all the formulas in both cedents are in prenex normal form, and the quantifiers are grouped into at most i blocks starting with an existential quantifier ($\exists$). Therefore, G_0 is basic PK, G_1 allows sequents to contain formulas of the form $\exists x_1 \exists x_2 \dots \exists x_n \alpha(x_1, x_2, \dots, x_n)$, G_2 allows:

$$\exists x_1 \exists x_2 \dots \exists x_i \forall x_{i+1} \forall x_{i+2} \dots \forall x_n \alpha(x_1, x_2, \dots x_n)$$

and so on.

G_i^* is G_i with the added restriction that the proofs be treelike. A treelike proof is one in which each sequent appears at most once as the upper sequent in a rule. That means that if we wish to reuse a sequent in a treelike proof, we must rederive a separate copy of it and, in doing so, increase the size of the proof.

3.1 G_1^* and Extended PK are p-equivalent

We show that G_1^* can p-simulate the substitution rule, and therefore can p-simulate Substitution PK. Then, by Lemma 6 we obtain that G_1^* p-simulates Extended PK.

Lemma 11 *G_1^* p-simulates Substitution PK.*

Proof: This proof can be found in [8]. Given a Substitution PK proof π_S, we convert it to a G_1^* proof π_G as follows. Given an instance of the substitution rule:

$$\frac{\Gamma(a) \to \Delta(a)}{\Gamma(\Theta) \to \Delta(\Theta)} \tag{3.7}$$

we replace it as follows:

$$\begin{array}{cl}
\Gamma(a_1,\ldots,a_n) \to \Delta(a_1,\ldots,a_n) & \\ \hline\hline
(\bigwedge\Gamma)(a_1,\ldots,a_n) \to (\bigvee\Delta)(a_1,\ldots,a_n) & \wedge - left, \vee - right \\ \hline\hline
\to (\neg\bigwedge\Gamma \vee \bigvee\Delta)(a_1,\ldots,a_n) & \neg - right, \vee - right \\ \hline\hline
\to \forall x_1\ldots\forall x_n(\neg\bigwedge\Gamma \vee \bigvee\Delta)(x_1,\ldots,x_n) & \forall - right
\end{array}$$

Then, from:

$$\begin{array}{cl}
(\neg\bigwedge\Gamma \vee \bigvee\Delta)(\Theta_1,\ldots,\Theta_n) \to (\neg\bigwedge\Gamma \vee \bigvee\Delta)(\Theta_1,\ldots,\Theta_n) & \\ \hline\hline
\forall x_1\ldots\forall x_n(\neg\bigwedge\Gamma \vee \bigvee\Delta)(x_1,\ldots,x_n) \to (\neg\bigwedge\Gamma \vee \bigvee\Delta)(\Theta_1,\ldots,\Theta_n) & \forall - left
\end{array}$$

Then weaken and cut to obtain:

$$\rightarrow (\neg \bigwedge \Gamma \vee \bigvee \Delta)(\Theta_1, \ldots, \Theta_n) \tag{3.8}$$

And there is a short proof of:

$$\Gamma(\Theta_1, \ldots, \Theta_n), (\neg \bigwedge \Gamma \vee \bigvee \Delta)(\Theta_1, \ldots, \Theta_n) \rightarrow \Delta(\Theta_1, \ldots, \Theta_n) \tag{3.9}$$

from which we can cut to obtain:

$$\Gamma(\Theta_1 \ldots \Theta_n) \rightarrow \Delta(\Theta_1 \ldots \Theta_n)$$

as required (and with a polynomial increase in size).

Observe that we have no alternative but to put the whole sequent into a single formula on the right, because of the restriction that the variable which gets captured by the quantifier cannot appear in the lower sequent in the $\forall - \textit{right}$ rule. □

To show that Extended PK p-simulates G_1^*, we show a method to efficiently transform a G_1^* proof (π_G), into an Extended PK proof (π_E). The basic idea is that variables are introduced on the left to 'witness' values for quantified variables on the right, in this fashion:

$$q_1 \equiv \phi_1, \ldots, q_n \equiv \phi_n, \Gamma \rightarrow \Delta$$

where the q_i are the quantified variables on the right of the arrow, and the ϕ_i are Boolean formulas which specify values for q_i that make the sequent valid.

Any formula in a G_1^* proof can be efficiently converted to a formula in prenex form. All $\forall$ quantifiers in any such formula must be nested inside an odd number of $\neg$ connectives (by the definition of G_1^*), and so will become $\exists$ quantifiers when the formula is in prenex form.

We can convert a formula to prenex normal form by giving each quantified variable a unique name, and then pushing quantifiers 'out' towards the front of the formula (using deMorgan's Laws, etc). For example:

$$\neg \forall x A(x, \vec{y}) \quad \text{becomes} \quad \exists x \neg A(x, \vec{y})$$

$$(A(\vec{y}) \vee \forall x B(\vec{y}, x)) \quad \text{becomes} \quad \forall x (A(\vec{y}) \vee B(\vec{y}, x))$$

and vice versa for $\exists$ quantifiers (which are guaranteed to be nested inside an even number of $\neg$ connectives). Since we can efficiently perform this transformation, we can assume that formulas appear in prenex form in the upcoming proof.

Lemma 12 *Extended PK p-simulates G_1^*.*

PROOF: Let π_G be a G_1^* proof of the classical (i.e. quantifier free) sequent $\Gamma \longrightarrow \Delta$. We construct an extended PK proof, π_E, of the same sequent, with only a polynomial increase in proof size.

Note: Since π_G is a G_1^* proof, we know that all the formulas are in Σ_1, and that therefore none of the $\forall$ introduction rules can appear in π_G (because the resulting formula would be in Π_2)[1]. For the same reason, the $\neg$ rule cannot be applied to any quantified formulas. Therefore, the only ways that a $\forall$ symbol could appear in π_G is in an axiom, or through weakening.

For convenience, we assume that we have two lists of variables $\mathbf{p}_1, \mathbf{p}_2, \ldots$, which we use for new free variables (on the left), and $\mathbf{q}_1, \mathbf{q}_2, \ldots$ which we use for new 'witness' variables (on the right). Whenever we need to introduce a new one of either type, we take the next unused variable in sequence. This technicality helps us when we come to the cut rule. Any variables introduced during the following translation between π_G and π_E are meta-variables which stand for one of the $\mathbf{p}_i$ or $\mathbf{q}_i$.

First, we show how the axioms are translated. Given an axiom, $A \longrightarrow A$, if A is quantifier-free there is nothing to change, and we use the same axiom in π_E. For axioms with quantifiers, we convert to prenex normal form, which guarantees that the axiom must be of the form $\exists \vec{x} B(\vec{x}, \vec{a}) \longrightarrow \exists \vec{x} B(\vec{x}, \vec{a})$ where $\vec{x}$ and $\vec{a}$ are respectively the bound and free variables in B. To replace the axiom, we then construct the following sequent in π_E:

$$q_1 \equiv p_1, \ldots q_n \equiv p_n, B(\vec{p}, \vec{a}) \longrightarrow B(\vec{q}, \vec{a}) \tag{3.10}$$

We can show how to construct sequents like (3.10) with polynomial size Extended PK proofs by induction on the structure of A. The inductive hypothesis is that sequents

[1] Technically, the $\forall$ rule could be applied, but only by quantifying over a variable which doesn't appear in the formula. In these cases, we would simply remove that step from the proof altogether.

of the form (3.10) have proofs of length $O(|A|^2)$ (specifically, proofs that are bounded by $c|A|^2$ for some constant c).

If $A(x)$ is simply x then we have:

$$\begin{prooftree}
\AxiomC{$p \rightarrow p$}
\doubleLine
\UnaryInfC{$\neg p, p \rightarrow q$}
\AxiomC{$q \rightarrow q$}
\doubleLine
\UnaryInfC{$q, p \rightarrow q$}
\BinaryInfC{$(\neg p \vee q), p \rightarrow q$}
\doubleLine
\UnaryInfC{$(\neg p \vee q) \wedge (\neg q \vee p), p \rightarrow q$}
\end{prooftree}$$

Then for longer formulas, remove the outermost logical connective in A and use to the induction hypothesis to show the existence (and length) of π_1 and π_2. For example, if $A(\vec{p}, \vec{a})$ is $B(\vec{p}, \vec{a}) \wedge C(\vec{p}, \vec{a})$ then:

$$\begin{prooftree}
\AxiomC{$\vdots\ \pi_1$}
\noLine
\UnaryInfC{$\vec{p} \equiv \vec{q}, B(\vec{p}, \vec{a}) \rightarrow B(\vec{q}, \vec{a})$}
\doubleLine
\UnaryInfC{$\vec{p} \equiv \vec{q}, B(\vec{p}, \vec{a}) \wedge C(\vec{p}, \vec{a}) \rightarrow B(\vec{q}, \vec{a})$}
\AxiomC{$\vdots\ \pi_2$}
\noLine
\UnaryInfC{$\vec{p} \equiv \vec{q}, C(\vec{p}, \vec{a}) \rightarrow C(\vec{q}, \vec{a})$}
\doubleLine
\UnaryInfC{$\vec{p} \equiv \vec{q}, B(\vec{p}, \vec{a}) \wedge C(\vec{p}, \vec{a}) \rightarrow C(\vec{q}, \vec{a})$}
\BinaryInfC{$\vec{p} \equiv \vec{q}, B(\vec{p}, \vec{a}) \wedge C(\vec{p}, \vec{a}) \rightarrow B(\vec{q}, \vec{a}) \wedge C(\vec{q}, \vec{a})$}
\end{prooftree}$$

The $\vee$, $\neg$, and $\exists$ cases proceed similarly. We need only choose c large enough to cover the largest of these three cases.

Secondly, we consider the rules one-by-one. We should note that, in π_E, the other formulas in a sequent (for example, those in Γ and Δ) may have been altered in previous steps so we must write Γ' and Δ' to acknowledge this.

Furthermore, when two sequents appear on top of a rule, Γ and Δ may not have been altered in the same ways in both subproofs, so we should write Γ'' and Δ'' in one of them to differentiate.

Exchange

The exchange rule needs no alteration, so wherever it is used in π_G we make the same exchange in π_E.

Weakening

If the new formula is quantifier free, there is nothing special to be done in π_E. Suppose the new formula has one or more quantifiers. If we introduce it on the left, then we choose new free variables (from the set of $\mathbf{p_i}$):

$$\frac{\Gamma' \rightarrow \Delta'}{\Gamma', A(\vec{p}, \vec{a}) \rightarrow \Delta'} \tag{3.11}$$

or, if we introduce a formula on the right, we use new $\mathbf{q}_i$ variables and perform further weakenings to introduce equivalances to give them values.

$$\frac{\Gamma' \rightarrow \Delta'}{q_1 \equiv 0, \ldots, q_n \equiv 0, \Gamma' \rightarrow A(\vec{q}, \vec{a}), \Delta'}\ \mathit{weak.}$$

We are free to select any value in the latter case, so we arbitrarily pick 0 for each of the n new variables.

Contraction

On unquantified formulas, we can simply perform the same contraction in π_E without any trouble. Applying the contraction rule to quantified formulas is slightly more difficult, because the formulas to be contracted may have different variable names (for formulas on the left), or different witnessing formulas (for formulas on the right).

Contracting on the left is simple. We can use the treelikeness of π_G and simply rename any or all of the variables in one instance of the formula to be contracted so that they are identical. Given:

$$\Gamma', A(\vec{p}, \vec{a}), A(\vec{q}, \vec{a}) \rightarrow \Delta' \tag{3.12}$$

We arbitrarily rename $\vec{q}$ to $\vec{p}$ throughout the proof up to this point, to obtain:

$$\Gamma', A(\vec{p}, \vec{a}), A(\vec{p}, \vec{a}) \rightarrow \Delta' \tag{3.13}$$

which we contract normally.

For contraction on the right, we start from the following top sequent:

$$\Gamma' \rightarrow A(\vec{p}, \vec{a}), A(\vec{q}, \vec{a}), \Delta' \tag{3.14}$$

where Γ' contains the definitions of $\vec{p}$ and $\vec{q}$. One, both, or neither or $A(\vec{p}, \vec{a})$ and $A(\vec{q}, \vec{a})$ may be true, and we need to be sure we do not eliminate the wrong one. We define a new set of variables $\vec{r}$ to be (for all i):

$$r_i \equiv \begin{cases} p_i \text{ if } A(\vec{p},\vec{a}) \text{ is true} \\ q_i \text{ otherwise} \end{cases} \tag{3.15}$$

One possible formula for this is $r_i \equiv (p_i \wedge A(\vec{p},\vec{a})) \vee (q_i \wedge \neg A(\vec{p},\vec{a}))$. We abbreviate this as $r_i \equiv \psi_i$.

Then a simple short proof of:

$$\vec{r} \equiv \vec{\psi}, \Gamma' \rightarrow A(\vec{r}), \Delta' \tag{3.16}$$

exists, which can be shown in a similar manner to the translations of axioms in (3.10).

This process for taking two possible sets of witnessing variables, and selecting the correct set by introducing a third set of variables will be useful when dealing with some of the other rules as well.

Connective Introduction

Since our formulas are in Σ_1, we know that the $\neg$ introduction rules will never be applied to quantified formulas, so there is no alteration necessary in π_E. For the $\wedge$ and $\vee$ rules, we convert to prenex normal form at the same time as we introduce the connective (as detailed above). This means that at subsequent stages of the proof, we can still assume that all formulas in all sequents in π_G start with a (possibly empty) block of quantifiers followed by a quantifier-free formula. All quantified variables in π_E are chosen to be unique, so we don't need to rename variables when converting to prenex normal form.

After this, for $\wedge$-left and $\vee$-right, there is nothing else to do. $\wedge$-right and $\vee$-left could have conflicting Γ and/or Δ which need to be resolved in the same way as in contraction (by renaming on the left, and by introducing new variables on the right).

