Power of Non-Uniformity in Proof Complexity

by

Steven Perron

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Your file Votre référence
ISBN: 978-0-494-58984-7
Our file Notre référence
ISBN: 978-0-494-58984-7

Abstract

Power of Non-Uniformity in Proof Complexity

Steven Perron

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2009

As the title indicates, this thesis is concerned with the strength of non-uniformity in proof complexity. The non-uniform part is there because we look at quantified propositional proof systems. With these proof systems we are interested in the minimum size of proofs that prove a family of tautologies. Like circuits, these proofs are not necessarily easy to construct. We measure the strength of a proof system by characterizing which families of tautologies have polynomial-size proofs in the proof system.

The proof systems we examine were first introduced by Krajícek and Pudlák [23], but have only received limited attention since then. These systems are called G_i and G_i^*. G_i is the propositional version of Gentzen's LK with cut formulas restricted to formulas with $i-1$ quantifier alternations, and G_i^* is the treelike version of G_i.

We look at the strength of these proof systems in three ways. The first is to compare them with bounded arithmetic. We find the weakest theory that can prove that a given proof system is sound. The second method is to compare proof systems with respect to computational complexity. We determine the complexity of finding a witness for the outermost existential quantifiers of a formula given a proof of that formula. The final method is to compare the proof systems to each other. This is done using the notion of p-simulations. An important tool in all of this work is an adaptation of the Herbrand theorem for the propositional setting. Most proof-theoretic proofs of the Herbrand theorem require cut-elimination which causes the size of the proof to increase exponentially.

We adapt the statement of the theorem and the proof to avoid this increase.

We finish by defining new proof systems that meet given requirements. The goal is to determine where proof systems get their strength. In all cases, it comes down to the eigenvariables. The strength of a proof system that allows quantifiers in the cut formulas is determined by the difficulty of witnessing the eigenvariables.

Acknowledgements

I would like to thank my parents for raising me to be the person I am today. I want to thank my supervisor Stephen Cook for guiding me through the research and preparation for this thesis. I also thank my supervisory committee for many useful suggestions at the checkpoints. I also want to thank Phuong Nguyen for taking time to discuss some of the ideas surrounding the topics of this thesis. I should not leave out the reviewers of the papers I wrote that became part of this thesis. As well, very little would have been done without the financial support of NSERC and the University of Toronto. Finally, I want to thank God for all He has done for me.

Contents

Part I

Basics

Chapter 1

Introduction

In this thesis, we investigate the strength of quantified propositional proof systems. The motivation for this investigation comes from its connection with computational complexity and bounded arithmetic. These systems are often viewed as the non-uniform versions of the fragments of bounded arithmetic.

Computational complexity is the area of computer science that is concerned with finding the minimum resources needed to compute a function (or to decide if a given input is in a set). The main concept is that of a complexity class. Problems are classified according to the resources needed to solve the problem. Some examples of these complexity classes are

$$AC^0 \subsetneq TC^0 \subseteq NC^1 \subseteq L \subseteq NL \subseteq P \subseteq PH. \tag{1.0.1}$$

AC^0 is the set of problems that can be solved by polynomial-size, constant-depth circuits with unbounded fan-in.[1] The class TC^0 is the same as AC^0 except threshold gates can now be used. NC^1 is the set of problems that can be solved with polynomial-size, logarithmic-depth circuits with fan-in 2. The class L is the set of problems that

[1]In this thesis, we are interested in the uniform version. So, unless stated otherwise, circuits class are *DLogtime*-uniform.

can be solved on a deterministic Turing machine using logarithmic space. NL is the same except the Turing machine is now non-deterministic. P is the set of problems that can be solved in polynomial-time on a Turing machine. Finally, PH refers to the polynomial-time hierarchy.

There are many problems that we know are in one class, and we believe they are not in a smaller class. However, it is difficult to prove. In fact, except for the first inclusion, none of the inclusions in (1.0.1) are known to be proper. We are unable to separate TC^0 from PH despite all of the research over the past 30 years.

With this in mind, it might be useful to find other ways of looking at these problems. One such alternative is bounded arithmetic. The area is called bounded arithmetic because we are considering logical theories of arithmetic where induction is limited to formulas where every quantifier is bound by some term. For example, existential quantifiers are of the form $\exists x < t$, which means there exists an x less than t. Bounded arithmetic was first introduced by Parikh in [32]. In that paper, he introduced the theory that is now commonly known as $I\Delta_0$. This theory was studied from a logical perspective, but also in connection with computational complexity. (We will not mention exactly what the connection is until we get to the theories that we use throughout the thesis.) A few years later, Cook introduced the theory PV as a way of capturing polynomial-time reasoning [7]. Later, Buss introduced his theory S_2 with its fragments [4]. S_2 is a theory that is typically associated with PH. Then by restricting the induction axiom in S_2, we get theories that we can associate with the levels of the PH. Buss's theories have become the standard theories for bounded arithmetic, but unfortunately the language of S_2 does not lend itself to defining nice theories for complexity classes smaller than P. So when Cook and Nguyen decided to define theories for the smaller classes, they used the two-sorted language of Zambella [41].

This language has two sorts: numbers and binary strings. It includes basic arithmetic for the numbers and a length function for the strings. As well, we are able to check if the ith bit of a string is 0 or 1. The Σ_0^B formulas are the formulas with any number of bounded number quantifiers, but no string quantifiers. We can define a number of theories that correspond to the complexity classes above

$$V^0 \subsetneq VTC^0 \subseteq VNC^1 \subseteq VL \subseteq VNL \subseteq TV^0 \subseteq V^\infty. \tag{1.0.2}$$

The theory V^0 is the base theory. It includes axioms that define addition, multiplication, and the other functions in the language [8]. On top of that, it includes comprehension for Σ_0^B formulas. The other theories below TV^0 are defined by taking V^0 and adding an axiom that says a function that is complete for the corresponding complexity class is total [11]. TV^0 is V^0 plus an appropriate induction scheme. V^∞ is the same as V^0 except it has comprehension for all bounded formulas. These theories are defined more formally in Chapter 2.

Each of these theories is associated with their corresponding complexity class by a witnessing theorem. For example, if TV^0 proves $\exists X\phi(X, Y)$, where ϕ is Σ_0^B or Σ_1^B, then there is a polynomial-time computable function $F(Y)$ such that $\phi(F(Y), Y)$ is true [8]. In the other direction, TV^0 can prove that every function in P is total [8]. This gives us the connection between the theory and the complexity class.

Using the witnessing theorems, we can connect the provability of certain theorems with complexity classes. One example is Fermat's Little Theorem: If N (encoded as a string) is a prime, then $A^{N-1} \equiv 1(modN)$ for $1 \leq A < N$ (as a string again). If this can be proved in TV^0, then this will lead to a randomized polynomial-time algorithm that factors integers [11]. This is interesting because it is generally believed that no such algorithm exists, and, in fact, the security of the RSA encryption algorithm depends on it. So, this example says that Fermat's Little Theorem is provable with polynomial-time concepts only if RSA is insecure.

Another interesting example is the min-cut/max-flow theorem. Think of a graph that represents a network of computers. The maximum transfer speed from one computer to another is determined by the capacities of the cables connecting these computers. So given a network, we might want to determine this value. An obvious upper bound is a cut. If we can find a set of cables that, if cut, separate the two computers in question, then any transfer of information must pass through these cables. So we cannot have more than these cables allow. The min-cut/max-flow theorem says that, in fact, the maximum flow is equal to the minimum cut. It is possible to prove this theorem in VNL; however, if this theorem is provable in VL, then $L = NL$. (See Appendix A for more information on this result.) The central idea is that we can define a decision based on the statement of the theorem, and that problem is complete for NL. The corresponding decision problem is, given a directed graph with integer capacities and a valid flow, is the flow a maximum flow?