For $\wedge$-left, if we have:

$$\Gamma' \rightarrow \Delta', A \quad \Gamma'' \rightarrow \Delta'', B \tag{3.17}$$

We unify these two sequents in the same way as in the contraction rule, renaming the variables on the left to obtain:

$$\Gamma''' \rightarrow \Delta', A \quad \Gamma''' \rightarrow \Delta'', B \tag{3.18}$$

We can then weaken and introduce the $\wedge$ connective, giving:

$$\Gamma''' \rightarrow \Delta', \Delta'', A \wedge B \tag{3.19}$$

Then, for each of the formulas in Δ which contain quantifiers (in π_G) we construct a new set of variables as in contraction (on the right). This lets us write:

$$\Gamma''' \rightarrow \Delta''', A \wedge B \tag{3.20}$$

$\vee$-left works in the same way, again following the procedure from the contraction case.

Cut

For cuts, in which the formula to be cut is quantifier-free, in π_E we will have:

$$\Gamma', A \rightarrow \Delta' \quad \Gamma'' \rightarrow A, \Delta'' \tag{3.21}$$

And we can weaken and cut to obtain:

$$\Gamma', \Gamma'' \rightarrow \Delta', \Delta'' \tag{3.22}$$

which we deal with as in contraction, $\wedge$-right, and $\vee$-left. Note that some of the formulas inside Γ'' may be equivalences for quantified variables in one upper sequent that are free variables in the other upper sequent.

If, in π_G, A contains a quantifier, then we have:

$$\Gamma', A(\vec{p}) \rightarrow \Delta' \quad \Gamma'' \rightarrow A(\vec{q}), \Delta'' \tag{3.23}$$

Since whenever we introduce a new variable in this translation, we always choose a previously unused $\mathbf{p}_i$ or $\mathbf{q}_i$, and since we never duplicate a formula, we know that $\Gamma', \Gamma'', \Delta'$ and Δ'' must not contain $\vec{p}$ or $\vec{q}$. Therefore, in the left sub-proof, we rename $\vec{p}$ to $\vec{q}$ throughout, and weaken in the witness definition for $\vec{q}$ (taken from the right sub-proof). We then perform any other unification necessary (as in contraction) and cut the formula.

$\exists$ introduction

We do not need to change anything whenever π_G contains an instance of $\exists$-left, because since the lower sequent in π_G doesn't contain the variable that was quantified.

However, for consistency, we choose the next $\mathbf{p}_i$ that is unused, and rename the variable to be quantified throughout the proof (again, it is the treelikeness of π_G that allows us to do this).

On the right, we need to make sure that we construct a new extension definition to witness the formula that was eliminated by the quantifier. That is:

$$\cfrac{\cfrac{\Gamma' \rightarrow \Delta', A(B)}{q \equiv B, \Gamma' \rightarrow A(B), \Delta', A(q)} \qquad \cfrac{\begin{array}{c}\pi_1\\ \vdots\\ q \equiv B, A(B) \rightarrow A(q)\end{array}}{q \equiv B, \Gamma', A(B) \rightarrow \Delta', A(q)}}{q \equiv B, \Gamma' \rightarrow \Delta', A(q)}$$

where again, π_1 is constructed in the same way as in (3.10) and in Lemma 27.

Finally, once this translation is complete we have π_E, a proof of some sequent $\Gamma' \rightarrow \Delta'$. Now, since the sequent proven by π_G is quantifier free, we know that:

$$\Gamma' \rightarrow \Delta' \overset{synt.}{=} \mathbf{q}_1 \equiv \phi_1, \ldots, \mathbf{q}_n \equiv \phi_n, \Gamma \rightarrow \Delta \tag{3.24}$$

and that none of the $\mathbf{q}_i$ appear in Γ or Δ. Therefore, to avoid violating the restriction that extension definitions must involve previously unused variables, we can go back up to the start of the proof, and introduce the n equivalence axioms $\rightarrow \mathbf{q}_i \equiv \phi_i$ in the same order that they were introduced in the above translation (which is in ascending order by i). We then weaken appropriately, and use n applications of the cut rule to produce the final sequent $\Gamma \rightarrow \Delta$ as required.

Since each of the steps described here require only polynomial-sized modifications, and we must perform at most one step per line of the proof (plus n polysized steps at the end) the whole process needs only a polynomial increase in the size of π_E. □

From the previous two lemmas we obtain:

Theorem 13 *Extended PK and G_1^* are p-equivalent.*

That is, the ability to quantify over Boolean variables gives the same power as substituting formulas, or introducing extension variables to abbreviate formulas.

Chapter 4

H: PK with permutation quantification

We can construct a new extension to PK which, like G, adds rules which introduce quantifiers. However, rather than quantifying over all possible assignments to a variable, our system, which we call H quantifies over all possible *transpositions* of two or more variables.

We will show that it is sufficient to quantify over transpositions of variables and to build more extensive permutations out of sequences of transpositions. That is, we need only give rules for permuting two variables at a time.

Syntax and Semantics

In PK and G, we were not explicit about the domain from which we selected our variables, but for H we need to be a little more precise. All formulas which appear in H proofs will have their variables chosen from the ordered, infinite set $\{P_1, P_2, P_3, \ldots\}$. We will sometimes use the symbols $a, b, c, \ldots$ or $a_1, a_2, a_3, \ldots$, but it should be understood that these are *meta-variables.* That is, each of them refers to a particular P_i.

Our new quantifiers use the same symbols as variable quantifiers, but we enclose them in brackets to delimit the two (or more) variables being permuted. If α is a formula, then so are $(\exists ab)\alpha$ and $(\forall ab)\alpha$, where one, both, or neither of a and b may appear in α and where a and b may be the same variable. These quantifiers are (semantically) equivalent to these formulas:

$$(\forall ab)\alpha(a,b) \equiv \alpha(a,b) \wedge \alpha(b,a) \quad \text{and} \quad (\exists ab)\alpha(a,b) \equiv \alpha(a,b) \vee \alpha(b,a) \tag{4.1}$$

We introduce the notation $\alpha^{(ab)}$ to represent a transposition of the variables a and b within α. It is important to note that occurrences of a or b within other permutation quantifiers are also affected. Hence, the formula $((\exists ac)\alpha)^{(ab)} = (\exists bc)\alpha^{(ab)}$

Every permutation can be decomposed into a series of transpositions and so transposition quantifiers will be sufficient to construct permutation quantifiers. However, the formal nature of propositional proof systems requires us to formulate a single, unique, and polynomial sized series of transposition quantifiers which will allow us to express *any* permutation.

Theorem 14 *Arbitrary permutations of* n *variables can be represented by the following series of transpositions (reading from left to right):*

$$\prod_{k=1}^{n}\prod_{j=1}^{n}(jk)^{i_{jk}} \tag{4.2}$$

where (i_{jk}) *is an* $n \times n$ *matrix over 0,1 indicating which transpositions are taken and which are left as the identity.*

PROOF: Take some arbitrary permutation of n variables $a_1 \mapsto a_{l_1}, a_2 \mapsto a_{l_2}, \ldots, a_n \mapsto a_{l_n}$.

We go through (4.2) in blocks of n transpositions at a time, in order. After the i^{th} block, we want to ensure that the first i variables have been permuted back to their correct places.

Each number in $1, \ldots, n$ appears exactly once in this permutation. In particular, some l_k must be 1. In the first block of n transpositions, we set $i_{1k} = 1$, and $i_{ij} = 0$ for all $j \neq k$. This ensures that the variable that should end up in position 1 is there, and that no other transpositions will be taken later to move it again.

Once we have done this for all n blocks of n transpositions, all n variables will be in the correct positions. □

Example: An example may be illustrative. Take $n = 6$, and let the variables begin in the following permutation: $a_2, a_4, a_3, a_5, a_6, a_1$. We must take the following transpositions to permute them back to $a_1, a_2, a_3, a_4, a_5, a_6$. (This is, of course, equivalent to starting in order, and then reordering the variables to some permutation).

We consider all the transpositions in (4.2) in order, but only the ones shown below are taken.

To prevent confusion between variable name and variable position, we show which i_{jk} should be set to 1, which indicates that the variables currently in the j^{th} and k^{th} positions should be exchanged.

$$\begin{array}{lll}
i_{16} = 1 & \text{gives} & a_1, a_4, a_3, a_5, a_6, a_2 \\
i_{26} = 1 & \text{gives} & a_1, a_2, a_3, a_5, a_6, a_4 \\
i_{33} = 1 & \text{gives} & a_1, a_2, a_3, a_5, a_6, a_4 \\
i_{46} = 1 & \text{gives} & a_1, a_2, a_3, a_4, a_6, a_5 \\
i_{56} = 1 & \text{gives} & a_1, a_2, a_3, a_4, a_5, a_6 \\
i_{66} = 1 & \text{gives} & a_1, a_2, a_3, a_4, a_5, a_6
\end{array}$$

Informally, this procedure works because it provides all n of the possible transpositions for each of the n variables, allowing us to pick and choose to construct any arbitrary permutation.

Definition: We define the *canonical transposition representation* of a permutation of k variables to be (4.2).

In H, we may use the following notation $(\forall a_1 a_2 \ldots a_n)$, which is an abbreviation for:

$$(\forall a_1 a_2)(\forall a_2 a_3) \ldots (\forall a_n a_n) \tag{4.3}$$

to express all possible permuatations of n variables (and similarly for $\exists$).

The semantics of permutation quantifiers are not difficult, but there are two different ways of thinking about what it means for a truth assignment τ to satisfy a formula containing a permutation quantifier. Firstly, we can say that τ satisfies a formula $(\exists ab)\alpha$ (written as $\tau \models (\exists ab)\alpha$), if and only if $\tau \models \alpha$ or $\tau \models \alpha^{(ab)}$. The second method is to define $\tau^{(ab)}$ by setting $\tau^{(ab)}(a) = \tau(b)$, $\tau^{(ab)}(b) = \tau(a)$, and $\tau^{(ab)}(x) = \tau(x)$

whenever $x \notin a, b$. We then say that $\tau \models (\exists ab)\alpha$ if and only if $\tau \models \alpha$ or $\tau^{(ab)} \models \alpha$. The cases for universal permutation quantifiers are similar, but require both transpositions to be satisfied, rather than just one. We refer to these methods respectively as *variable permutation* and *assignment permutation.*

Lemma 15 *Variable permutation and assignment permutation give equivalent semantics.*

PROOF: Take $(\exists ab)\alpha(a, b, x_1, \ldots, x_n)$ as an example. Let $\tau(a) = t_1$ and $\tau(b) = t_2$ where $t_1, t_2 \in \{0, 1\}$. If $\tau \models (\exists ab)\alpha(a, b, x_1, \ldots, x_n)$, then either $\alpha(t_1, t_2, \tau(x_1), \ldots, \tau(x_n))$ is true, or $\alpha(t_2, t_1, \tau(x_1), \ldots, \tau(x_n))$ is true.

Since $\tau(\alpha) \equiv \alpha(t_1, t_2, \tau(x_1), \ldots, \tau(x_n))$ and $\tau^{(ab)}(\alpha) \equiv \alpha(c_2, c_1, \tau(x_1), \ldots, \tau(x_n))$, either $\tau \models \alpha$ or $\tau^{(ab)} \models \alpha$ and we have one direction of the equivalence. The other direction (and the $\forall$ cases) follow from similar arguments. □

Rules

We present four unrestricted rules for introducing $\exists$ and $\forall$ transposition quantifiers on the left and on the right. For the existential quantifiers:

$$\frac{\alpha, \Gamma \rightarrow \Delta \quad \alpha^{(ab)}, \Gamma \rightarrow \Delta}{(\exists ab)\alpha, \Gamma \rightarrow \Delta} \quad \frac{\Gamma \rightarrow \Delta, \alpha}{\Gamma \rightarrow \Delta, (\exists ab)\alpha} \tag{4.4}$$

and for the universal quantifiers:

$$\frac{\alpha, \Gamma \rightarrow \Delta}{(\forall ab)\alpha, \Gamma \rightarrow \Delta} \quad \frac{\Gamma \rightarrow \Delta, \alpha \quad \Gamma \rightarrow \Delta, \alpha^{(ab)}}{\Gamma \rightarrow \Delta, (\forall ab)\alpha} \tag{4.5}$$

We also have two restricted introduction rules, in which a and b cannot appear in Γ or Δ:

$$\frac{\alpha, \Gamma \rightarrow \Delta}{(\exists ab)\alpha, \Gamma \rightarrow \Delta} \quad \frac{\Gamma \rightarrow \Delta, \alpha}{\Gamma \rightarrow \Delta, (\forall ab)\alpha} \tag{4.6}$$

From these rules, we can derive the permutation rule as follows:

$$\dfrac{\dfrac{\begin{array}{c}\pi\\ \vdots\\ \rightarrow \alpha(a,b)\end{array}}{\rightarrow (\forall ab)\alpha(a,b)} \quad \dfrac{\alpha(b,a) \rightarrow \alpha(b,a)}{(\forall ab)\alpha(a,b) \rightarrow \alpha(b,a)}}{\rightarrow \alpha(b,a)}$$

In [9] we defined a different set of rules which included the permutation rule, and where the restricted rules were derived rules. However, it turns out that that definition of H did not allow us to translate $\forall P$**LA** (see Chapter 5) while preserving treelikeness.

Soundness and Completeness

We can easily show that these concepts apply to H as well as to G and to basic PK.

Theorem 16 *H is sound and complete. Furthermore, the inversion principle applies to all the rules of H, except for the weakening rule*

PROOF: H is complete because it contains the rules of PK, and PK is complete. We show soundness and inversion as before, by taking any truth assignment and showing that if it satisfies the upper sequent then it must satisfy the lower (and vice versa, except for weakening). For example, consider the $\exists$-left rule. Suppose that:

$$\Gamma, \alpha(a,b) \rightarrow \Delta \tag{4.7}$$

is valid, but that there is some truth assignment τ that falsifies:

$$\Gamma, \exists(ab)\alpha(a,b) \rightarrow \Delta \tag{4.8}$$

Then $\tau \models \Gamma$, and $\tau \models \exists(ab)\alpha(a,b)$ but $\tau \not\models \Delta$. Since $\tau \models \exists(ab)\alpha(a,b)$ it must be true that either $\tau \models \alpha(a,b)$ or $\tau^{(ab)} \models \alpha(a,b)$. Furthermore, since a and b do not appear in Γ and Δ, we know that $\tau^{(ab)} \models \Gamma$ and $\tau^{(ab)} \not\models \Delta$, and so either τ or $\tau^{(ab)}$ must falsify the top sequent. We can argue similarly about the other new rules. □

Extended Completeness of H

Previously, we showed that if S was any valid sequent which possibly contained variable quantifiers, then a G-proof of S exists. We can show a similar result for H.