These are just two examples that show that proof complexity is closely connected to computational complexity. Another part of proof complexity is to look at propositional proof complexity. In this setting, we are no longer interested in finding the axioms needed to prove a formula. In fact, from the completeness theorem, we know that every tautology is provable using a very simple set of axiom schemes. Instead we are interested in the minimum size of a proof of a tautology. Proof systems are defined by giving the axioms and rules of inferences. Then we examine the size of proofs in a given proof system. The two most common proof systems are resolution and Frege proof systems. There have been a number of non-trivial lower bounds for resolution, but no super-polynomial lower bounds are known for Frege.

This line of research is motivated by a result of Cook and Reckhow: there exists a polynomially-bounded proof system if and only if $NP = coNP$ [13]. This result led to the search for lower-bounds in different proof systems. As in computational complexity, there has been limited success.

As research continued, a connection between bounded arithmetic and propositional proof complexity was found. The first example was in [7], where Cook showed that, if PV proves a formula ϕ, then that formula can be translated into a polynomial-size family of propositional formulas that have polynomial-size extended-Frege proofs. A similar connection exists between $I\Delta_0(R)$ and bounded-depth Frege [33]. For the two-sorted theories, there are similar results. For example, if VNC^1 proves a Σ_0^B formula ϕ, then there are polynomial-size Frege proofs of the translation of ϕ. This connection is not just one sided. For the other direction, the theory can prove that its corresponding proof system is sound. So, if you have uniform[2] proofs of the translation of a Σ_0^B formula ϕ in a proof system, then the corresponding theory proves ϕ. This has led to the view that the proof system is the non-uniform version of the theory.

For the two-sorted theories, this connection only exists for the Σ_0^B formulas, but it is fair to wonder about formulas with bounded string quantifiers. To take care of this, we move into quantified propositional proof systems. For now, we can think of a Σ_i^B formula as a bounded formula with $i-1$ alternations in the string quantifiers. A more formal definition is given in the next chapter.

1.1 Outline of results in Part II

In [23], Krajícek and Pudlák introduced the quantified propositional proof system G and its fragments. These fragments have close connections with bounded arithmetic and computational complexity. In particular, the collapse of PH, V^∞, and the fragments of G are all related [23, 24, 22, 28]. Even with these close connections to important open problems in logic and computer science, little work has been done investigating the fragments of G. The goal of this thesis is to take a closer look at these proof systems.

The proof system G is the sequent calculus for quantified propositional formulas. A Σ_i^q formula is a quantified propositional formula with at most $i-1$ quantifier alternations

[2]The uniformity that we use here is very strong. The proofs have to be more than constructable, but the theory must be able to prove that the construction is correct.

starting with $\exists$ on the outside. Following Morioka, the proof system G_i is defined by restricting G to proofs where all cut formulas are Σ_i^q and G_i^* is the treelike version of G_i [10, 29]. Note that originally G_i and G_i^* were defined by restricting all formulas, not just cut formulas, to Σ_i^q formulas [23, 22].

For those unfamiliar with these proof systems, you can think of G_i as a proof system where any lemma you use to prove a theorem is in the ith level of PH. The proof system G_i^* is the same except each lemma can be used only once.

When G_i^* was first considered in [26], it was used to axiomatize V^i. For this reason G_i^* has informally been described as the non-uniform version of V^i. (V^i is the restriction of V^∞ to comprehension on Σ_i^B formulas). This axiomatization comes from the close connection between the Σ_i^B theorems of V^i and G_i^* proofs of Σ_i^q formulas [22]. Every bounded theorem of V^i can be translated into a family of quantified propositional formulas that have polynomial-size G_i^* proofs. As well, V^i can prove that G_i^* is sound when proving Σ_i^q formulas. So, if V^1 can prove that there exists G_i^* proofs of the translation of a Σ_i^B formula ϕ, then V^i proves ϕ.

The same can be said for G_i and TV^i. When G_i was first introduced in [23], it was viewed as TV^i from a different perspective. The idea was to cast questions about TV^i in a more combinatorial setting.

In Part II, we intend to examine the strength of these proof systems. Since these proof systems are complete, we do not measure the strength of a proof system by what it can prove as in bounded arithmetic. Instead, we measure the strength of a proof system by determining which families of tautologies have polynomial-size proofs in that proof system. This idea leads to three ways of exploring these proof systems.

The first way to measure the strength of a proof system is to look at its witnessing problems. The problem is, given a proof of a Σ_i^q formula and values for the free variables, find values for the outermost existential quantifiers that satisfy the formula. We are interested in finding the complexity of this function for different proof systems and

Σ_4^q	$FP^{\Sigma_3^p}[wit, O(\log n)]$ †	$FP^{\Sigma_3^p}[wit, O(\log n)]$ †	$FP^{\Sigma_3^p}[wit, O(\log n)]$ †	$FP^{\Sigma_3^p}[wit, O(\log n)]$ †
Σ_3^q	$FP^{\Sigma_2^p}[wit, O(\log n)]$ †	$FP^{\Sigma_2^p}[wit, O(\log n)]$ †	$FP^{\Sigma_2^p}[wit, O(\log n)]$	$FP^{\Sigma_2^p}$ †
Σ_2^q	$FP^{NP}[wit, O(\log n)]$	FP^{NP} †	FP^{NP}	PLS^{NP}
Σ_1^q	FP	PLS	PLS †	$CPLS$ †
	G_1^*	G_1	G_2^*	G_2
Σ_4^q	$FP^{\Sigma_3^p}[wit, O(\log n)]$	$FP^{\Sigma_3^p}$ †	$FP^{\Sigma_3^P}$	$PLS^{\Sigma_3^p}$
Σ_3^q	$FP^{\Sigma_2^P}$	$PLS^{\Sigma_2^p}$	$PLS^{\Sigma_2^p}$ †	$CPLS^{\Sigma_2^p}$ †
Σ_2^q	PLS^{NP} †	$CPLS^{NP}$ †	$CPLS$ †	??
Σ_1^q	$CPLS$ †	??	??	??
	G_3^*	G_3	G_4^*	G_4

Table 1.2: Complexity of the Witnessing Problem for G_i and G_i^*. Observe the patterns that exists along the diagonals. The results marked with † indicate those that improve on previous results.

different values of i. A lot of work has been done on this problem in [29]. Some of our results are summarized in Table 1.2. As the table indicates, we improved on the results in [29] by proving that the Σ_j^q witnessing problem for G_i^* is complete for $FP^{\Sigma_{j-1}^p}[wit, \log]$ when $j \geq i+2$. Previously, it was known to be in the class, but not known to be hard for the class. An important observation is that, if $i > j > k$, then we cannot see a difference between G_j^* and G_k^* for Σ_i^q formulas. That is, we are unable to differentiate between G_j^* and G_k^* using the witnessing theorem. The other interesting observation is that the complexity of witnessing a G_i^* proof of a complex formula is harder than it is for V^i. In the uniform setting, only a constant number of calls the oracle is needed while, in the non-uniform setting, a logarithmic number is needed.

In order to try to differentiate between the proof systems at this level, we also look at these problems in an alternative model of computation: interactive computations [20]. This was a model that was first introduced as a way of looking at witnessing in bounded arithmetic [24]. In this model, there are two people, the student and the teacher. The job of the student is to find the witness and he can get help from the teacher. As the capabilities of the student and teacher change, we are able to capture witnessing for the different proof systems. In this model, we find that the difference between G_j^* and G_k^* is the capability of the student.

We also take a look at the witnessing problems for simple formulas. That is, we want to witness G_i^* or G_i proofs of Σ_j^q formulas when $j < i$. Unfortunately, we are not able to completely determine the complexity of these problems, but we make some progress. First, we prove that G_i is p-equivalent to G_{i+1}^* for Σ_{i+1}^q formulas. As well, we observe that the witnessing problem for the proof systems is related to witnessing in the corresponding theory of bounded arithmetic. We are able to use the results in [25] to solve the Σ_{i-1}^q witnessing problem for G_i and G_{i+1}^*. The idea is to extend the class polynomial local search (PLS). In PLS, we are essentially looking for a sink in a very large directed acyclic graph. Colour PLS (CPLS) is the same except we are searching for a sink that meets some special property.