Theorem 17 *H is complete for sequents with permutation quantifiers. That is, if S is a valid sequent which possibly contains permutation quantifiers, then an H-proof of S exists.*

PROOF: The proof is again by induction on the maximum number of transposition quantifiers in any formula in a sequent.

The base case is a valid sequent without transposition quantifiers. By Theorem 3 there is a PK proof of this sequent, and this proof can also be constructed in H.

Assume that any sequent in which each formula has at most k transposition quantifier has an H proof, and then let S be a sequent containing a maximum of $k+1$ quantifiers in any of its formulas.

For each formula, α in S if α contains $k+1$ quantifiers, we do the following:

If the outermost connective in α is not a permutation quantifier, then apply the $\vee$, $\wedge$, and $\neg$ introduction rules in reverse. Continue this until a permutation quantifier is the outermost connective.

At this point, S is one of the following: (The cases on the left proceed analogously)

$$\Gamma \to (\exists ab)\alpha(a,b), \Delta \qquad \Gamma \to (\forall ab)\alpha(a,b), \Delta \tag{4.9}$$

For the $\forall$-right case, we build a partial proof as follows:

$$\cfrac{\begin{matrix}\pi_1 & \pi_2 \\ \vdots & \vdots \\ \Gamma \to \alpha, \Delta & \Gamma \to \alpha^{(ab)}, \Delta\end{matrix}}{\Gamma \to (\forall ab)\alpha^{(ab)}, \Delta}\ \forall - right$$

We then continue with all the other formulas in Γ and Δ, removing a permutation quantifier from any formula with $k+1$ quantifiers. At this point, all the uppermost sequents will have k or fewer permutation quantifiers in each formula and then the π_i will exist by the induction hypothesis.

The $\exists$-right case is slightly more complicated:

$$\begin{array}{cl}
\pi & \\
\vdots & \\
\Gamma \to \alpha, \alpha^{(ab)}, \Delta & \\
\cline{1-1}
\Gamma \to (\exists ab)\alpha^{(ab)}, \alpha^{(ab)}, \Delta & \raise{1.2ex}{\exists - right} \\
\cline{1-1}\cline{1-1}
\Gamma \to (\exists ab)\alpha^{(ab)}, (\exists ab)\alpha^{(ab)}\Delta & \raise{1.2ex}{exch., \exists - right} \\
\cline{1-1}
\Gamma \to (\exists ab)\alpha^{(ab)}, \Delta & \raise{1.2ex}{contr.}
\end{array}$$

□

Now we would like to explore H's power as compared to that of G. It is quite simple to see that Extended PK can p-simulate H_1^*, but a significantly longer process to demonstrate that H_1^* can p-simulate Extended PK.

4.1 Extended PK p-simulates H_1^*

To show this, we use a similar method as used in the previous chapter to show that Extended PK could p-simulate G_1^*. Given a proof in H_1^*, we build an equivalent Extended PK proof in which extension definitions are introduced on the left to witness whether or not a transposition is taken on the right, in this fashion:

$$\{d_1, d_2\} \equiv \{\phi_1, \phi_2\}, \ldots, \{d_{n-1}, d_n\} \equiv \{\phi_{n-1}, \phi_n\}, \Gamma \to \Delta \tag{4.10}$$

where the d_i are the variables which appear inside transposition quantifiers on the right, and the ϕ_i are formulas which specify whether the transposed or untransposed pair of variables makes the sequent valid (that is, the values which would have satisfied the quantified formulas in π_H). The notation $\{d_1, d_2\} \equiv \{c_1, c_2\}$ is used to abbreviate $d_1 \equiv c_1 \wedge d_2 \equiv c_2$, where $\equiv$ is now the usual abbreviation.

We refer to new variables on the left as free variables, and new variables on the right as 'witness' variables. Every witness variable has a defining formula on the left.

As we did when simulating G_1^*, we show that we can convert the axioms and weakened formulas to a form containing only existential quantifiers in prenex form, and then only work with formulas in prenex form for the rest of the conversion. The process is the same; rename the quantified variables, and push out the quantifiers past the other connectives (converting $\neg\forall$ to $\exists\neg$ as before).

Theorem 18 *Extended PK p-simulates H_1^*.*

Proof: Let π_H be a H_1^* proof of the classical (i.e. quantifier free) sequent $\Gamma \rightarrow \Delta$. We construct an extended PK proof, π_E, of the same sequent with only a polynomial increase in proof size.

We convert the axioms of π_H as follows. Any quantifier-free axioms require no alteration. For any quantified axiom, we put the axiom into prenex normal form, so that it is of the form:

$$(\exists a_1 a_2)\ldots(\exists a_{2n-1}a_{2n})A(\vec{a},\vec{x}) \rightarrow (\exists a_1 a_2)\ldots(\exists(a_{2n-1}a_{2n})A(\vec{a},\vec{x}) \tag{4.11}$$

We then construct the following valid sequent:

$$\{c_1,c_2\} \equiv \{d_1,d_2\},\ldots,\{c_{2n-1},c_{2n}\} \equiv \{d_{2n-1},d_{2n}\}, A(\vec{d},\vec{x}) \rightarrow A(\vec{c},\vec{x}) \tag{4.12}$$

We use structural induction on A to show an efficient construction of (4.12). The induction is basically identical to induction used to deal with axioms in the simulation of G_1^* by Extended PK, and so is not repeated here.

We also need to change the way we define the new witnessing variables that are introduced during the unification of two sequences of formulas. Take for example contraction on the left. In π_E we will have (as in the translation from π_G to π_E in the previous chapter):

$$\Gamma' \rightarrow \Delta', A(p,q), A(r,s) \tag{4.13}$$

where p, q, r, and s are witnessed variables which represent quantified variables from π_H. We need to introduce a new pair of variables t and u, such that in the sequent:

$$\Gamma' \rightarrow \Delta', A(t,u) \tag{4.14}$$

if one of $A(p,q)$ or $A(r,s)$ was true, then $A(t,u)$ is now equivalent to the true formula. To ensure this, we use this extension definition:

$$\{t,u\} \equiv \{\phi_1, \phi_2\} \tag{4.15}$$

where

$$\phi_1 \equiv \begin{cases} p \text{ if } A(p,q) \text{ is true} \\ r \text{ otherwise} \end{cases} \tag{4.16}$$

and likewise

$$\phi_2 \equiv \begin{cases} q \text{ if } A(p,q) \text{ is true} \\ s \text{ otherwise} \end{cases} \tag{4.17}$$

Explicitly, (4.16) can be written as $t \equiv (p \wedge A(p,q)) \vee (r \wedge \neg A(p,q))$, and (4.17) can be written $u \equiv (q \wedge A(p,q)) \vee (s \wedge \neg A(p,q))$.

For the rest of the translation, we follow the procedure to translate G_1^* proofs into Extended PK. The transposition quantifiers are dealt with slightly differently to the ordinary quantifiers (as one would expect). For existential quantifiers on the left there is still nothing that needs to be done, and on the right we need to introduce extension definitions which witness whether or not a transposition was taken. In π_H we would have had:

$$\dfrac{\Gamma \to \Delta, A(a,b)}{\Gamma \to \Delta, (\exists ab) A(a,b)}$$

which appears in π_E as:

$$\dfrac{\dfrac{\Gamma' \to \Delta', A(a,b)}{\{c,d\} \equiv \{a,b\}, \Gamma' \to A(a,b), \Delta', A(c,d)} \qquad \dfrac{\begin{matrix} \pi_1 \\ \vdots \\ \{c,d\} \equiv \{a,b\}, A(a,b) \to A(c,d) \end{matrix}}{\{c,d\} \equiv \{a,b\}, \Gamma', A(a,b) \to \Delta', A(c,d)}}{\{c,d\} \equiv \{a,b\}, \Gamma' \to \Delta', A(c,d)}$$

where the existence of (and polynomial bound on the size of) π_1 follows from Lemma 27. □

We currently have no direct simulation of Extended PK by H_1 or H_1^*. However, we can show how H_1^* can indirectly p-simulate Extended PK. The remaining chapters of this thesis are devoted to introducing the necessary intermediate systems, and giving the step-by-step simulations that connect H_1^* to Extended PK.

Chapter 5

LA and its translations

LA is a first order logical theory of Linear Algebra. **LA** has three sorts: *indices*, *field elements*, and *matrices*, and operates over sequents. All of the rules of PK are available for constructing **LA** proofs from the axiom schemes described in the appendix, or in [12]. **LA** is quantifier-free[1], but every sequent is implicitly universally quantified (in the same way that PK operates only over valid propositional sequents). We describe only those elements of **LA** that are important for our purposes; for a full introduction of LA see [12], and see [14] for an introduction of the related theory $\forall P$**LA**.

$\forall P$**LA** is important to us, because it is powerful enough to prove the soundness of the Hajós Calculus, which is p-equivalent to Extended Frege, but on the other hand it is weak enough that it can in turn be easily translated into a propositional proof system like H_1^*. $\forall P$**LA** therefore forms a crucial link in our proof that H_1^* can p-simulate Extended PK. First let us define **LA**.

Function Symbols

There are several function symbols for each type. Those for indices:

$$0_{\text{index}}, 1_{\text{index}}, +_{\text{index}}, *_{\text{index}}, -_{\text{index}}, \text{cond}_{\text{index}} \tag{5.1}$$

For field elements:

[1] Except for bounded index quantifiers

$$0_{\text{field}}, 1_{\text{field}}, +_{\text{field}}, *_{\text{field}}, -_{\text{field}}, ^{-1}, \text{cond}_{\text{field}} \tag{5.2}$$

And for matrices:

$$r, c, e, \Sigma \tag{5.3}$$

0 and 1 are constant functions of the indicated type. $+, *$ are binary functions of the indicated type. $-_{\text{index}}$ is a binary function symbol, while $-_{\text{field}}$ and $^{-1}$ are unary function symbols. All of these have their conventional meanings.

cond(α, t, s) is a conditional statement equal to the term t if the formula α is true, and to the term s otherwise. $e(A, i, j)$ is the element of the matrix A in the i^{th} row and the j^{th} column, or 0 if A has fewer that i rows or fewer than j columns. r and c take a matrix, A, as an argument and return index values corresponding to the number of rows and columns in A respectively. Finally, Σ takes a matrix A and returns a term which is the sum of the entries of A.

There are also several predicate symbols: $\leq_{\text{index}}, =_{\text{index}}, =_{\text{field}}$, and $=_{\text{matrix}}$. From these we can build the atomic formulas of **LA**:

$$\begin{gathered} i \leq_{\text{index}} j \\ i =_{\text{index}} j \\ s =_{\text{field}} t \\ A =_{\text{matrix}} B \end{gathered} \tag{5.4}$$

From these base cases, terms and formulas are built inductively using function symbols and logical connectives respectively. The logical connectives are introduced using the rules from PK, constructing sequents from the axioms of **LA**. Other function symbols may be defined in terms of these symbols to abbreviate formulas and sequents, but any such symbols are not part of the language, they are merely meta-symbols. Where the type of an expression is obvious, we will omit the subscripts.

5.1 Translations of LA theorems into propositional formulas

Theorems of **LA** can be translated into families of short propositional tautologies. Let α be some formula of **LA**, and let σ be some assignment to the objects in α.

Furthermore, let n be the largest number that σ assigns to any index variable in α. We define the size of σ, written $|\sigma|$, to be n. Then there is a propositional formula which is equivalent to α under σ, and this formula is only polynomially larger (in $|\sigma|$) than α. This construction follows Soltys and Cook [14] to show that basic **LA** translates to PK. We repeat their proof here, and then extend it to cover translations of $\forall P\mathbf{LA}$ into H_1^*.

This general idea behind this translation is to evaluate as much as possible within each formula explicitly. We translate a formula to 1 or 0 if we can tell that (under σ) it will always be true or false. For example, for two matrix variables A and B, the formula $A = B$ will always be false if the dimensions of A and B (namely $\sigma(r(A)$, $\sigma(r(B))$ etc.) are not equal. If the dimensions are equal, then we translate $A = B$ into a particular formula asserting that two $\sigma(r(A)) \times \sigma(c(A))$ matrices are equal.

It is important to remember that at the time of translation we know what σ is, and therefore, amoungst other things, what the dimensions of each matrix in α are.

Note also that, in particular, all formulas of type index can be evaluated to a single constant during the translation. This allows us to avoid dealing with **LA**'s bounded index quantifiers, which would be impossible in PK.

Once we can translate **LA** formulas into propositional formulas, it is natural to proceed to translate **LA** proofs into propsitional proofs. PK provides a convenient formalism, because both systems work over sequents, and both systems use the same set of rules for connective introduction and sequent manipulation. However, for convenience, we modify PK to include an exclusive-OR connective, $\oplus$ with the following left- and right- introduction rules:

$$\frac{\Gamma, A, \neg B \rightarrow \Delta \quad \Gamma, \neg A, B \rightarrow \Delta}{\Gamma, A \oplus B \rightarrow \Delta} \quad \frac{\Gamma \rightarrow (A \vee B), \Delta \quad \Gamma \rightarrow (\neg A \vee \neg B), \Delta}{\Gamma \rightarrow A \oplus B, \Delta} \tag{5.5}$$

Since any two Frege systems over different (but complete) sets of logical connectives are equivalent[3], so are any two PK systems over differing sets of complete connectives, so we are free to make this modification. Also, since we are interested primarily in the case that field variables are chosen from $\mathbb{Z}_2$, we will restrict our translations to this case. Similar arguments hold over other fields, but much more detail is needed to deal with field variables and field operations.

Lemma 19 *Given a formula α of **LA** and an object assignment σ, we can construct a* *propositional formula $||\alpha||_\sigma$. In this way, we build a family of propositional formulas, depending on σ for each formula of **LA**.*

PROOF: We must show how to translate the three types of objects that may appear in an **LA** formula, namely indices, field elements, and matrices.