As well, we use this connection to give a new proof of the results of Krajícek, Skelley, and Thapen. The intent is that this new proof may be able to generalize to the rest of the witnessing problems.

The second way of examining the strength of these proof systems is to compare them to each other. This is done using the standard idea of a p-simulation. These results are summarized in Figure 1.1, and should be contrasted with the corresponding results concerning the theories (See Figure 1.2). The first result in this direction is to compare extended-Frege and G_1^*. In [22], it was shown that extended-Frege is p-equivalent to G_1^* with respect to quantifier-free formulas. This means that, when proving quantifier-free formulas, G_1^* only needs to cut quantifier-free formulas and extension formulas. This raises the question of whether or not this holds when G_1^* is used to prove more complicated formulas. We define a quantified version of extended-Frege called GPV^*, and prove that GPV^* and G_1^* are p-equivalent with respect to all prenex formulas. This result is surprising because the class of formulas that GPV^* can cut is much less expressive than the class of formulas that G_1^* can cut. As well, this result does not fit with the view that GPV^* corresponds to TV^0 and G_1^* with V^1. This is because VPV is a strict sub-theory

$$\begin{tikzcd}
 & G_1 & & G_2 & & G_3 & & G_4 & \\
 & \Sigma^q_2 & & \Sigma^q_3 & & \Sigma^q_4 & & & \\
G^*_1 & & G^*_2 & & G^*_3 & & G^*_4 & & \cdots \\
GPV^* & & GPV^*_2 & & GPV^*_3 & & GPV^*_4 & &
\end{tikzcd}$$

Figure 1.1: Summary of Simulations: An arrow from proof system P_1 to proof system P_2 indicates that P_2 p-simulates P_1. A label on the arrow means the simulation holds with respect to that class of formulas. Dashed arrows indicate the new results given in this thesis.

$$\begin{tikzcd}
 & TV^1 & & TV^2 & & TV^3 & & TV^4 & \\
 & \Sigma^B_2 & & \Sigma^B_3 & & \Sigma^B_4 & & & \cdots \\
V^1 & & V^2 & & V^3 & & V^4 & &
\end{tikzcd}$$

Figure 1.2: Summary of Conservation Results: An arrow from theory T_1 to theory T_2 indicates that T_2 is conservative over T_1. A label on the arrow means holds for that class of formulas only.

of V^1 assuming PH does not collapse [24]. We can generalize the definition of GPV^* to a proof system GPV^*_i by allowing it to cut formulas of the form $\exists x[x \leftrightarrow A]$, where A is a Σ^q_{i-1} formula. Then we can show that GPV^*_i is p-equivalent to G^*_i for prenex formulas. The interesting observation is that there is a simulation that is possible in the non-uniform setting that is not possible in the uniform setting.

We also take a look at G_i and G^*_{i+1}. If we used the connections with bounded arithmetic as a guide, we would expect G^*_{i+1} to be a strictly stronger proof system than G_i. This is because G^*_{i+1} is associated with V^{i+1} and G_i with TV^i. Again, TV^i is a strict sub-theory of V^{i+1} assuming PH does not collapse [24]. If we were to make this assumption, we would be wrong. Nguyen showed that G^*_{i+1} is not stronger than G_i [30]. This was done by showing that, under an appropriate complexity assumption, G^*_{i+1} does not simulate G_i or even cut-free G for Σ^q_{i+2} formulas. As well, we prove that G_i is stronger than G^*_{i+1}, which is surprising. This is done by showing that G_i p-simulates G^*_{i+1} for all quantified propositional formulas, not just Σ^q_i formulas as in [22].

The third way of examining G^*_i is to find the weakest fragment of V^∞ that can prove that G^*_i is sound. So, we are looking for a theory that proves that, if there is a G^*_i proof of a formula, then that formula is valid. Informally, this gives an upper bound on the reasoning power of G^*_i. The idea is that if a theory T proves that a proof system P is sound and T proves that there are P proofs of the propositional translation of a formula ϕ, then T proves ϕ. This is the important idea that was used when G_i was first introduced [23]. Then these results lead to new axiomatizations of the theories of bounded arithmetic. Informally, this can be viewed as applying uniformity to the proof systems.

In [22], it was shown that V^1 proves that G^*_1 is sound with respect to Σ^q_1 formulas. However, assuming PH does not collapse, V^1 does not prove that G^*_1 is sound with respect to Σ^q_3 formulas [29]. This result seems to indicate that, as the quantifier complexity of the formulas we are proving grows, the reasoning power of G^*_1 grows beyond any finite level of V^∞. In fact, the same proof also shows that, assuming PH hierarchy does not collapse, TV^i does not prove that G^*_1 is sound with respect to Σ^q_{i+2} formulas; however, we show that V^{i+1} proves that G^*_{i+1} is sound with respect to Σ^q_{i+2} formulas. Informally this means that the reasoning power of G^*_1 relative to Σ^q_{i+2} formulas is not stronger than the reasoning power of V^{i+1}. These results are summarized in Table 1.3.

We then use these results to see what happens if we apply some type of uniformity to these proofs systems. We do this by looking at a theory that is axiomatized by V^1 plus statements stating that a proof systems is sound. In [23], Krajícek and Pudlák were able to prove that V^∞ can be axiomatized by V^1 plus axioms stating G_i is sound relative to

Σ_6^q	V^5	V^5	V^5	V^5	V^5	TV^5
Σ_5^q	V^4	V^4	V^4	V^4	TV^4	TV^5
Σ_4^q	V^3	V^3	V^3	TV^3	TV^4	TV^5
Σ_3^q	V^2	V^2	TV^2	TV^3	TV^4	TV^5
Σ_2^q	V^1	TV^1	TV^2	TV^3	TV^4	TV^5
Σ_1^q	TV^0	TV^1	TV^2	TV^3	TV^4	TV^5
	G_1^*	G_2^*	G_3^*	G_4^*	G_5^*	G_6^*

Table 1.3: Summary Of Reflection Principles: The entries indicate that weakest theory that is known to prove the proof system is sound with respect to the class of formulas.

Σ_i^q formulas, for all $i \geq 1$. We show that the same is true when G_i is replaced by G_1^*. In fact, we can replace G_i by the cut-free version of G^*. This is a surprising result for two reasons. The first reason is that the cut-free version of G^* is a strictly weaker proof system that G_1^*; however, when we apply uniformity to these proof systems we get the same theory. The second reason this is surprising is that it captures all of V^∞. Since we associated G_1^* with V^1, we expected to get a theory that was closely related to V^1.

At the same time, we are able to give a new axiomatization of TV^i and V^i. We can axiomatize TV^i by V^1 plus an axiom stating G_{i+1}^* (or G_i) is sound with respect to Σ_{i+1}^q formulas, and V^i as V^1 plus an axiom stating G_i^* is sound with respect to Σ_{i+1}^q formulas

From these axiomatizations, we get a corollary that if V^∞ collapses to V^i, then, for $j > i$, G_i^* p-simulates G_j^* for Σ_{i+1}^q formulas (see [23]). Note that this result holds for Σ_{i+1}^q formulas only, but it is fair to ask if this result can be improved. We partly answer this question. We show that if $V^i = V^\infty$, then G_{i+3}^* p-simulates G_j^* $(j > i+3)$ for all quantified propositional formulas. We take a stronger proof system, but the simulation holds for every formula not just Σ_{i+1}^q formulas. The idea is that, if $V^i = V^\infty$, then V^i proves that $\Sigma_{i+3}^p = \Pi_{i+3}^p = PH$ [6]. So, every Σ_j^q formula can be turned into an equivalent Σ_{i+3}^q formula.