Indices: Let m and n be two index variables in α. σ assigns some natural number to each. The only atomic formulas that can appear are of the form $m =_{\text{index}} n$ or $m <_{\text{index}} n$. Therefore, rather than use propositional variables to represent the indices themselves we assign a single propositional variable to each such atomic formula, where:

$$||m = n||_\sigma = P_{m=n} = \begin{cases} 1 & \text{if } \sigma(m) = \sigma(n) \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

and similarly for $m \leq n$. This allows us to explicitly evaluate some more complicated fomulas involving indices as well. For example, the conditional statement; we translate the formula $\text{cond}_{\text{index}}(\beta, m, n)$, where β is some atomic formula, to:

$$||\text{cond}(\beta, m, n)||_\sigma = \begin{cases} ||m||_\sigma & \text{if } ||\beta||_\sigma \text{ is true} \\ ||n||_\sigma & \text{otherwise} \end{cases} \tag{5.7}$$

We evaluate addition, multiplication and other operations on index variables in the usual manner to determine the value of each P_β.

Field Elements: Recall that, for our purposes, σ assigns each field variable a value from $\mathbb{Z}_2$. This means that for each field variable a, we can use a single propositional variable which is true iff $\sigma(a) = 1$. We make the obvious choices for 0_{field} and 1_{field}. Therefore, we have $||a||_\sigma = a$, $||0_{\text{field}}||_\sigma = 0$, and $||1_{\text{field}}||_\sigma = 1$. A term that is a matrix element is translated in the expected way, namely:

$$||e(A, m, n)||_\sigma = \begin{cases} ||A_{||m||_\sigma ||n||_\sigma}||_\sigma & \text{if } 1 \leq m \leq \sigma(r(A)) \text{ and } 1 \leq n \leq \sigma(c(A)) \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

From these base cases, we inductively construct a formula for each term as follows, where s and t are terms which have already been converted to propositional formulas:

$||s+t||_\sigma = (||s||_\sigma \oplus ||t||_\sigma)$, $||s*t||_\sigma = (||s||_\sigma \wedge ||t||_\sigma)$, $||-t||_\sigma = ||t||_\sigma$, and $||t^{-1}||_\sigma = ||t||_\sigma$. $||\text{cond}(\beta, s, t)||_\sigma$ is the same as for index variables. It is simple to show that these formulas give the correct values over the field of two elements. Constructed terms and matrix summations are a little more complex. If we wish to describe a term selected from a constructed matrix, we must modify our object assignment to select the correct element:

$$||e(\lambda ij(m', n', t), m, n)||_\sigma = \begin{cases} ||t||_{\sigma'} & \text{if } 1 \leq ||m||_\sigma \leq ||m'||_\sigma \text{ and } 1 \leq ||n||_\sigma \leq ||n'||_\sigma \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

σ' is the modified object assignment. It is identical to σ, except that $\sigma'(i) = ||m||_\sigma$ and $\sigma'(j) = ||n||_\sigma$. In this way, σ' is used to select the desired element from the λ term.

Similarly, we translate Σ (matrix summation) terms over λ terms by introducing one new object assignment for each element of the λ term being summed over:

$$||\Sigma(\lambda ij(m, n, t))||_\sigma = \bigoplus_{\substack{1 \leq p \leq ||m||_\sigma \\ 1 \leq q \leq ||n||_\sigma}} ||t||_{\sigma_{pq}} \tag{5.10}$$

where each of the σ_{pq} is identical to σ, except that $\sigma_{pq}(i) = p$ and $\sigma_{pq}(j) = q$. Σ formulas over non-constructed matrices are translated in the expected fashion:

$$||\Sigma(A)||_\sigma = \bigoplus_{\substack{1 \leq p \leq ||m||_\sigma \\ 1 \leq q \leq ||n||_\sigma}} A_{pq} \tag{5.11}$$

where here the A_{pq} are the propositional variables corresponding to the entries of A.

The only atomic formula over field elements is $s =_{\text{field}} t$, which we translate to $(||s||_\sigma \equiv ||t||_\sigma)$.

Matrices: Variables of type matrix are represented by a number of propositional variables. For example, an $m \times n$ matrix, A, is represented by the mn propositional variables A_{pq} where $1 \leq p \leq m$ and $1 \leq q \leq n$. Constructed matrices are represented by the terms from which they are constructed, under the appropriate object assignment.

Equality for matrices is more complicated than for field elements. Given an atomic formula:

$$A =_{\text{matrix}} B \tag{5.12}$$

if $||r(A)||_\sigma \neq ||r(B)||_\sigma$ or $||c(A)||_\sigma \neq ||c(A)||_\sigma$ (that is, if the matrices are of different sizes) then $||A = B||_\sigma = 0$. If they are of the same size, then they are equal if all their elements are equal, and in that case:

$$||A = B||_\sigma = \bigwedge_{1 \leq p,q \leq ||r(A)||_\sigma} \left(||e(A,i,j)||_{\sigma_{pq}} \equiv ||e(B,i,j)||_{\sigma_{pq}}\right) \tag{5.13}$$

Connectives: Once we have translations for the atomic formulas of **LA** we can introduce logical connectives $\wedge$, $\vee$, and $\neg$ in the usual way namely $||\alpha \wedge \beta||_\sigma = ||\alpha||_\sigma \wedge ||\beta||_\sigma$, $||\alpha \vee \beta||_\sigma = ||\alpha||_\sigma \vee ||\beta||_\sigma$, and $||\neg\alpha||_\sigma = \neg||\alpha||_\sigma$. In this way we build translations of entire **LA** formulas. □

Theorem 20 *Translations of LA into PK are polynomially bounded. That is, for each formula α, there is a polynomial p_α such that for any object assignment σ to α, the length of $||\alpha||_\sigma$ is bounded by $p(|\sigma|)$. Also, α is valid under σ over the field $\mathbb{Z}_2$ iff $||\alpha||_\sigma$ is a tautology.*

PROOF: We prove the two claims separately, starting with the size bound. We first prove a polynomial bound on the *value* of $||m||_\sigma$ for each index term m, and then a polynomial bound on the *size* of $||t||_\sigma$ for each field term t.

Indices: Index variables appear in the atomic formulas $n = m$ or $n \leq m$ and therefore an index term appears in the translated formula as either True or False.

However, we also need to prove that the value of any index term m is polynomially bounded by $p_{ind}(|\sigma|)$, because this will bound the size of any matrix term in our formula. This is easily shown: if m is some index variable, then the value of m is bounded by $|\sigma|$ (by the definition of $|\sigma|$). Then, assume that n, p are two index terms, bounded in value by $|n||\sigma|$ and $|p||\sigma|$ respectively, then $n + p$ is bounded in value by $|n + p||\sigma|$.

Field Elements: Recall that for this translation, the underlying field is assumed to be $\mathbb{Z}_2$, so all field variables are either equal to 0 or 1. This makes our translations much simpler, because 0_{field} and 1_{field} can be translated to the Boolean variables 0 and 1 respectively.

Then, if t and u are terms of type field, and are bounded respectively by $p_t(|\sigma|)$ and $p_u(|\sigma|)$, then $||t+u||_\sigma = ||t||_\sigma \oplus ||u||_\sigma$ and $||t*u||_\sigma = ||t||_\sigma \wedge ||u||_\sigma$ are both bounded by $(p_t + p_u + 1)(|\sigma|)$. $||-t||_\sigma$ and $||t^{-1}||_\sigma$ are similarly bounded by $(p_t + 1)(|\sigma|)$.

That leaves the more complicated cases. Assume $\lambda ij(m,n,t)$ is some constructed term, and assume that the translations of each entry t_{ij} are polynomially bounded $p_{t_{ij}}(|\sigma|)$. Define p_{max} to be the maximum of these polynomials. Then $||e(\lambda ij(m,n,t),k,l)||_\sigma$ is bounded by $p_{kl}(|\sigma|)$ if $k \le m$ and $l \le n$, and bounded by 1 otherwise (see (5.8)). The translation of a summation term $||\Sigma(\lambda ij(m,n,t))||_\sigma$ is a formula with mn translated terms, and $mn - 1$ $\oplus$ connectives. Each term is bounded by a polynomial less than or equal to p_{max}, so the whole translation is bounded by $(mn * p_{\text{max}}(|\sigma|) + mn - 1)$.

Matrices: Matrix terms are either matrix variables or constructed terms. A matrix variable A is translated into polynomially many field variables ($\sigma(r(A))\sigma(c(A))$ many, in fact). Each field variable is translated into a single propositional variable. A constructed matrix $\lambda ij(m,n,t)$ is translated into mn terms. Each term gets translated as above, and is bounded by a polynomial, so we choose the largest of these to bound the translation of the constructed matrix (as for summation above).

Formulas: Equality of field variables translates to a constant sized formula. For matrix variables, $A = B$ translates to:

$$\bigwedge_{\substack{1 \le i \le \sigma(r(A)) \\ 1 \le j \le \sigma(c(A))}} (P_A)_{ij} \equiv (P_B)_{ij} \tag{5.14}$$

which is bounded by $c|\sigma|^2$, for some constant c. For equality of constructed terms, we replace c by the largest polynomial that bounds an element of the constructed matrices. From here, we proceed by structural induction to show polynomial bounds (in $|\sigma|$) for each formula.

Lastly, we prove the claim that $||\alpha||_\sigma$ is a tautology if α is valid under σ. Valid atomic formulas of type index are trivially translated to a tautology. If $m =_{\text{field}} n$ is true (under σ), then $||m||_\sigma \equiv ||n||_\sigma$ is a tautology, and similarly for equality of matrix variables. From here, we prove by structural induction that larger valid formulas are translated into tautologies. Note that our task is considerably easier here because it is clear that $\oplus$ and $\wedge$ are equivalent to addition and multiplication over $\mathbb{Z}_2$. Translations exist for more complicated fields, but need to use a number of propositional variables for each field element. For more details, see [14]. □

So we have shown that we can construct translations of **LA** formulas, and use those to translate **LA** sequents into a propositional proof system. PK (with $\oplus$) is a natural choice because **LA** is based on the sequent calculus introduced in Chapter 2. Note that we only need $\oplus$ for translating addition; it will never appear in the original **LA** proof we are translating.

If α is a valid sequent under some object assignment σ, then we know that $||\alpha||_\sigma$ is a tautology, and hence has a PK proof. It is natural to ask how to find this proof. Luckily, α's validity implies the existence of an **LA** proof. PK (with $\oplus$) can use this **LA** proof as a template for the proofs of the family $\{||\alpha||_\sigma\}$.

We have shown how **LA** axioms get translated into propositional formulas, but in order to be able to use the **LA** proof as a template for a PK proof, we also need to show that PK can efficiently prove the **LA** axioms. The full set of **LA** axioms are given in the appendix. Their translations are proven in the following way:

Equality Axioms: Recall that $\rightarrow x = x$ translates to $\rightarrow x \equiv x$ which is an abbreviation of $\rightarrow (\neg x \vee x) \wedge (x \vee \neg x)$, which has a simple PK proof from the PK axiom $x \rightarrow x$ if x is a field element, and or from a set of axioms if x is a matrix variable. The other equality axioms are proven similarly.

Index Axioms: The index axioms translate into either true or false depending on σ. Therefore, the translated index axioms can be proven from $0 \rightarrow$ or $\rightarrow 1$ with a constant number of weakenings.

Field Axioms: We give an example proof for field axiom **F18**. The axiom is $\rightarrow 0 \neq 1 \wedge a + 0 = a$. It translates to:

$$\rightarrow \neg(0 \equiv 1) \wedge (||a||_\sigma \oplus 0) \equiv ||a||_\sigma \tag{5.15}$$

where the brackets are inserted for readability. We prove this in PK (with $\oplus$) as follows. It is easy to obtain a proof of $\rightarrow \neg(0 \equiv 1)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\rightarrow 1}{\neg 1 \rightarrow} \quad 0 \rightarrow}{\neg 1 \vee 0 \rightarrow}}{\neg 0 \vee 1, \neg 1 \vee 0 \rightarrow}}{(\neg 0 \vee 1) \wedge (\neg 1 \vee 0) \rightarrow}}{\rightarrow \neg((\neg 0 \vee 1) \wedge (\neg 1 \vee 0))}}{}$$

Likewise, we prove $\rightarrow (||a||_\sigma \oplus 0) \equiv ||a||_\sigma$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{||a||_\sigma \rightarrow ||a||_\sigma}{||a||_\sigma, \neg 0 \rightarrow ||a||_\sigma} \quad \dfrac{0 \rightarrow}{\neg ||a||_\sigma, 0 \rightarrow ||a||_\sigma}}{||a||_\sigma \oplus 0 \rightarrow ||a||_\sigma}}{\rightarrow \neg(||a||_\sigma \oplus 0), ||a||_\sigma}}{\rightarrow \neg(||a||_\sigma \oplus 0) \vee ||a||_\sigma} \qquad \dfrac{\dfrac{\dfrac{\dfrac{||a||_\sigma \rightarrow ||a||_\sigma}{\rightarrow ||a||_\sigma, 0, \neg ||a||_\sigma}}{\rightarrow ||a||_\sigma \vee 0, \neg ||a||_\sigma} \quad \dfrac{\dfrac{0 \rightarrow}{\rightarrow \neg ||a||_\sigma, \neg 0, \neg ||a||_\sigma}}{\rightarrow \neg ||a||_\sigma \vee \neg 0, \neg ||a||_\sigma}}{\rightarrow ||a||_\sigma \oplus 0, \neg ||a||_\sigma}}{\rightarrow (||a||_\sigma \oplus 0) \vee \neg ||a||_\sigma}}{\rightarrow (\neg(||a||_\sigma \oplus 0) \vee ||a||_\sigma) \wedge ((||a||_\sigma \oplus 0) \vee \neg ||a||_\sigma)}$$

From these two proofs we use $\wedge$ introduction to get the required result. Note that $||a||_\sigma$ is some propositional variable (a is a reasonable choice) so that $||a||_\sigma \rightarrow ||a||_\sigma$ is a PK axiom. The other axioms for field elements are translated similarly.

Matrix Axioms: The proofs for matrix axioms are similar, but depend on whether the index variables are within the matrix dimensions (in which case we get a family of proofs similar to the one above, with one for each entry in the matrix), or whether they are outside in which case we have a short proof from the axiom $0 \rightarrow$.

Once we have proven the translated axioms, we can use the rules of PK to follow the original **LA** proof.

5.2 $\forall P$LA and its translations

$\forall P$**LA** augments **LA** by adding the following rules for introducing bounded universal permutation quantification:

$$\frac{(P \leq n \wedge Perm(P)) \supset \alpha, \Gamma \rightarrow \Delta}{(\forall P \leq n)\alpha, \Gamma \rightarrow \Delta} \quad \frac{\Gamma \rightarrow \Delta, (P \leq n \wedge Perm(P)) \supset \alpha}{\Gamma \rightarrow \Delta, (\forall P \leq n)\alpha} \tag{5.16}$$

where in the right introduction rule, P cannot appear free in the lower sequent. Similarly, $\exists P$**LA** adds two rules for introducing bounded existential permutation quantification (although the restriction on P applies instead to the left introduction rule). Note that $(P \leq n)$ is an abbreviation of $(r(P) \leq n \wedge c(P) \leq n)$ where $r(P)$ and $c(P)$ are the rows and columns of P respectively.

By adding these rules, $\forall P$**LA** (and $\exists P$**LA**) allows for induction over formulas $(\forall P \leq n)\alpha$ (and $(\exists P \leq n)\alpha$), which **LA** was unable to express. Happily, it turns out that H_1^* is an adequate extension to PK, as we now prove.

Recall that H_i is H restricted to formulas with i alternations of $\exists$ and $\forall$ quantifiers, and that H_i^* is treelike H_i. We will use Π_1 to denote formulas of the form $(\forall P)\alpha$ where α is quantifier free. $(\forall P)$ is short for $(\forall P_{11}P_{12})(\forall P_{11}P_{13})\ldots(P_{n-1n}P_{nn})$, and P_{ij} is a propositional variable. H_1^* is therefore PK, where formulas are allowed to be in Π_1, and all proofs must be treelike.

Lemma 21 *$\forall P$**LA** formulas under some object assignment σ translate into families of Π_1 formulas. The size of these formulas is polynomial in $|\sigma|$.*

Proof: When α is a formula of **LA**, we have already shown how to construct the family of propositional formulas $||\alpha||_\sigma$, and shown that this family of formulas has size polynomial in α. Let $(\forall P \leq n)\alpha$ be some formula of $\forall P$**LA**. We can assume that α is quantifier-free (because if it did contain quantifiers we could uniquely rename the quantified matrices and push them to the front block of quantifiers). Translate this formula to the following propositional family:

$$(\forall P)(\bigwedge_i P_{ii} \equiv 1 \wedge \bigwedge_{j \neq i} P_{ij} \equiv 0 \wedge \bigwedge_{i,j}(P_{ij} \supset \bigwedge_{r \neq i, s \neq j} \neg P_{rs})) \supset ||\alpha||_\sigma \tag{5.17}$$

or, for $\exists$:

$$(\exists P)(\bigwedge_i P_{ii} \equiv 1 \wedge \bigwedge_{j \neq i} P_{ij} \equiv 0 \wedge \bigwedge_{i,j}(P_{ij} \supset \bigwedge_{r \neq i, s \neq j} \neg P_{rs})) \wedge ||\alpha||_\sigma \tag{5.18}$$

where the initial block of transposition quantifiers is the $O(n^2)$ block of transposition quantifiers used to represent any permutation, $||\alpha||_\sigma$ is the propositional formula corresponding to the **LA** formula α, under object assignment σ, and where the rest of the formula asserts that, no matter which transpositions are taken, P remains a valid permutation matrix. This is important, because since we quantify over transpositions of variables (elements of P), we need some mechanism to ensure that we are left with a valid permutation of the identity matrix.

To see that this formula accomplishes that, note that the first two conjunctions assert that P 'begins' as the identity matrix (that is, if no transpositions are taken, $P = I$). Note that any formula asserting that P has n 1's and $n^2 - n$ 0's for its entries will suffice, and that the only reason we prefer to equate P to the identity matrix (other than aesthetic reasons) is that all of the transpositions can be left 'off' and still produce a valid permutation matrix.

Then, the last conjunction is satisfied if and only if at most one entry per row of P, and one entry per column of P is 1. Since there are exactly n entries of P which are 1, this means any permutation of the elements of P must have exactly one 1 per row and per column to satisfy the last conjunction. That is, only transpositions which leave P as a valid permutation matrix will satisfy this new formula. Since it plays an analogous role to the Perm(P) formula from $\forall P$**LA**, we will denote this propositional formula as $||\text{Perm}(P)||_\sigma$. □

Next, we must show that $\forall P$**LA** proves a formula, then the family of translated formulas $\{||\alpha||_\sigma\}$ has short proofs in H_1^*.

Lemma 22 *If* $\forall P\boldsymbol{LA} \vdash \alpha$ *then* $\{||\alpha||_\sigma\}$ *is a family of valid sequents with polynomial sized (in* $|\sigma|$*)* H_1^* *proofs.*

Proof: We have already seen how formulas of **LA** translate into tautologies with short PK proofs, and those arguments will hold here also. What remains is to show that the rules for quantifier introduction can also be dealt with. We show the cases for $\forall - left$ and $\forall - right$; the $\exists$ cases would be analogous.

The base cases need no modification. Axioms of $\forall P$**LA** are substitution instances of **LA** axioms, and therefore are proven in H_1^* in the same way that translations of **LA** axioms were proven in PK.

Then, for rules of the form:

$$\frac{\Gamma \rightarrow (P \leq n \wedge Perm(P)) \supset \alpha, \Delta}{\Gamma \rightarrow (\forall P \leq n)||\alpha||_\sigma, \Delta} \tag{5.19}$$

we assume inductively that a proof of the translated top sequent $(||\Gamma||_\sigma, \rightarrow ||\Delta||_\sigma, (||P \leq n||_\sigma \wedge ||Perm(P)||_\sigma) \supset ||\alpha||_\sigma)$ exists, and that this proof is bounded by some polynomial $p(|\sigma|)$. Note that $||P \leq n||_\sigma$ always evaluates to 1 or 0 immediately, depending on σ, so we do not write it in the following proof for the sake of readability. Then, we construct a proof of the bottom sequent as follows:

$$\begin{array}{c} \pi_1 \\ \vdots \\ \dfrac{||\Gamma||_\sigma, \rightarrow ||\Delta||_\sigma, ||Perm(P)||_\sigma \supset ||\alpha||_\sigma}{||\Gamma||_\sigma \rightarrow ||\Delta||_\sigma, (\forall ab)(||Perm(P)||_\sigma \supset ||\alpha||_\sigma)} \; \forall - right \end{array}$$

Since the original proof was a valid $\forall P$**LA** proof, we know that P did not appear free in the bottom sequent, and that therefore the variables in P (a, b, etc.) do not appear in Γ or Δ, so H's restriction on $\forall$-right is satisfied and we are able to apply the rule. We repeat this step once for each transposition that we need to introduce ($O(n^2)$ of them), to obtain the translated sequent:

$$||\Gamma||_\sigma \rightarrow ||\Delta||_\sigma, (\forall P)||Perm(P) \supset \alpha||_\sigma \tag{5.20}$$

where again $(\forall P)$ is a shorthand for $(\forall P_{11}P_{12})(\forall P_{11}P_{13}) \ldots (\forall P_{n-1n}P_{nn})$. n is the number of variables being permuted, and also the dimension of P under σ. Therefore, the number of quantifiers introduced at any one step is bounded by a polynomial in $|\sigma|$ (recall that $|\sigma|$ is the maximum value σ assigns to any variable). Therefore, if q is the polynomial that bounds the translation of $\forall P$**LA** formulas, then the degree of p (from the inductive hypothesis) must be at least two higher than the degree of q.

It is at this point that the definition of H given in [9] proves to be inadequate. The unrestricted introduction rule would have required a second upper sequent, which would have been generated by applying the permutation rule to the original upper sequent. In a treelike proof, we would need to duplicate π_1 to do this, causing a potentially exponential increase in proof size.

The $\forall$-left rule proceeds in the same way as above introducing $O(n^2)$ transposition quantifiers.

Now we can see the real value of the canonical representation of a permutation by transposition quantifiers (from 4.2). Although n transposition quantifiers are sufficient to represent a particular permutation, that formulation is able to express *any* permutation. This means that the same translation can be performed (with only a polynomial increase in proof size) regardless of the particular P involved.

After this, we have an H_1^* proof of the bottom sequent. In total, the H_1^* proof is bounded in size by a polynomial, because each step in the $\forall P$**LA** proof takes at most polynomially many steps in the H_1^* proof. □

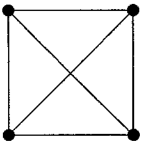
While interesting in its own right, demonstrating that H_1^* can prove translations of $\forall P$**LA** does not seem to bring us closer to the desired result, namely that H_1^* can p-simulate Extended PK. However, by introducing a third system, the Hajós Calculus, we can show just that.

Chapter 6

The Hajós Calculus

The next stage in the simulation is to link $\forall P\mathbf{LA}$ to Extended Frege. We do this via another seemingly unrelated system, the Hajós Calculus.

A graph is k-colourable if there exists a way to assign each of its nodes one of k different colours such that no two adjacent nodes receive the same colour. For $k \geq 3$, this problem is NP complete (see, for example, [11]). The Hajós Calculus is a procedure for constructing non-k-colourable undirected graphs and was first introduced in [7]. It has a single axiom, K_{k+1}, the complete graph with $k+1$ vertices. We are interested in the case where $k = 3$. Here is the graph K_4, and its associated adjacency matrix:

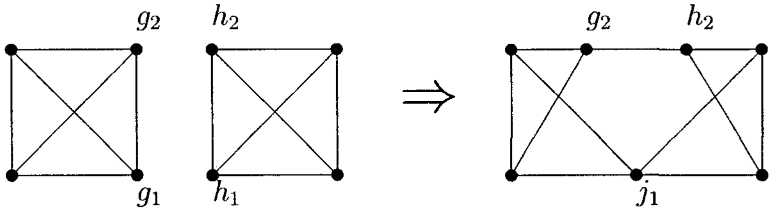
$K_4 =$ 

$$A_{K_4} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

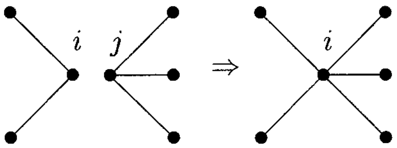
From this axiom, the Hajós Calculus uses three rules for constructing larger non-3-colourable graphs. We will deal only with 'simple' graphs, that have no multiple edges, and no edges from a vertex back to itself. The three rules are:

Addition: Given any graph, G, add any number of nodes and edges.

Join: Given two graphs G and H, with edges (g_1, g_2) and (h_1, h_2), construct a new graph J by taking the union $G \cup H$, but with those two edges removed, a new edge (g_2, h_2) added, and the vertices g_1 and h_1 contracted into a single new vertex j_1. For example:



Contraction: Given a graph G, take any two non-adjacent vertices i and j. Remove j, and add edges (i, h) for every node h that was adjacent to j (though do not add duplicate edges).



A *derivation* in the Hajós Calculus is a sequence of graphs $\{G_1, G_2, \ldots, G_n\}$ in which each G_k is either an axiom (i.e. $G_k = K_4$), or else follows from one or two previous graphs (G_i where $i < k$) by one of the three rules given above. It is natural to ask what kinds of graphs can be generated with such a derivation.

Lemma 23 *The Hajós Calculus is sound. That is, any graph that can be derived using the Hajós Calculus is non-3-colourable.*

PROOF: K_4 is clearly non-3-colourable. The Addition rule is sound, because any graph with a non-3-colourable subgraph is non-3-colourable.

For the Join rule, assume that G and H are non-3-colourable, but that a 3-colouring $f : V \longrightarrow \{\text{Red, Blue, Yellow}\}$ of J exists. Then we can produce a 3-colouring f' of either G or H as follows. Firstly, if the colours of j_1, g_2, and h_2 are all different, then we let f' be the same as f, but with $f'(g_1) = f'(h_1) = f(j_1)$. In this case, f' 3-colours both G and H. Otherwise, we know that g_2 and h_2 are different colours, and that j_1 is the same colour as one of them. If $f(j_1) \neq f(g_2)$, then let $f'(g_1) = f(j_1)$, and then f' 3-colours G. If $f(j_1) \neq f(h_2)$, we let $f'(h_1) = f(g)$ and then f' 3-colours H.

For the Contraction rule, assume that G is non-3-colourable, but let f be a 3-colouring of the resulting graph. Since i and j_1 are non-adjacent in G, we can colour G with $f' = f$ except $f'(i) = f'(j_1) = f(i)$.

So, since the rules preserve non-3-colourability, and the axiom K_4 is non-3-colourable, the Hajós Calculus produces only non-3-colourable graphs. □

The Hajós Calculus is also complete, that is, a derivation exists for *any* non-3-colourable graph. This is unnecessary for our purposes, so we do not prove it here. For more information see [7].

6.1 The Hajós Calculus and Extended PK

Since graph 3-colourability is NP complete, there must exist graphs for which only exponential sized HC deriviations exist, unless NP = coNP. Therefore, it is expected that no polynomial bounds exist for the Hajós Calculus. Pitassi and Urquhart showed further that the HC is polynomially bounded if and only if Extended Frege proof systems are polynomially bounded. This proof forms a key link in our proof that H_1^* is p-equivalent to Extended PK, and so we outline it here. For more thorough treatment see [10].

The basic idea behind the simulation is that both 3-SAT and 3-COL are NP complete problems, and there exists some polytime reduction back and forth between them, that is, given a formula Φ, we obtain a graph G_Φ such that G_Φ is 3-colourable if and only if Φ is satisfiable (and vice versa). Since Extended Frege/PK constructs formulas, and the Hajós Calculus builds graphs, we can look for a way to simulate one with the other. (Of course, a similar argument could be advanced to show that basic Frege/PK p-simulates the Hajós Calculus, but it turns out that the additional power of the renaming rule is instrumental in providing a simple proof of equivalence). In their paper, Pitassi and Urquhart use a different kind of Frege system which is based on resolution. Resolution systems operate only on formulas in Conjunctive Normal Form, and are used to construct unsatisfiable sets of clauses. On the other hand, the systems presented in Chapter 2 were used only to construct valid formulas. However, Cook and Reckhow showed the equivalence of all Frege systems, which means that all these results apply to the systems presented in Chapter 2, although a direct simulation is not likely to be as elegant.