The main tool used is the proofs of some of these theorems is a witnessing theorem in the style of the KPT witnessing theorem [24]. The original KPT witnessing theorem describes how hard it is to witness Σ_{i+3}^B theorems of TV^i, for $i \geq 0$. This theorem has been used to prove that the collapse of the V^∞ hierarchy implies the collapse of PH [24], and to show that certain weak theories do not prove the Σ_1^B replacement scheme, relative to some complexity assumptions [12]. In this thesis, we adapt the statement of the KPT witnessing theorem to G_i^*, and then prove it. The main difficulty is that proofs of the KPT witnessing theorem rely on the cut-elimination theorem, which unfortunately causes the size of the proof to increase exponentially. We must avoid this increase, so we have to find a way to work around cut formulas.

1.2 Outline of the results in Part III

In the third part, we are interested in finding ways of defining new proof systems that meet certain desired properties. The goal is to better understand the source of the strength of the proof systems.

For the most part, proof systems have been defined by changing the set of formulas that can be cut. The intuition was that, if we change the cut formulas, we change the strength of the proof system. However, recall that GPV^* (quantified extended-Frege) and G_1^* are p-equivalent. It does not matter if the proof system can cut every Σ_1^q formula, or just the simple formulas of GPV^*. This raises the question of where the strength of these proof systems comes from. If it is not just the cut formulas, what is it? In [34], it was observed that the source is the eigenvariables. In the quantifier rules $\exists$-left and $\forall$-right (given in next chapter), there is a free-variable in the upper sequent that gets quantified. This is an eigenvariable. In [34], a proof system for logarithmic-space reasoning is defined by restricting the use of eigenvariable. In particular, the free variables in non-Σ_0^q formulas cannot be eigenvariables. This proof system was called GL^*. We explore this proof system a little further. In that work, it was shown that Σ_1^B theorems of VL can be translated into a family of tautologies that have polynomial-size GL^* proofs. However, it was not shown that VL can prove that GL^* is sound. This was later shown in [35], and we give this result here too. The main idea is to formalize an algorithm that finds a satisfying assignment to a given $CNF(2)$ formula if one exists. A $CNF(2)$ formula is a CNF formula where no variable appears more than twice in the formula.

Then to prove the reflection principle, we define a logarithmic-space function that witnesses every eigenvariable in the proof. This is not possible in GPV^* unless $L = P$, for the eigenvariables can be used to simulate circuits.

To see if this observation holds more generally, we define proof system GNL^* in a similar fashion to GL^*. We start with a set of formulas that can be witnessed in NL. These are the $\Sigma Krom$ formulas, and they are based on the descriptive complexity characterization of NL given in [15]. Then GNL^* is obtained from G_1^* by restricting cuts to $\Sigma Krom$ formulas where no free variable is an eigenvariable.

As with other proof systems, we then prove a connection between GNL^* and VNL by proving the reflection principles for GNL^* and translating theorem of VNL. Proving the refection principle is essentially the same as proving the reflection principle for GL^*. The only difference is the function that witnesses the eigenvariables is now a non-deterministic logarithmic-space function.

The translation theorem involves the main ideas of the GL^* case, but the style of the proof is changed. In [34, 35], the translation theorem for GL^* is proved in a proof theoretic way. We start with a proof and change it to get what we want. For the GNL^*, we use a model theoretic construction. We start with VNL and restrict the use of the axioms. Using model theoretic arguments, we prove that we can do this restriction. This corresponds to the normal form that we want.

Even if the style of the proof changes, the main difficulty remains the same: the need to formalize a proof that the NL functions are closed under composition. This is only true because NL is closed under complement. This means we must formalize a proof of the Immerman-Szelepcsenyi Theorem. This was done in [9, 18], but the proof was complicated. We redid the proof in a different way, so that we can say that it is provable in the restricted theory mentioned above.

The proof systems GL^* and GNL^* show how we can define a proof system that change the strength of the Σ_1^q proofs; however, it is still unclear how this affects the more complicated formulas. We pursue this line of reasoning by trying to define a proof system that is truly a non-uniform version of TV^i.

Earlier we mentioned that bounded-depth Frege is the non-uniform version of $I\Delta_0(R)$. We want to find a proof system that has the same connection with TV^i. The proof system G_{i+1}^* meets some of the qualifications. In particular, every theorem of TV^i can be translated into a family of polynomial-size G_{i+1}^* proofs. On the other hand, TV^i proves the Σ_{i+1}^q reflection principle, but, assuming PH does not collapse, it does not prove the Σ_{i+2}^q reflection principle. This gives an indication that we want to restrict G_{i+1}^*.

To understand how to restrict it, we look at the witnessing theorem. Comparing the witnessing theorems for TV^i and for G_{i+1}^*, we notice that for TV^i the "student-teacher" game has a constant number of rounds before the student is guaranteed to win; however, for G_{i+1}^* there is potentially a polynomial number of rounds. This gives an indication that, if we can somehow restrict G_{i+1}^* to reduce the number of rounds to a constant, this may give us a proof system that truly corresponds to TV^i.

In fact, this is what we do, except, for technical reasons, we use GPV_{i+1}^*. First we observe that the number of rounds corresponds to certain special uses of $\forall$-right and $\exists$-left–the rules with eigenvariables. Once this is done, we have a family of proof systems that correspond to TV^i in the same way that bounded-depth Frege corresponds to $I\Delta_0(R)$. Namely, for any bounded formula ϕ, if TV^i proves there exists $GPV_{i+1}^*(c)$ proofs of the translation of ϕ, then TV^i proves ϕ. As well, the translation theorem still holds: every theorem of TV^i can be translated into a family of tautologies that have polynomial-size $GPV_{i+1}^*(c)$ proofs.

Chapter 2

Basic Definitions and Notations

2.1 Two-Sorted Computational Complexity

We use two-sorted computational complexity. The two sorts are numbers and binary strings (aka finite sets). The numbers are intended to range over the natural numbers and will be denoted by lower-case letters. For example, i, j, x, y, and z will often be used for number variables; r, s, and t will be used for number terms; and f, g and h will be used for functions that return numbers. The strings are intended to be finite strings over $\{0,1\}$ with leading 0's removed. Since the strings are finite, they can be thought of as sets where the ith bit is 1 if i is in the set. The strings will be denoted by uppercase letters. The letters X,Y, and Z will often be used for string variables.

We assume that people are familiar with the basic complexity classes L, NL, P and NP. We will only describe the definitions that are not be so conventional. For those unfamiliar with these complexity classes, you can get all of the necessary background for these classes in any introductory text on the subject ([38] is one example).

The first unconventional feature is the format of the input. The inputs are a series of numbers represented in unary and a series of strings over $\{0,1\}$ with leading zeros removed. The idea being that the strings are the objects of interest and the numbers are just auxiliary. For example, in the graph connectivity problem, the graph is encoded as a string, but the nodes s and t are given as numbers. The idea is that the size of the graph is what really determines the size of the instance.

We are also interested in the polynomial-time hierarchy (PH). The ith level of the polynomial-time hierarchy (denoted Σ_i^p) is defined as $NP^{\Sigma_{i-1}^p}$ for $i \geq 1$ and $\Sigma_0^p = P$. Then

$$PH = \bigcup_{i=0}^{\infty} \Sigma_i^p.$$

We say that PH collapses if there is an i such that $PH = \Sigma_i^p$. This is fairly standard, but is not always covered in undergraduate level courses.

For functions, we say a number function $f(\vec{x}, \vec{X})$ is in FC, where C is a complexity class, if there is a polynomial p such that $f(\vec{x}, \vec{X}) < p(\vec{x}, |\vec{X}|)$, and the relation $f(\vec{x}, \vec{X}) = y$ is in C. A string function $F(\vec{x}, \vec{X})$ is in FC if the size of $F(\vec{x}, \vec{X})$ is bounded by a polynomial and if the relation

$$R(i, \vec{x}, \vec{X}) \equiv \text{ the } i\text{th bit of } F(\vec{x}, \vec{X}) \text{ is } 1$$

is in C.