Converting Graphs to Formulas

Given a graph, we would like to construct a propositional formula that asserts that the graph is 3-colourable. Let $G = (E, V)$ be a graph, and let R_i, B_i and Y_i be propositional variables that assert that the node i is red, blue, and yellow respectively. Then the following formula asserts that each vertex gets exactly one colour:

$$\bigwedge_{i \in V} (R_i \supset (\bar{B}_i \wedge \bar{Y}_i) \wedge (B_i \supset (\bar{R}_i \wedge \bar{Y}_i) \wedge (Y_i \supset (\bar{R}_i \wedge \bar{B}_i) \wedge (R_i \vee B_i \vee Y_i) \tag{6.1}$$

and this one asserts that no two neighbouring vertices get the same colour:

$$\bigwedge_{i,j \ (i,j) \in E} (\bar{R}_i \vee \bar{R}_j) \wedge (\bar{B}_i \vee \bar{B}_j) \wedge (\bar{Y}_i \vee \bar{Y}_j) \tag{6.2}$$

and so then (6.1)$\wedge$ (6.2) is satisfiable iff the underlying graph is 3-colourable.

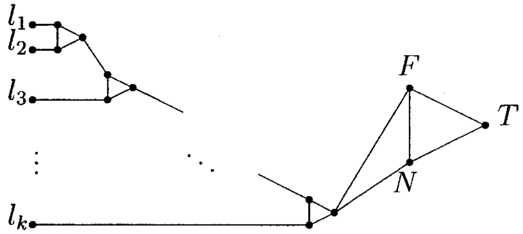
Simulating HC with Extended Frege/PK

Pitassi and Urquhart[10] show that a basic Frege system can p-simulate the **Join** and **Addition** rules. This can be seen by noting first that the formula representation of K_4 has a constant number of symbols. Then, for each application of the **Addition** rule, for each added vertex we weaken in a copy of (6.1), and for each added edge we weaken in a copy of (6.2). The **Join** rule is handled in a similar fashion.

It is believed that Basic Frege/PK systems can not easily simulate the **Contraction** rule, but by adding the renaming rule we are able to do so. This is because the contraction rule is a kind of renaming. If vertices i and j are the ones that are contracted, we can simulate that with a constant number of applications of the renaming rule, where we rename R_i to R_j, B_i to B_j, and so forth. Since Renaming PK is equivalent to Extended PK (see Theorem 8), we obtain the desired result.

Converting Formulas to Graphs

Since for now we are working with a resolution Frege system, we are free to assume that all the formulas we need to convert are in CNF. Then we construct a graph as follows. For each clause, add the following subgraph:



where each l_i is a literal. For each variable x, we make a pair of vertices x and $\bar{x}$, and make an edge between them. It is these nodes that take the place of the l_i in the above subgraph. Then, if the graph is 3-colourable, at least one of the literals in each clause subgraph must have the same colour as the vertex T. Then, any assignment that assigns true to every literal with the same colour as T must satisfy the associated formula (because it assigns true to at least one literal per clause).

Simulating Extended Frege/PK with the Hajós Calculus

Pitassi and Urquhart[10] show this result for a Renaming Frege system based on resolution of clauses. The basic idea is that the shortest possible Renaming Frege resolution refutation is the one that refutes $p \wedge \neg p$. The corresponding graph for this formula contains a K_4 subgraph over the vertices F, N, p and $\bar{p}$, and therefore can be efficiently constructed by the Hajós Calculus from an axiom and a constant number of applications of the **Addition** rule. Then, graphs corresponding to larger unsatisfiable CNF formulas can be simulated by simulating the individual rules of the Renaming system.

We are now nearly finished with the proof of simulation. We have shown that valid $\forall P\mathbf{LA}$ sequents translate into families of H_1^* tautologies with only a polynomial increase in sequent size and proof length. Seperately, we have outlined a proof that the Hajós Calculus is p-equivalent to Extended PK. To complete the chain, we need to show that Hajós Calculus derivations can be represented in $\forall P\mathbf{LA}$.

Chapter 7

The Hajós Calculus, $\forall P$**LA**, and H_1^*

In this chapter our aim is to show that Hajós Calculus deriviations can be expressed succintly in $\forall P$**LA**. To do this, we give a way to represent individual graphs in $\forall P$**LA** formulas, and then show that chains of graphs can likewise be represented.

$\exists P$**LA** and $\forall P$**LA** are well suited to expressing graph theoretic properties. For example, a graph is k-colourable if and only if it can be permuted into a graph which has a k-co-clique (k subsets of edges such that all the nodes in a given subset have no edges between them). We can express this with the following formula of $\exists P$**LA**:

$$(\exists P \le r(A))(\exists i_1, i_2, \ldots, i_k \le r(A)) \left(PAP^t = \left[\begin{array}{c|c|c|c} 0_{i_1} & * & \cdots & * \\ \hline * & 0_{i_2} & \cdots & * \\ \hline \vdots & \ddots & \ddots & \vdots \\ \hline * & \cdots & * & 0_{i_k} \end{array}\right] \right) \tag{7.1}$$

A is the adjacency matrix of the graph in question, where 0_n is the $n \times n$ matrix of all zeros, and where $*$ can be anything. Therefore, A represents a k-colourable graph if and only if (7.1) is a true formula of $\exists P$**LA**.

The negation of this formula likewise expresses non-k-colourability. In particular, we are interested in the case where $k = 3$. Let **Non-3-Col** be the negation of (7.1) with $k = 3$:

$$(\forall P \leq r(A))(\forall i_1, i_2, i_3 \leq r(A)) \left(PAP^t \neq \left[\begin{array}{c|c|c} 0_{i_1} & * & * \\ \hline * & 0_{i_2} & * \\ \hline * & * & 0_{i_3} \end{array} \right] \right) \tag{7.2}$$

7.1 $\forall P$LA proves the soundness of the Hajós calculus

Clearly, (7.2) is a formula of $\forall P$LA. It is also interesting, because, combined with the ability to represent graphs as matrices, it gives us a way to assert (in $\forall P$LA) that a derivation in the Hajós Calculus is correct.

Lemma 24 *$\forall P$LA proves the soundness of the rules of the Hajós Calculus.*

PROOF: We can show that $\forall P$**LA** can prove the soundness of the Hajós Calculus (HC) by showing that it can prove the soundness of the HC rules, and that it can prove that the axioms of the HC are non-3-colourable graphs. Since the HC has only one axiom K_4, this second part amounts to showing that $\forall P$**LA** $\vdash$ **Non-3-Col**(K_4).

We can do this by proving a sequent that asserts that none of the 16 valid permutations of K_4 is equal to a matrix of the correct form. K_4 is:

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \tag{7.3}$$

and if K_4 were 3-colourable, one of its permutations would equal this matrix:

$$\begin{bmatrix} 0 & 0 & * & * \\ 0 & 0 & * & * \\ * & * & 0 & * \\ * & * & * & 0 \end{bmatrix} \tag{7.4}$$

where $*$ is 0 or 1. So we can prove the 16 sequents of the form:

$$\rightarrow K_4^i \neq (7.4) \tag{7.5}$$

where K_4^i denotes one of the permutations of K_4. Then, separately prove the sequent:

$$\bigwedge_{i=1...16} K_4^i \neq (7.4) \to \textbf{Non-3-Col}(K_4) \tag{7.6}$$

and from here we cut 16 times to obtain the desired sequent.

We must also show that $\forall P\mathbf{LA}$ can prove the soundness of HC's three rules. Once more, these are:

1) **Addition:** Add vertices and/or edges to a graph.

2) **Join:** Take two graphs, G and H, with (g_1, g_2) and (h_1, h_2) be edges in G and H respectively. Construct J by removing those two edges, adding the edge (g_2, h_2) and contracting the nodes g_1 and h_1 into a single vertex (j_1).

3) **Contraction:** Take any two non-adjacent vertices i and j. Remove j, and add edges (i, h) for every node h which was adjacent to j where necessary (that is, do not add duplicate edges).

To show that $\forall P\mathbf{LA}$ proves the soundness of the join rule, assume inductively that $\forall P\mathbf{LA} \vdash \textbf{Non-3-Col}(A)$. Then let A' be the graph obtained from A by adding extra vertices and/or edges (i.e. by applying the addition rule). An edge in A implies an edge in A', or, in $\forall P\mathbf{LA}$:

$$r(A) \leq r(A') \wedge \forall(i, j \leq r(A))[e(i, j, A) \supset e(i, j, A')] \tag{7.7}$$

So A' contains a partial copy of A (with at least all the edges of A, possibly more) in its upper-left corner. We can therefore derive the sequent:

$$\textbf{Non-3-Col}(A) \to \textbf{Non-3-Col}(A') \tag{7.8}$$

in $\forall P\mathbf{LA}$ and cut to obtain the conclusion we want.

Similarly, assume we have proofs of **Non-3-Col**(A) and **Non-3-Col**(B). Let (i_1, i_2) and (j_1, j_2) be edges in those graphs, respectively, and let C be the graph obtained by joining vertex i_1 to vertex j_1, and merging i_2 and j_2. C can therefore be described as follows:

$$C = \left[\begin{array}{cc|c} A[i_2|i_2] & Q & D_1 \\ Q^t & B[j_2|j_2] & D_2 \\ \hline D_1^t & D_2^t & 0 \end{array}\right] \tag{7.9}$$

where $A[i|j]$ denotes A with the i^{th} row and j^{th} column removed. The last row and column of this matrix corresponds to the new vertex that was created by merging i_2 and j_2. D_1 and D_2 are column vectors. D_1 has a 1 in the k^{th} row if and only if $e(A, i_2, k) = 1$, likewise D_2 has a 1 in the k^{th} row if and only if $e(B, j_2, k) = 1$. Q is the matrix of all zeros, except for a single 1 in the (i_1, j_1) position (because the edge added between those two vertices is the only new one between the two subgraphs).

Then, because A and B appear on the diagonal of C, we can derive the sequent **Non-3-Col**(A) $\wedge$ **Non-3-Col**(B) $\rightarrow$ **Non-3-Col**(C).

The contraction case is similar, except here we have replaced two vertices from the same graph with one new one. If i and j are the contracted vertices, then this matrix represents the resulting graph A':

$$A' = \left[\begin{array}{c|c} A[i|i][j|j] & D \\ \hline D^t & 0 \end{array}\right] \tag{7.10}$$

where $A[i|j][k|l]$ is A with rows i and k, and columns j and l removed. D is again a column vector with $D[k] = 1$ iff $e(A, i, k) = 1 \vee e(A, j, k) = 1$. Then in a similar way, we can derive **Non-3-Col**(A) $\rightarrow$ **Non-3-Col**(A'). □

Encoding Hajós deriviations in ∀PLA

We can then show that ∀**PLA** can prove the soundness of the Hajós Calculus as a whole. A useful tool in doing so is the idea of encoding an entire HC deriviation in a single matrix. Let $A_1, A_2, A_3, \ldots, A_n$ be a HC derivation of some non-3-colourable graph A_n, where each A_j is either K_4, or else follows from one or two $A_i, i < j$.

Then, pad each of these matrices with rows and columns of all zeros, until all the A_i are of the same dimensions. Next, embed them along the main diagonal of a new matrix, Y. We can then write:

$$Y = \begin{bmatrix} A_1 & 0 & \cdots & \cdots & 0 \\ 0 & A_2 & 0 & \cdots & 0 \\ 0 & 0 & A_3 & \cdots & 0 \\ 0 & \cdots & \cdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & A_n \end{bmatrix} \tag{7.11}$$

and let **HC**(Y) be a formula stating that Y is indeed a valid HC derivation. Such a formula can be stated as follows:

$$\begin{array}{lrl} \forall_{i \leq n}(A_i = K_4) \vee & [(\exists j_1, j_2 < i) & \mathbf{Addition}(A_i, A_{j_1}) \\ & \vee & \mathbf{Join}(A_i, A_{j_1}, A_{j_2}) \\ & \vee & \mathbf{Contraction}(A_i, A_{j_1}))] \end{array} \tag{7.12}$$

where **Addition**, **Join**, and **Contraction** are formulas expressing that A_i is formed from each of the rules of the Hajós Calculus.

For example, one way of formulating **Addition**(B, A) would be:

$$(r(A) \leq r(B) \wedge c(A) \leq r(B)) \wedge \bigwedge_{\substack{i=1 \ldots r(A), i \neq i_2 \\ j=1 \ldots c(A), j \neq j_2}} e(A, i, j) \supset e(B, i, j) \tag{7.13}$$

Join(C, A, B) is (assuming the vertices are named as above):

$$\begin{array}{ll} & (r(C) = r(A) + r(B) - 1) \wedge (c(C) = c(A) + c(B) - 1) \wedge \\ \bigwedge_{\substack{i=1 \ldots r(A), i \neq i_2 \\ j=1 \ldots c(A), j \neq i_2}} & e(C, i, j) \equiv e(A, i, j) \wedge \\ \bigwedge_{\substack{i=1 \ldots r(B), i \neq j_2 \\ j=1 \ldots c(B), j \neq j_2}} & e(C, r(A) + i, c(A) + j) \equiv e(B, i, j) \end{array} \tag{7.14}$$

Theorem 25 *∀PLA proves the soundness of the Hajós Calculus.*

PROOF: To prove that the Hajós Calculus is sound, $\forall P$**LA** must prove that any graph produced by a Hajós Calculus derivation is indeed non-3-colourable. In other words, if Y is (7.11) then $\forall P$**LA** can prove the sequent:

$$\mathbf{HC}(Y) \rightarrow \mathbf{Non\text{-}3\text{-}Col}(A_n) \tag{7.15}$$

We can prove this by induction on k, the number of intermediate matrices encoded in Y. If $k = 1$, ($Y = [A_1]$), then A_1 must encode K_4, so the base case is simple. The inductive step follows from the previous lemma, because if $Y = [A_1 A_2 \ldots A_i \ldots A_j \ldots A_n]$, and we assume that a proof exists for $(\forall k' < n)\mathbf{HC}(Y) \rightarrow \mathbf{Non\text{-}3\text{-}Col}(A_{k'})$ then the lemma gives us either $\mathbf{Non\text{-}3\text{-}Col}(A_i) \wedge \mathbf{Non\text{-}3\text{-}Col}(A_j) \rightarrow \mathbf{Non\text{-}3\text{-}Col}(A_n)$ or $\mathbf{Non\text{-}3\text{-}Col}(A_i) \rightarrow \mathbf{Non\text{-}3\text{-}Col}(A_n)$ for some $i, j < n$ depending on which rule was used to derive A_n. □

Since Hajós derivations can be shown to be correct in $\forall P$**LA**, we can, by the results of Chapter 5, translate (7.11) into a family of H_1^* tautologies with only a polynomial increase in size. This completes our proof that Extended PK can be p-simulated by H_1^*, and together with Theorem 18 allows us to state the main result of this thesis:

Theorem 26 *Extended PK and H_1^* are p-equivalent*

PROOF: Theorem 18 shows that Extended PK p-simulates H_1^*. For the other direction, suppose there is a sequent S with an associated Extended PK proof π_e of S. There is a polytime function f such that $f(\pi_e)$ is an H_1^* proof of S.