The function classes can be defined by talking about output tapes, but this is equivalent. As well, for non-deterministic classes like NL, defining functions using an output tape is not as elegant.

As you may have already noticed from Table 1.2, we also consider complexity classes that have non-standard use of oracles. These should be mentioned here, but we will not go into formal definitions. If we ask an oracle if something exists, then the standard oracle response is a yes or no answer. However, it is also possible to consider a model where the oracle answers with a witness. For example, if an oracle is asked if a formula is satisfiable, it would answer with a satisfying assignment if one exists. As well, we may consider the possibility of restricting the use of the oracle. This leads to the following complexity classes.

Definition 2.1.1. Let $f : \mathbb{N} \to \mathbb{N}$ be a function and let n refer to the size of the input. The complexity class $FP^A[wit, f]$ is the set of functions that can be computed in polynomial-time with at most $f(n)$ calls to an oracle A that returns a witness.

We are particularly interested in the cases when $f(n)$ is sub linear. For example, we look at $f(n) \in O(1)$ and $f(n) \in O(\log n)$. These classes were originally defined for their association with bounded arithmetic (see [21, 36] for two examples.)

As you may have noticed in the introduction, these are many results that hold under the assumption that PH does not collapse. This assumption is relevant via the following theorem:

Theorem 2.1.2 (Theorem 8.7 [29])**.** *If* $FP^{\Sigma_i^p}[wit, O(1)] = FP^{\Sigma_i^p}[wit, O(\log n)]$, *then* $PH = \Sigma_{i+1}^p$.

2.2 Bounded Arithmetic

We will use two-sorted theories of bounded arithmetic. We follow the presentation in [8, 11]. As with two-sorted computational complexity, the two sorts are numbers and binary strings (aka finite sets).

The base language is

$$\mathcal{L}_A^2 = \{0, 1, +, \times, <, =, =_2, \in, ||\}.$$

The constants 0 and 1 are number constants. The functions + and × take two numbers as input and return a number–the intended meanings are the obvious ones. The language also includes two binary predicates that take two numbers: $<$ and $=$. The predicate $=_2$ is meant to be equality between strings, instead of numbers. In practice, the 2 will not be written because which equality is meant is obvious from the context. The membership predicate $\in$ takes a number i and a string X. It is meant to be true if the ith bit of X is 1 (or i is in the set X). This will also be written as $X(i)$. The final function $|X|$ takes a string as input and returns a number. It is intended to be the number of bits needed to write X when leading zeros are removed (or the least upper bound of the set X). The set of axioms 2BASIC is the set of defining axioms for $\mathcal{L}_A^2$.

We use $\exists X < b\ \phi$ as shorthand for $\exists X[(|X| < b) \ \wedge \ \phi]$. The shorthand $\forall X < b\ \phi$ means $\forall X[(|X| < b) \supset \phi]$. The set $\Sigma_0^B = \Pi_0^B$ is the set of formulas whose only quantifiers are bounded number quantifiers. For $i > 0$, the set Σ_i^B is the set of formulas of the form $\exists \vec{X} < \vec{t}\phi$ where ϕ is a Π_{i-1}^B formula. For $i > 0$, the set Π_i^B is the set of formulas of the form $\forall \vec{X} < \vec{t}\phi$ where ϕ is a Σ_{i-1}^B formula.

Now we can define the two main axiom schemes:

$$\Sigma_i^B\text{-comp: } \exists X \leq b \forall i < b[X(i) \leftrightarrow \phi(i)],$$

$$\Sigma_i^B\text{-string-ind: } [\phi(\emptyset) \wedge \forall X[\phi(X) \supset \phi(S(X))]] \supset \phi(Y)$$

where $\phi(i)$ is a Σ_i^B formula, and, for Σ_i^B-COMP, ϕ does not contain X, but may contain other free variables. The constant $\emptyset$ is the empty string, and the function $S(X)$ interprets X as a binary number and adds 1 to it. Note that we still view Σ_i^B-string-ind as a formula over $\mathcal{L}_A^2$. We simply replace the instances of $\emptyset$ and $S(X)$ by their Σ_0^B bit-definition.

We can now define two hierarchies of theories.

Definition 2.2.1. The theory V^i is axiomatized by the 2BASIC axioms plus Σ_i^B-comp. The theory TV^i is axiomatized by the 2BASIC axioms, Σ_0^B-comp, and Σ_i^B-string-ind.

For $i > 0$, V^i corresponds to S_2^i, and TV^i corresponds to T_2^i in that they are RSUV-isomorphic [11].

Another theory we often use is VPV, a universal theory with a function symbol for every polynomial-time function. The function symbols have the following defining axioms based on Cobham's Theorem:

Definition 2.2.2 (PV function symbols)**.** The language $\mathcal{L}_{FP}$ is the smallest set satisfying the following:

1. $\mathcal{L}_{FP}$ includes $\mathcal{L}_A^2 \cup \{pd, CHOP\}$.
2. For each open formula $\phi(z, \vec{x}, \vec{X})$ over $\mathcal{L}_{FP}$ and term $t = t(\vec{x}, \vec{X})$ over $\mathcal{L}_A^2$, there is a string function $F_{\phi,t}$ and a number function $f_{\phi,t}$ in $\mathcal{L}_{FP}$.
3. For each triple G, H, t, where $G(\vec{x}, \vec{X})$ and $H(y, \vec{x}, \vec{X}, Z)$ are functions in $\mathcal{L}_{FP}$ and $t(y, \vec{x}, \vec{X})$ is an $\mathcal{L}_A^2$ term, there is a function $F_{G,H,t}$ in $\mathcal{L}_{FP}$.

The 2BASIC axioms define the function symbols in $\mathcal{L}_A^2$. The rest of the function symbols have the following defining axioms:

- $pd(0) = 0$, $x \neq 0 \supset pd(x) + 1 = x$
- $CHOP(X, y)(i) \leftrightarrow i < y \wedge X(i)$
- $F_{\phi,t}(\vec{x}, \vec{X})(i) \leftrightarrow i < t(\vec{x}, \vec{X}) \wedge \phi(i, \vec{x}, \vec{X})$
- $i < f_{\phi,t}(\vec{x}, \vec{X}) \supset \neg\phi(i, \vec{x}, \vec{X})$
- $f_{\phi,t}(\vec{x}, \vec{X}) < t(\vec{x}, \vec{X}) \supset \phi(f_{\phi,t}(\vec{x}, \vec{X}), \vec{x}, \vec{X})$
- $F_{G,H,t}(0, \vec{x}, \vec{X}) = G(0, \vec{x}, \vec{X})$
- $F_{G,H,t}(y + 1, \vec{x}, \vec{X}) = CHOP(H(y, \vec{x}, \vec{X}, F_{G,H,t}(y, \vec{x}, \vec{X})), t(y, \vec{x}, \vec{X}))$

The theory VPV is axiomatized by quantifier-free equivalents of the 2BASIC axioms, induction on all open $\Sigma_0^B(\mathcal{L}_{FP})$ formulas, and the defining axioms for all of the $\mathcal{L}_{FP}$ function symbols. See [8, 11] for more information on VPV.

Another scheme of formulas we use is the Σ_i^B-MAX scheme:

$$[\exists x < b\ \phi(x)] \supset \exists x < b[\phi(x) \wedge \forall y < b(x < y \supset \neg\phi(y))]$$

where ϕ is Σ_i^B. This scheme essentially says that, if there exists a value for x less than b that satisfies $\phi(x)$, then there exists a maximum x less than b that satisfies $\phi(x)$. It can be shown that Σ_i^B-MAX is provable in V^i ([11], Corollary 5.8).

From time to time, we will use function symbols that are not in $\mathcal{L}_A^2$. The first is $X(i,j) \equiv X(\langle i,j\rangle)$, where $\langle i,j\rangle = (i+j)(i+j+1)+2j$ is the pairing function. It can be thought of as a two dimensional array of bits. The second is the row function. The notation we use is $X^{[i]}$. This functions returns the ith row of the two dimensional array X. In the same way, we can also describe three dimensional arrays. We also want to pair strings. So if $X = \langle Y_1, Y_2\rangle$, then $X^{[0]} = Y_1$ and $X^{[1]} = Y_2$. Note that, if we add these functions with their Σ_0^B defining axioms to the theory V^i, we get a conservative extension. They can also be used in the induction axioms [8]. This means that, if there is a V^i proof of a formula that uses these functions, there is a V^i proof of the same formula that does not use these functions.

These theories are interesting because of their connection with computational complexity. This is done by looking at which functions are definable in the theory.

Definition 2.2.3. A string function $F(X)$ is Σ_i^B-definable in a theory T if and only if there is a Σ_i^B formula $\phi(X,Y)$ such that

$$Y = F(X) \leftrightarrow \phi(X,Y)$$

and T proves

$$\forall X \exists! Y \phi(X,Y).$$

To define a search problem (multi-function) the only the first condition is used.

Then the important results are the witnessing theorems for V^i and TV^i.

Theorem 2.2.4. • *A function F is Σ_{i+1}^B-definable in V^{i+1} if and only if F is in $FP^{\Sigma_i^p}$ [4].*

- *A function F is Σ^B_{i+1}-definable in TV^i if and only if F is in $FP^{\Sigma^p_i}$ [2].*
- *A search problem P is Σ^B_{i+1}-definable in TV^{i+1} if and only if P is in $PLS^{\Sigma^p_i}$ [3].*
- *A function F is Σ^B_{i+1}-definable in V^i if and only if F is in $FP^{\Sigma^p_{j-1}}[wit, log]$ [21].*
- *For $j > i+1$, a function F is Σ^B_j-definable in TV^i or V^i if and only if F is in $FP^{\Sigma^p_{j-1}}[wit, O(1)]$ [36].*

2.3 The Quantified Propositional Calculus

We are also interested in quantified propositional proof systems. The proof systems we use were originally defined in [23], and then they were redefined in [10, 29], which is the presentation we follow.

The set of connectives are $\{\wedge, \vee, \neg, \exists, \forall, \top, \bot\}$, where $\top$ and $\bot$ are constants for true and false, respectively. Formulas are built using these connectives in the usual way. We will often refer to formulas by the number of quantifier alternations.

Definition 2.3.1. The set of formulas $\Sigma^q_0 = \Pi^q_0$ is the set of quantifier-free propositional formulas. For $i > 0$, the set of Σ^q_i (Π^q_i) formulas is the smallest set of formulas that contains Π^q_{i-1} (Σ^q_{i-1}) and is closed under $\wedge$, $\vee$, existential (universal) quantification, and if $A \in \Pi^q_i$ ($A \in \Sigma^q_i$) then $\neg A \in \Sigma^q_i$ ($\neg A \in \Pi^q_i$).

The first proof system, from which all others will be defined, is the proof system G. This proof system is a sequent calculus based on Gentzen's system LK. The system G is essentially the DAG-like, propositional version of LK. A sequent is two series of formulas, which we will write as

$$\Gamma \to \Delta.$$

The intended meaning is if every formula in Γ is true then at least one of the formulas in Δ is true. In a G proof, a sequent is derived from other sequents using one of the rules of inference:

- Weakening rules:

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

- Exchange rules:

$$\frac{\Gamma_1, A, B, \Gamma_2 \to \Delta}{\Gamma_1, B, A, \Gamma_2 \to \Delta} \qquad \frac{\Gamma \to \Delta_1, A, B, \Delta_2}{\Gamma \to \Delta_1, B, A, \Delta_2}$$

- Contraction rules:

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

- $\neg$ introduction rules:

$$\frac{\Gamma \to \Delta, A}{\neg A, \Gamma \to \Delta} \qquad \frac{A, \Gamma \to \Delta}{\Gamma \to \Delta, \neg A}$$

- $\wedge$ introduction rules:

$$\frac{A, B, \Gamma \to \Delta}{A \wedge B, \Gamma \to \Delta} \qquad \frac{\Gamma \to \Delta, A \qquad \Gamma \to \Delta, B}{\Gamma \to \Delta, A \wedge B}$$

- $\vee$ introduction rules:

$$\frac{A, \Gamma \to \Delta \qquad B, \Gamma \to \Delta}{A \vee B, \Gamma \to \Delta} \qquad \frac{\Gamma \to \Delta, A, B}{\Gamma \to \Delta, A \vee B}$$

A rule of special interest is the cut rule. This rule is not needed to have a complete proof system, but it is used to make proofs shorter. For this reason, it plays a special role in proof complexity. The cut rule is

$$\text{cut}\ \frac{A, \Gamma \to \Delta \qquad \Gamma \to \Delta, A}{\Gamma \to \Delta}$$

In this rule, we call A the cut formula. There are also four rules that introduce quantifiers:

$$\exists\text{-left}\ \frac{A(x), \Gamma \to \Delta}{\exists z A(z), \Gamma \to \Delta} \qquad \exists\text{-right}\ \frac{\Gamma \to \Delta, A(B)}{\Gamma \to \Delta, \exists z A(z)}$$

$$\forall\text{-left}\ \frac{A(B), \Gamma \to \Delta}{\forall z A(z), \Gamma \to \Delta} \qquad \forall\text{-right}\ \frac{\Gamma \to \Delta, A(x)}{\Gamma \to \Delta, \forall z A(z)}$$

These rules have conditions on them. In $\exists$-left and $\forall$-right, the variable x must not appear in the bottom sequent. In these rules, x is called the eigenvariable. In the other two rules, the formula B must be a Σ_0^q formula, and no variable that appears free in B can be bound in $A(z)$.

The initial sequents of G are sequents of the form $\to \top$, $\bot \to$, or $x \to x$, where x is any propositional variable. A G proof is a series of sequents such that each sequent is either an initial sequent or can be derived from previous sequents using one of the rules of inference. The proof system G_i is G with cut formulas restricted to Σ_i^q formulas.

We define G^* as the treelike version of G. So, a G^* proof is a G proof where each sequent is used as an upper sequent in an inference at most once. A G_i^* proof is a G^* proof in which cut formulas are prenex Σ_i^q. In [29], it was shown that, for treelike proofs, it did not matter if the cut formulas in G_i^* were prenex or not. So when we construct G_i^* proofs the cut formulas will not always be prenex, but that does not matter.

To make proofs simpler, we assume that all treelike proofs are in *free-variable normal form.*

Definition 2.3.2. A parameter variable for a G_i^* proof π is a variable that appears free in the final sequent of π. A proof π is in *free-variable normal form* if (1) every non-parameter variable is used as an eigenvariable exactly once in π, and (2) parameter variables are not used as eigenvariables.

Note that, if a proof is treelike, we can always put it in free-variable normal form by simply renaming variables. In fact, VPV proves that every treelike proof can be put in free-variable normal form.

A useful property of these proof systems is the *subformula property*. It can be shown in VPV that every formula in a G_i^* proof is an ancestor (and therefore a subformula) of a cut formula or a formula in the final sequent. This is useful because it tells us that any non-Σ_i^q formula in a G_i^* proof must be an ancestor of a final formula.

2.3.1 Propositional Translations

There is a close connection between the theory V^i and the proof system G_i^*. You can think of G_i^* as the non-uniform version of V^i. This idea might not make much sense at first until you realize you can translate a V^i proof into a polynomial-size family of G_i^* proofs. The translation that we use is described in [8, 10]. It is a modification of the Paris-Wilkie translation [33]. Given a Σ_i^B formula $\phi(\vec{x}, \vec{X})$ over the language $\mathcal{L}_A^2$, we want to translate it into a family of propositional formulas $||\phi(\vec{x}, \vec{X})||[\vec{m}; \vec{n}]$, where the size of the formulas is bounded by a polynomial in $\vec{m}$ and $\vec{n}$. The formula $||\phi(\vec{x}, \vec{X})||[\vec{m}; \vec{n}]$ is meant to be a formula that is a tautology when $\phi(\vec{x}, \vec{X})$ is true in the standard model whenever $x_i = m_i$ and $|X_i| = n_i$. If $\phi(\vec{x}, \vec{X})$ is true in the standard model for all $\vec{x}$ and $\vec{X}$, then every $||\phi(\vec{x}, \vec{X})||[\vec{m}; \vec{n}]$ is a tautology.