This is a several stage process, following from chapters 5, 6, and 7. Firstly, let $\neg S$ be the negation of the formula equivalent to S (namely $\neg(\bigwedge \Gamma \supset \bigvee \Delta)$), and construct the corresponding graph $G_{\neg S}$ as given in chapter 6. This graph is non-3-colourable if and only if $\neg S$ is unsatisfiable. Pitassi and Urquhart [10] show that there is a polytime function that makes this conversion.

Then, as shown above, if $Y_{\neg S}$ encodes a Hajós Calculus deriviation of $G_{\neg S}$ then there are $\forall P$**LA** proofs of $\rightarrow HC(Y_{\neg S})$ and $HC(Y_{\neg S}) \rightarrow \mathbf{Non\text{-}3\text{-}Col}(G_{\neg S})$. From these, we obtain a $\forall P$**LA** proof of $\rightarrow \mathbf{Non\text{-}3\text{-}Col}(G_{\neg S})$.

In chapter 5, building on an idea from [13], we have shown that $\forall P$**LA** translates into short H_1^* proofs. What this means is that there is a polytime function that takes a $\forall P$**LA** sequent and an object assignment σ and produces an H_1^* proof of the translated sequent which is polysized in $|\sigma|$. There must therefore be short H_1^* proofs of:

$$\rightarrow ||\textbf{Non-3-Col}(G_{\neg S})||_\sigma$$

This does not give us precisely what we need, because we are looking for an H_1^* proof of S. However, Pitassi and Urquhart also give a procedure for transforming non-3-colourable graphs into formulas. H_1^* could efficiently prove this procedure is correct, and therefore prove the sequent

$$||\textbf{Non-3-Col}(G_{\neg S})||_\sigma \rightarrow S$$

from which we cut to obtain the H_1^* proof of S that we want. This multistep procedure constitutes the polytime formula that we need to convert Extended PK proofs into H_1^* proofs, and completes the main result of this thesis. □

Chapter 8

Conclusion

Therefore, we arrive at the interesting conclusion that Extended PK (and hence Extended Frege) is p-equivalent to H_1^*. We showed one direction, that Extended PK can p-simulate H_1^*, directly. The other direction was shown indirectly, by showing that H_1^* can efficiently prove translations of true formulas from $\forall P\mathbf{LA}$, and then repeating results from Pitassi[10] and from Soltys[13] to show that this implies that H_1^* can in fact p-simulate Extended Frege. The ramifications of this are powerful. A single block of transposition quantifiers possesses the same power as a single block of classical quantifiers, and the same power as being able to substitute a formula for a variable in a propositional proof. Futhermore, H_1^* is able to capture all of polynomial time reasoning, and is another propositional proof system that corresponds to P/poly (decision problems solvable with non-uniform polysize circuits).

During the course of the proof, several interesting points occurred which lead naturally to further work in this field. Firstly, does a direct simulation of Extended PK by H_1^* exist? This situation is analgous to the proof of equivalence of Extended PK and Substitution PK mentioned in Chapter 2. It was only after Dowd[5] gave an indirect proof of the equivalence that Krajíček[8] was able to give a direct proof. Krajíček's proof is modelled on Dowd's, and is nearly indecipherable without knowledge of the indirect simulation. A direct simulation based on our results may be possible.

Secondly, H and G appear to be very similar systems. In Chapter 3 we gave a proof (from [8]) that G_1^* and Extended PK are p-equivalent. Similarly, we showed that H_1^* is p-equivalent to Extended PK provided the definition of H given in Chapter 4 is used. Treelikeness appears to be unpreservable if H is defined as in [9]. It would be interesting to see if the other, unrestricted definition of H could p-simulate Extended

PK and maintain treelikeness.

Thirdly, the link between H and G is open to further exploration. We have shown that H_1^* and G_1^* are p-equivalent. The natural extension to this is to ask whether H_i and G_i are equivalent for all i. If so, we would know that H and G were equivalent propositional proof systems.

Overall, H is a small but significant contribution to propositional proof complexity. In addition to being p-equivalent to Extended PK (and so to Substitution PK, Renaming PK, and TF PK), it provides a natural propositional representation of the first-order logical theory $\forall P\mathbf{LA}$, and is therefore a good system to use for low-level reasoning about graphs or algebraic structures.

Chapter 9

Appendix

First we show that formulas of the form $B, \alpha(1) \longrightarrow \alpha(B)$ have short PK proofs. Sequents of the form $B, \alpha(B) \longrightarrow \alpha(1)$ can be shown to have short proofs in almost exactly the same fashion. Some interesting special cases are when B is a single variable, or when B is a list of formulas. The former case is simply the base case shown below. The induction required to prove the latter case is slightly more complicated, but not much so.

Lemma 27 *For formulas B and α there are $O(|B||\alpha|^2)$ sized proofs in TF PK of $B, \alpha(1) \longrightarrow \alpha(B)$ and $\neg B, \alpha(0) \longrightarrow \alpha(B)$*

PROOF: The proof is by induction on the number of logical connectives ($\vee$, $\wedge$, and$\neg$) in α. We show the case for $B, \alpha(1) \longrightarrow \alpha(B)$; the $\neg B, \alpha(0) \longrightarrow \alpha(B)$ proceeds analogously.

Base Case: α has no logical connectives (i.e. $\alpha(x) \overset{synt.}{=} x$, for some x). Then:

$$\frac{\dfrac{x \longrightarrow x}{1 \longrightarrow 1}}{B, 1 \longrightarrow 1}$$

which has size $|B| + 7$, which is $O(|B||1^2|) = O(|B||\alpha^2|)$.

Induction Hypothesis: Let α be any formula with k or fewer connectives and assume that $B, \alpha(1) \longrightarrow \alpha(B)$ has a TF PK proof of size at most $C \cdot |B||\alpha^2|$. The actual value of C will become apparent later.

Induction Step: Let α be a formula with $k+1$ logical connectives. There are three cases, one for each possible outermost connective of α.

Case 1) $\vee$. Then α is of the form $\gamma \vee \delta$ where there are a total of k logical connectives in γ and δ. Therefore, by the induction hypothesis, they have proofs π_γ and π_δ of length at most $C \cdot |B||\gamma|^2$ and $C \cdot |B||\delta|^2$ respectively. From this:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi_\gamma\\ B,\gamma(1)\rightarrow\gamma(B)\end{array}}{B,\gamma(1)\rightarrow\gamma(B),\delta(B)}\,\textit{weak.}\qquad \dfrac{\dfrac{\begin{array}{c}\vdots\,\pi_\delta\\ B,\delta(1)\rightarrow\delta(B)\end{array}}{B,\delta(1)\rightarrow\delta(B),\gamma(B)}\,\textit{weak.}}{B,\delta(1)\rightarrow\gamma(B),\delta(B)}\,\textit{exch.}}{B,\gamma(1)\vee\delta(1)\rightarrow\gamma(B),\delta(B)}\,\vee-\textit{left}}{B,\gamma(1)\vee\delta(1)\rightarrow\gamma(B)\vee\delta(B)}\,\vee-\textit{right}$$

Which, for C sufficiently high, has size at most:

$$\begin{aligned}
& |\pi_\gamma| + |\pi_\delta| + 7|B| + 6|\gamma||B| + 6|\delta||B| + 4|\gamma| + 5|\delta| + 3\\
\leq\ & C\cdot|B||\gamma|^2 + C\cdot|B||\delta|^2 + 7|B|(1+|\gamma|+|\delta|) + 5(1+|\gamma|+|\delta|)\\
=\ & C\cdot|B||\gamma|^2 + C\cdot|B||\delta|^2 + (7|B|+5)((1+|\gamma|+|\delta|)\\
\leq\ & C\cdot|B||\alpha^2|
\end{aligned}$$

Case 2) $\wedge$. Then α is of the form $\gamma \wedge \delta$, and we proceed as above:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi_\gamma\\ B,\gamma(1)\rightarrow\gamma(B)\end{array}}{\delta(1),B,\gamma(1)\rightarrow\gamma(B)}\,\textit{weak.}}{B,\delta(1),\gamma(1)\rightarrow\gamma(B)}\,\textit{exch.}}{B,\gamma(1),\delta(1)\rightarrow\gamma(B)}\,\textit{exch.}\qquad \dfrac{\dfrac{\begin{array}{c}\vdots\,\pi_\gamma\\ B,\delta(1)\rightarrow\delta(B)\end{array}}{\gamma(1),B,\delta(1)\rightarrow\delta(B)}\,\textit{weak.}}{B,\gamma(1),\delta(1)\rightarrow\delta(B)}\,\textit{exch.}}{B,\gamma(1),\delta(1)\rightarrow\gamma(B)\wedge\delta(B)}\,\wedge-\textit{right}}{B,\gamma(1)\wedge\delta(1)\rightarrow\gamma(B)\wedge\delta(B)}\,\wedge-\textit{left}$$

Which has size at most:

$$\begin{array}{rl}
& |\pi_\gamma| + |\pi_\delta| + 9|B| + 6|B||\gamma| + 5|B||\delta| + 8|\gamma| + 8|\delta| + 3 \\
\leq & C \cdot |B||\gamma|^2 + C \cdot |B||\delta|^2 + 9|B|(1 + |\gamma| + |\delta|) + 8(1 + |\gamma| + |\delta|) \\
\leq & C \cdot |B|(1 + |\gamma| + |\delta|)^2 \\
= & C \cdot |B||\alpha|^2
\end{array}$$

Case 3. $\neg$. If the outermost connective is $\neg$, then either i) α is of the form $\neg\neg\gamma$, or ii) α is of the form $\neg(\gamma \circ \delta)$ where $\circ$ is either $\wedge$ or $\vee$.

i) If α is $\neg\neg\gamma$, then γ has only $k - 1$ logical connectives, and, as above, there is proof of it with size at most $C \cdot |B||\gamma|^2$. We then obtain:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overset{\vdots\ \pi_\gamma}{B, \gamma(1) \to \gamma(B)}}{B \to \neg\gamma(1), \gamma(B)}\ \neg - right}{B, \neg\neg\gamma(1) \to \gamma(B)}\ \neg - left}{B, \neg\neg\gamma(1), \neg\gamma(B) \to}\ \neg - left}{B, \neg\neg\gamma(1) \to \neg\neg\gamma(B)}\ \neg - right$$

This has size at most:

$$\begin{array}{rl}
& |\pi_\gamma| + 5|B| + 5|\gamma||B| + 5|\gamma| + 10 \\
\leq & C \cdot |B||\gamma|^2 + 5|B|(1 + |\gamma|) + 10(2 + |\gamma|) \\
\leq & C \cdot |B||\gamma|^2 + (5|B| + 10)(2 + |\gamma|) \\
\leq & C \cdot |B|(2 + |\gamma|)^2 \\
= & C \cdot |B||\alpha|^2
\end{array}$$

ii) Let $\circ$ be $\vee$ (the case where $\circ$ is $\wedge$ proceeds similarly). Then α is $\neg(\gamma \vee \delta)$. Since α has $k + 1$ logical connectives, γ and δ each have fewer than k. Therefore, $\neg\gamma$ and $\neg\delta$ have k or fewer connectives and by the induction hypothesis, we have 0/1 PK proofs of $B, \neg\gamma(1) \to \neg\gamma(B)$ and $B, \neg\delta(1) \to \neg\delta(B)$. We denote them by $\pi_{\neg\gamma}$ and $\pi_{\neg\delta}$, and observe that they have size at most $C \cdot |B|(|\gamma| + 1)^2$ and $C \cdot |B|(|\delta| + 1)^2$ respectively. Then:

$$
\begin{prooftree}
\AxiomC{$\vdots\ \pi_{\neg\gamma}$}
\noLine
\UnaryInfC{$B, \neg\gamma(1) \rightarrow \neg\gamma(B)$}
\UnaryInfC{$B \rightarrow \neg\neg\gamma(1), \neg\gamma(B)$}
\RightLabel{$weak.$}
\UnaryInfC{$B \rightarrow \neg\neg\gamma(1), \neg\gamma(B), \gamma$}
\UnaryInfC{$B \rightarrow \neg\neg\gamma(1), \gamma, \neg\gamma(B)$}
\AxiomC{$\gamma(1) \rightarrow \gamma(1)$}
\RightLabel{$\neg - right$}
\UnaryInfC{$\rightarrow \neg\gamma(1), \gamma(1)$}
\RightLabel{$\neg - left$}
\UnaryInfC{$\neg\neg\gamma(1) \rightarrow \gamma(1)$}
\RightLabel{$weak.$}
\UnaryInfC{$B, \neg\neg\gamma(1) \rightarrow \gamma(1)$}
\RightLabel{$weak.$}
\UnaryInfC{$B, \neg\neg\gamma(1) \rightarrow \gamma(1), \neg\gamma(B)$}
\RightLabel{cut}
\BinaryInfC{$B \rightarrow \gamma(1), \neg\gamma(B)$}
\RightLabel{$exch.$}
\UnaryInfC{$B \rightarrow \neg\gamma(B), \gamma(1)$}
\RightLabel{$\neg - left$}
\UnaryInfC{$B, \neg\neg\gamma(B) \rightarrow \gamma(1)$}
\RightLabel{$weak.$}
\UnaryInfC{$\gamma(B), B, \neg\neg\gamma(B) \rightarrow \gamma(1)$}
\RightLabel{$exch$}
\UnaryInfC{$B, \gamma(B), \neg\neg\gamma(B) \rightarrow \gamma(1)$}
\end{prooftree}
$$

and separately derive:

$$
\begin{prooftree}
\AxiomC{$\gamma(B) \rightarrow \gamma(B)$}
\RightLabel{$\neg - left$}
\UnaryInfC{$\gamma(B), \neg\gamma(B) \rightarrow$}
\RightLabel{$\neg - right$}
\UnaryInfC{$\gamma(B) \rightarrow \neg\neg\gamma(B)$}
\UnaryInfC{$B, \gamma(B) \rightarrow \neg\neg\gamma(B)$}
\RightLabel{$weak.$}
\UnaryInfC{$B, \gamma(B) \rightarrow \neg\neg\gamma(B), \gamma(1)$}
\end{prooftree}
$$

and then cut to obtain the desired sequent:

$$
\begin{prooftree}
\AxiomC{$B, \gamma(B), \neg\neg\gamma(B) \rightarrow \gamma(1) B, \gamma(B) \rightarrow \neg\neg\gamma(B), \gamma(1)$}
\RightLabel{cut}
\UnaryInfC{$B, \gamma(B) \rightarrow \gamma(1)$}
\end{prooftree}
$$