The variables $\vec{m}$ and $\vec{n}$ will often be omitted since they are understood. The free variables in the propositional formula will be $p_j^{X_i}$ for $j < n_i - 1$. The variable $p_j^{X_i}$ is meant to represent the value of the jth bit of X_i; we know that the n_ith bit is 1, and for $j > n_i$, we know the jth bit is 0. The definition of the translation proceeds by structural induction on ϕ.

Suppose ϕ is an atomic formula. Then it has one of the following forms: $s = t$, $s < t$, $X_i(t)$, or one of the trivial formulas $\bot$ and $\top$, for terms s and t. Note that the terms s and t can be evaluated immediately. This is because the exact value of every number variable and the size of each string variable is known. Let $val(t)$ be value of the term t.

In the first case, we define $||s = t||$ as the formula $\top$, if $val(s) = val(t)$, and $\bot$, otherwise. A similar construction is done for $s < t$. If ϕ is one of the trivial formulas, then $||\phi||$ is the same trivial formula. So now, if $\phi =_{syn} X_i(t)$, let $j = val(t)$. Then the translation is defined as follows:

$$||\phi|| =_{syn} \begin{cases} p_j^{X_i} & \text{if } j < n_i - 1 \\ 1 & \text{if } j = n_1 - 1 \\ 0 & \text{if } j > n_1 - 1 \end{cases}$$

Now for the inductive part of the definition. Suppose $\phi =_{syn} \alpha \wedge \beta$. Then

$$||\phi|| =_{syn} ||\alpha|| \wedge ||\beta||.$$

When the connective is $\vee$ or $\neg$, the definition is similar. If the outermost connective is a number quantifier bound by a term t, let $j = val(t)$. Then the translation is defined as

$$||\exists y \leq t, \alpha(y)|| =_{syn} \bigvee_{i=0}^{j} ||\alpha(y)||[i]$$

$$||\forall y \leq t, \alpha(y)|| =_{syn} \bigwedge_{i=0}^{j} ||\alpha(y)||[i]$$

$$||\exists Y \leq t, \alpha(Y)|| =_{syn} \exists p_0^Y, \ldots, \exists p_{m-2}^Y, \bigvee_{i=0}^{j} ||\alpha(Y)||[i]$$

$$||\forall Y \leq t, \alpha(Y)|| =_{syn} \forall p_0^Y, \ldots, \forall p_{m-2}^Y, \bigwedge_{i=0}^{j} ||\alpha(Y)||[i]$$

Now we are able to state the translation theorem for V^i and G_i^*.

Theorem 2.3.3 ([22, 29]). *Let $i > 1$. Suppose $V^i \vdash \phi(\vec{x}, \vec{X})$, where ϕ is a bounded formula. Then V^1 proves there are polynomial-size G_i^* proofs of the family of tautologies*

$$||\phi(\vec{x}, \vec{X})||[\vec{m}; \vec{n}].$$

This is the two-sorted version of the translation theorems from Section 9.2 in [22], and was proved in [11]. Note that the same theorem holds for TV^{i-1} in place of V^i since TV^{i-1} is a sub-theory of V^i.

This type of theorem is the standard way of proving that the reasoning power of the proof system is as least as strong as that of the theory. One way of viewing this is by looking at the complexity of the witnessing functions. The theorem above tells us that witnessing theorems of V^i is at least as hard as witnessing G_i^* proofs. The idea is that to witness a theorem of V^i $\phi(X)$ for a given X, we can construct a G_i^* proof of $||\phi(X)||[; |X|]$ in polynomial-time. Then we can witness ϕ with a given input by witnessing this proof.

2.3.2 Truth Definition

In order to reason about the proof systems in the theories, we must be able to reason about quantified propositional formulas. We follow the presentation in [22, 23].

Formally formulas will be coded as strings, but we will not distinguish between a formula and its encoding. So if F is a formula, we will use F as the string encoding the formula as well. The method of coding a formula can be found in [10]. The encoding of an assignment A will be a set of pairs $\langle i, 0\rangle$ and $\langle i, 1\rangle$ which mean that the variable x_i is assigned false and true, respectively.

The truth definition we give will be more complicated than usual, but this is because we want to be able to write the definition as a formula in the language of bounded arithmetic. To this end, the main part of the definition will be contained in two formulas: $eval_i^{\exists}(E, A, X, F)$ and $eval_i^{\forall}(E, A, Y, F)$. The formula $eval_i^{\exists}(E, A, X, F)$ is intended to be a formula that says E is an evaluation that shows that the Σ_i^q formula F is true or the Π_i^q formula F is false. The formula is evaluated by structural induction. For example, if $F \equiv F_1 \wedge F_2$, then E is a combination of an evaluation of F_1 and F_2. The base case of this induction is when we reach a Π_{i-1}^q formula or a Σ_{i-1}^q formula. At this point, we resort to an induction on i. In essence, we have a simultaneous induction on the structure of F (taken care of by E) and on the quantifier complexity. The formula $eval_i^{\forall}(E, A, Y, F)$

is similar. The main difference is that E is supposed to be an evaluation showing a Π_i^q formula is true or a Σ_i^q formula is false.

Given a Σ_i^q formula F, an evaluation of that formula will be a series of lines. Each line will consist of a truth value and a subformula of F. Plus each line will have to be consistent with previous lines. For example, if there is a line saying $F_1 \vee F_2$ is true, then there is an earlier line saying F_1 is true or a line saying F_2 is true. With this in mind, we have the following definition.

Definition 2.3.4. Let F be the encoding of a formula where all quantified variables are distinct, and different than the free variables. We recursively define $A \models_i F$. If F is a Σ_i^q formula, then

$$A \models_i F \equiv \exists X \exists E \; eval_i^{\exists}(E, A, X, F) \wedge \exists n < |E|, E^{[n]} = \langle \top, F \rangle,$$

where $eval_i^{\exists}(E, A, X, F)$ is a formula saying that E is a series of lines assigning truth values to subformulas of F and X assigns values to the outermost even existential quantifiers and the outermost odd universal quantifiers. An even quantifier is one that is in the scope of an even number of $\neg$, and an odd quantifier is one that is in the scope of an odd number of $\neg$. More formally $eval_i^{\exists}(E, A, X, F)$ is the conjunction of the following. Note that we do not give the bounds on the quantified variables, but the reader can fill in what they should be. As well, the quantification over the subformulas of F is written as quantification over string variables (i.e. $\forall F_1$); however, this can be replaced by quantifying over the position in F where these formulas appear. This turns it into a number quantifier.

- For every line l, if the outermost connective of the formula is $\wedge$ and the formula is true, then there are earlier lines j_1, j_2 saying both the left subformula F_1 and the right subformula F_2 are true.

$$\forall l \forall F_1 \forall F_2 \exists j_1 \exists j_2,$$
$$E^{[l]} = \langle \top, F_1 \wedge F_2 \rangle \supset E^{[j_1]} = \langle \top, F_1 \rangle \wedge E^{[j_2]} = \langle \top, F_2 \rangle$$

Note that the case of $\vee$ with a false formula is handled the same way.

- For every line l, if the outermost connective of the formula is $\wedge$ and the formula is false, then there is an earlier line j saying one of the left subformula F_1 or right subformula F_2 is false.

$$\forall l \forall F_1 \forall F_2 \exists j,$$
$$E^{[l]} = \langle \bot, F_1 \wedge F_2 \rangle \supset E^{[j]} = \langle \bot, F_1 \rangle \vee E^{[j]} = \langle \bot, F_2 \rangle$$

Note that the case of $\vee$ with a true formula is handled the same way.