This has size at most:

$$|\pi_{\neg\gamma}| + 14|B| + 21|\gamma| + 23|\gamma||B| + 34$$

Likewise, from $B, \neg\delta(1) \rightarrow \neg\delta(B)$ we have a proof of $B, \delta(B) \rightarrow \delta(1)$ of size:

$$|\pi_{\neg\delta}| + 14|B| + 21|\delta| + 23|\delta||B| + 34$$

From here, we can obtain a proof of α as follows:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\ \neg\neg\gamma - elim.\\ B,\gamma(B)\rightarrow\gamma(1)\end{array}}{B,\gamma(B)\rightarrow\gamma(1),\delta(1)}\,weak.\qquad \dfrac{\begin{array}{c}\vdots\ \neg\neg\delta - elim.\\ B,\delta(B)\rightarrow\delta(1)\end{array}}{B,\delta(B)\rightarrow\gamma(1),\delta(1)}\,exch.}{B,\gamma(B)\vee\delta(B)\rightarrow\gamma(1),\delta(1)}\,\vee - left}{B,\gamma(B)\vee\delta(B)\rightarrow\gamma(1)\vee\delta(1)}\,\vee - right}{B\rightarrow\neg(\gamma(B)\vee\delta(B)),\gamma(1)\vee\delta(1)}\,\neg - right}{B\rightarrow\gamma(1)\vee\delta(1),\neg(\gamma(B)\vee\delta(B))}\,exch.}{B,\neg(\gamma(1)\vee\delta(1))\rightarrow\neg(\gamma(B)\vee\delta(B))}\,\neg - left
$$

With size at most:

$$
\begin{aligned}
& |\pi_{\neg\gamma}| + 14|B| + 21|\gamma| + 23|\gamma||B| + 34\\
+\ & |\pi_{\neg\delta}| + 14|B| + 21|\delta| + 23|\delta||B| + 34\\
+\ & 10|B| + 7|\gamma||B| + 7|\delta||B| + 8|\gamma| + 8|\delta| + 13\\
=\ & |\pi_{\neg\gamma}| + |\pi_{\neg\delta}| + 38|B| + 30|\gamma||B| + 30|\delta||B| + 29|\gamma| + 29|\delta| + 81\\
\leq\ & C\cdot|B|(|\gamma|+1)^2 + C\cdot|B|(|\delta|+1)^2 + 38|B|(|\gamma|+|\delta|+2) + 29(|\gamma|+|\delta|+2)\\
=\ & C\cdot|B|(|\gamma|+1)^2 + C\cdot|B|(|\delta|+1)^2 + (38|B|+29)(|\gamma|+|\delta|+2)\\
\leq\ & C\cdot|B|(|\gamma|^2+|\gamma|+1+|\delta|^2+|\delta|+1+|\gamma|+|\delta|+2)\\
=\ & C\cdot|B|(|\gamma|^2+|\delta|^2+2|\gamma|+2|\delta|+4)\\
\leq\ & C\cdot|B|(|\gamma|+|\delta|+2)^2\\
=\ & C\cdot|B||\alpha|^2
\end{aligned}
$$

Therefore, there are TF PK proofs of $B,\alpha(1)\rightarrow\alpha(B)$ and $\neg B,\alpha(0)\rightarrow\alpha(B)$ of size $O(|B||\alpha|^2)$, where we choose C to be some reasonably high constant (say, 100). □

Lemma 28 *If $S = \Gamma\rightarrow\Delta$ is some valid sequent, and both Γ and Δ are variable-free (that is, Γ and Δ contain only $1, 0, \vee, \wedge,$ and $\neg$), then S has a PK proof of size $O(|S|^2)$.*

PROOF: From the basic semantics of a sequent, we see that if $\Gamma\rightarrow\Delta$ is a variable-free tautology, then either Γ is false, or Δ is a tautology, or both. This means that some formula in Γ is false, and/or some formula in Δ is true. Call this the *critical formula.*

We show how to construct a short proof of the critical formula (call it Ψ), and then we can use repeated weakenings to introduce the rest of Γ and Δ.

We proceed by induction on the number of logical connectives ($\wedge, \vee$, and $\neg$) in α. If α has no connectives, then it must be either a 0 on the left, or a 1 on the right. Then we proceed from the appropriate PK axiom:

$$0 \longrightarrow \qquad \text{or} \qquad \longrightarrow 1$$

and weaken to introduce the other formulas in Γ and Δ. This takes one step per formula in the sequent, and thus the proof is linearly bounded.

Assume that whenever the critical formula has k or fewer connectives, then, depending on whether α is true or false, an PK proof of either $\alpha \longrightarrow$ or $\longrightarrow \alpha$ exists, and that the proof has size $|\alpha|^2$.

Next, let α be a variable-free formula with exactly $k+1$ connectives. For clarity, suppose that α appears on the right (and hence is true). The other case is basically identical. Then we must give a short proof of $\longrightarrow \alpha$. There are three cases, depending on the outermost connective of α.

If $\longrightarrow \alpha$ is $\longrightarrow \Psi \vee \Phi$ then either Ψ or Φ is true. Assume that Ψ is true. Since $\longrightarrow \Psi$ has at most k connectives, we have a proof π_Ψ with size at most $|\Psi^2|$. Then:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\ \pi_\Psi \\ \longrightarrow \Psi\end{array}}{\longrightarrow \Psi, \Phi}}{\longrightarrow \Psi \vee \Phi}$$

Which has size at most:

$$\begin{aligned} & |\Psi|^2 + 2|\Psi| + 2|\Phi| + 1 \\ < \ & |\Psi|^2 + |\Phi|^2 + 2|\Phi||\Psi| + 2|\Psi| + 2|\Phi| + 1 \\ = \ & (|\Psi| + |\Phi| + 1)^2 \\ = \ & |\alpha|^2 \end{aligned}$$

If α is $\Psi \wedge \Phi$ then both Ψ and Φ must be true and have at most k connectives each. Then proofs π_Ψ and π_Φ exist of $\longrightarrow \Psi$ and $\longrightarrow \Phi$. These proofs have size at most $|\Psi^2|$ and $|\Phi^2|$. Then:

$$\dfrac{\begin{array}{c}\vdots\ \pi_\Psi \\ \longrightarrow \Psi\end{array} \qquad \begin{array}{c}\vdots\ \pi_\Phi \\ \longrightarrow \Phi\end{array}}{\longrightarrow \Psi \wedge \Phi}\ \wedge - \mathit{right}$$

which has size at most:

$$\begin{array}{rl} & |\Psi|^2 + |\Phi|^2 + |\Psi| + |\Phi| + 1 \\ < & (|\Psi| + |\Phi| +)^2 \\ = & |\alpha|^2 \end{array}$$

Lastly, if α is $\neg\Phi$ then Φ must be false. Since it has k connectives, there is a proof π_Φ of $\Phi \longrightarrow$ with size at most $|\Phi^2|$. Then:

$$\dfrac{\begin{array}{c}\vdots\ \pi_\Phi \\ \Phi \longrightarrow\end{array}}{\longrightarrow \neg\Phi}\ \neg - \mathit{right}$$

Which has size at most $|\Phi|^2 + |\Phi| + 1 < (|\Phi| + 1)^2 = |\alpha|^2$.

□

9.1 LA Axioms

Here we enumerate the axioms of **LA**. Recall that **LA** has three types, which we will represent by the following letters: indices $\{i, j, k, \ldots\}$, field elements $\{a, b, c, \ldots\}$, and matrices $\{A, B, C, \ldots\}$.

Firstly, the five axioms for equality:

E1 $\to x = x$

E2 $x = y \to y = x$

E3 $(x = y \wedge y = z) \to x = z$

E4 $x_1 = y_1, \ldots, x_n = y_n \to fx_1 \ldots x_n = fy_1 \ldots y_n$

E5 $i_1 = i_2, j_1 = j_2, i_1 \leq i_2 \to j_1 = j_2$

Then the 12 axioms for indices:

I6 $\to i + 1 \neq 0$

I7 $\to i * (j + 1) = (i * j) + i$

I8 $i + 1 = j + 1 \to i = j$

I9 $\to i \leq i + j$

I10 $\to i + 0 = i$

I11 $\to i \leq j, j \leq i$

I12 $\to i + (j + 1) = (i + j) + 1$

I13 $i \leq j, j \leq i \to i = j$

I14 $\to i * 0 = 0$

I15 $i \leq j, i + k = j \to j - i = k$

I15a $i \not\leq \to j - i = 0$

I16 $j \neq 0 \to \mathrm{rem}(i, j) < j$

I16a $j \neq 0 \to i = \mathrm{div}(i, j) + \mathrm{rem}(i, j)$

I17 $\alpha \to \mathrm{cond}(\alpha, i, j) = i$

I17a $\neg\alpha \to \mathrm{cond}(\alpha, i, j) = j$

The 10 axioms for field elements:

F18 $\rightarrow 0 \neq 1 \wedge a + 0 = a$

F19 $\rightarrow a + (-a) = 0$

F20 $\rightarrow 1 * a = a$

F21 $a \neq 0 \rightarrow a * (a^{-1}) = 1$

F22 $\rightarrow a + b = b + a$

F23 $\rightarrow a * b = b * a$

F24 $\rightarrow a + (b + c) = (a + b) + c$

F25 $\rightarrow a * (b * c) = (a * b) * c$

F26 $\rightarrow a * (b + c) = a * b + a * c$

F27 $\alpha \rightarrow \mathrm{cond}(\alpha, a, b) = a$

F27a $\neg\alpha \rightarrow \mathrm{cond}(\alpha, a, b) = b$

Finally, the 7 axioms for matrices:

M28 $(i = 0 \vee r(A) < i \vee j = 0 \vee c(A) < j) \rightarrow e(A, i, j) = 0$

M29 $\rightarrow r(\lambda ij \,\langle m, n, t\rangle) = m$

M29a $\rightarrow c(\lambda ij \,\langle m, n, t\rangle) = n$

M29b $1 \leq i, i \leq m, 1 \leq j, j \leq n \rightarrow e(\lambda ij \,\langle m, n, t\rangle, i, j) = t$

M30 $r(A) = 1, c(A) = 1 \rightarrow \Sigma(A) = e(A, 1, 1)$

M31 $r(A) = 1, 1 < c(A) \rightarrow \Sigma(A) = \Sigma(\lambda ij \,\langle 1, c(A) - 1, A)ij\rangle) + A_{1c(A)}$

M32 $c(A) = 1 \rightarrow \Sigma(A) = \Sigma(A^t)$

M33 $1 < r(A), 1 < c(A) \rightarrow \Sigma(A) = e(A, 1, 1) + \Sigma(R(A)) + \Sigma(S(A)) + \Sigma(M(A))$

M34 $r(A) = 0 \vee c(A) = 0 \rightarrow \Sigma(A) = 0$

Bibliography

[1] P. Beame and T. Pitassi, "Propositional Proof Complexity: Past, Present,and Future," *Theoretical Computer Science Entering the* 21^{st} *Century*, pp. 42–70, 2001.

[2] S. R. Buss, "Some remarks on the lengths of propositional proofs," *Archive for Mathematical Logic*, vol. 34, pp. 377–394, 1995.

[3] S. Cook and R. Reckhow, "On the Lengths of Proofs in the Propositional Calculus," in *The Sixth Annual ACM symposium on the Theory of Computing*, pp. 135–148, 1974.

[4] S. A. Cook, "CSC 438F/2404F Computability and Logic (course notes)." Offered at the University of Toronto, Department of Computer Science, 2003.

[5] M. Dowd, *Propositional Representation of Arithmetic Proofs.* PhD thesis, University of Toronto, 1979.

[6] S. B. (ed.), *Handbook of Proof Theory.* Elsevier, 1998.

[7] G. Hajós, "Über eine Konstruktion nicht n-färbbarer Graphen," *Wiss. Zeitschr. Martin Luther Univ. Halle-Wittenberg*, vol. 10, pp. 116–117, 1961.

[8] J. Krajíček, *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Cambridge University Press, 1995.

[9] T. Paterson and M. Soltys, "A propositional proof system with quantification over permutations of variables." Submitted for publication in the Journal of Discrete Applied Mathematics, 2005.

[10] T. Pitassi and A. Urquhart, "The complexity of the Hajós Calculus," *SIAM Journal on Discrete Mathematics*, vol. 8 (3), pp. 464–483, 1995.

[11] M. Sipser, *Introduction to the Theory of Computation, Second Edition.* Thomson Course Technology, 2006.

[12] M. Soltys, *The Complexity of Derivations of Matrix Identities.* PhD thesis, University of Toronto, 2001.

[13] M. Soltys, "LA, permutations, and the Hajós calculus," in *The 31st International Colloquium on Automata Languages and Programming*, pp. 1176–1187, 2004.

[14] M. Soltys and S. Cook, "The Proof Complexity of Linear Algebra," *Annals of Pure and Applied Logic*, vol. 130, pp. 277–323, 2004.