- For every line l, if the outermost connective of the formula is $\neg$ and the formula is true, then there is a previous line j saying the subformula is false.

$$\forall l \forall F_1 \exists j,$$
$$E^{[l]} = \langle \top, \neg F_1 \rangle \supset E^{[j]} = \langle \bot, F_1 \rangle$$

Note that this is the only case where the truth value changes, so the truth value can also be viewed as the parity of the number of negations that were passed to reach this subformula.

- For every line l, if the outermost connective of the formula is $\exists$ and the formula is true, then there is a previous line j with a witness for the quantifier and X gives us that value.

$$\begin{aligned}&\forall l \forall F_1 \exists j,\\ &E^{[l]} = \langle \top, \exists x_n F_1(x_n)\rangle \supset (E^{[j]} = \langle \top, F_1(x_n)\rangle \wedge (\langle n, 0\rangle \in X \vee \langle n, 1\rangle \in X))\end{aligned}$$

Note that the case of $\forall$ with a false formula is handled the same way.

- For every line l, if the outermost connective of the formula is $\exists$ and the formula is false, then the formula is a Σ^q_{i-1} formula, and it is false according to $\models_{i-1}$.

  $$\begin{aligned}&\forall l \forall F_1,\\ &E^{[l]} = \langle \perp, \exists y_n F_1(y_n)\rangle \supset \exists y_n F_1(y_n) \in \Sigma^q_{i-1} \wedge (X \cup A) \models_{i-1} \exists y_n F_1(y_n)\end{aligned}$$

  This is the base case in the structural induction, and this is the point at which we invoke the induction on the quantifier complexity. Note that the case of $\forall$ with a true formula is handled the same way.

- For every line l, if the formula is a single variable, then the truth value is consistent with A.

  $$\begin{aligned}&\forall l,\\ &E^{[l]} = \langle \top, x_n\rangle \supset \langle x_n, 1\rangle \in A\\ \wedge &E^{[l]} = \langle \perp, x_n\rangle \supset \langle x_n, 0\rangle \in A\end{aligned}$$

If F is a Π^q_i formula, then

$$A \models_i F \equiv \forall Y \forall E\ eval^{\forall}_i(E, A, Y, F) \supset \exists n E^{[n]} = \langle \top, F\rangle$$

where $eval_i^{\forall}(E, A, Y, F)$ is almost the same as $eval_i^{\exists}$ except Y now gives a truth value for the even universally-quantified variables and the odd existentially-quantified variables.

Notice that in $eval_i^{\exists}$ if the outermost connective is $\forall$ and we want to falsify it, then it is treated like $\exists$. The connectives $\wedge$ and $\vee$ are also treated the same when we are trying to satisfy one and falsify the other. When we see a $\forall$ and we want to satisfy the formula, we know the quantifier complexity of the formula has dropped. Therefore, we can get the value of this formula recursively. If we are looking at a Σ_0^q formula the recursive case never comes up.

For a Σ_i^q formula, we are saying there is an evaluation of the formula that says it is true. For a Π_i^q formula, we are saying that all evaluations of the formula say it is true. This is an important difference since a Σ_i^q formula is false if there is no evaluation of the formula, but a Π_i^q formula would be true.

For $i > 0$, this gives a Σ_i^B definition for $A \models_i F$ and, for $i = 0$, it has a $\Sigma_0^B(PV)$ definition in VPV. If F is a Π_i^q formula, the definition is Π_i^B.

Given a formula $F \equiv \bigwedge_{i=0}^{n} F_i$, there is a PV function $Parse_\wedge(F, j)$ that outputs $F_{min(j,n)}$. The same goes for $\vee$ in place of $\wedge$. The theory VPV proves the Tarski conditions for the truth definition.

Lemma 2.3.5 (Tarski's Conditions). *VPV proves the following*

1. $(A \models_i F_1 \wedge F_2) \leftrightarrow (A \models_i F_1 \wedge A \models_i F_2)$
2. $(A \models_i F_1 \vee F_2) \leftrightarrow (A \models_i F_1 \vee A \models_i F_2)$
3. $(A \models_i F) \leftrightarrow (\forall j \leq |F|\ A \models_i Parse_\wedge(F, j))$ *(where* $F \equiv \bigwedge_{j=0}^{n} F_j$ *and* $F \in \Pi_i^q$*)*
4. $(A \models_i F) \leftrightarrow (\exists j \leq |F|\ A \models_i Parse_\vee(F, j))$ *(where* $F \equiv \bigvee_{j=0}^{n} F_j$ *and* $F \in \Sigma_i^q$*)*
5. $(A \models_i \neg F) \leftrightarrow \neg(A \models_i F)$
6. $(A \models_i \exists \vec{x} F(\vec{x})) \leftrightarrow \exists X (A \cup X \models_i F(\vec{x}))$ *(for* $F \in \Sigma_i^q$*)*

7. $(A \models_i \forall \vec{x} F(\vec{x})) \leftrightarrow \forall X (A \cup X \models_i F(\vec{x}))$ *(for* $F \in \Pi_i^q$*)*

8. $(A \models_i F) \leftrightarrow (A \models_{i-1} F)$ *(for* $F \in \Sigma_{i-1}^q \cup \Pi_{i-1}^q$*).*

Proof. (1) Suppose $A \models_i F_1 \wedge F_2$ and the formula is Σ_i^q, then there is an evaluation of this formula. This evaluation would contain the line $(\top, F_1 \wedge F_2)$. Therefore this evaluation would also contain lines of the form $(\top, F_1)$ and $(\top, F_2)$. This means we have evaluations of F_1 and F_2. Suppose $A \models F_1 \wedge A \models F_2$. Then there exist evaluations of these formulas. An evaluation for $F_1 \wedge F_2$ is obtained from these evaluations by combining these evaluations and adding a new line using Σ_0^B-COMP. If the formula is Π_i^q, the proof is similar.

(2) The same way as (1).

(3) Suppose $A \models_i F$ is false and $\forall j \leq |F|\ A \models_i Parse_\wedge(F, j)$ is true, where $F \equiv \bigwedge_{j=0}^{n} F_j$. Let F^m be the formula $\bigwedge_{j=0}^{m} F_j$. This means $F^m \equiv F^{m-1} \wedge F_m$. By the first assumption, there is an evaluation of $F \equiv F^n$ with a line $\langle \bot, F \rangle$. If there is a line in the evaluation of the form $\langle \bot, F^{m+1} \rangle$, then there is a line $\langle \bot, F^m \rangle$. Note that $\langle \bot, F_m \rangle$ cannot appear in a line by the second assumption. So it follows by induction that there is a line saying that $F^1 \equiv F_1$ is false, but this contradicts the second assumption.

Now suppose $\forall j \leq |F|\ A \models_i Parse_\wedge(F, j)$ is false. Then there exists an evaluation of F_j, for some j, with a line $\langle \bot, F_j \rangle$. To this evaluation we can append the lines $\langle \bot, F^m \rangle$ for $j \leq m \leq n$ using Σ_0^B-COMP. This shows that $A \models_i F$ is false.

Note that the proof of (3) does not work if F is a Σ_i^q formula since $A \models_i F$ could be false because there is no evaluation of F. There is not necessarily an evaluation that shows F is false.

(4) This is the dual of (3).

(5) We assume F is a Π_i^q formula. The case where F is a Σ_i^q formula is essentially the same. Suppose $(A \models_i \neg F)$. Then there exists an evaluation of F with a line $\langle \top, \neg F \rangle$. This means the evaluation also has a line $\langle \bot, F \rangle$, proving $\neg(A \models_i F)$. Suppose

$\neg(A \models_i F)$. There there exists an evaluation of F with line $\langle \perp, F\rangle$. The line $\langle \top, \neg F\rangle$ can be appended to this evaluation, proving $(A \models_i \neg F)$.

(6) This follows directly from the $\exists X$ in the definition of $\models_i$.

(7) This is the dual of (6).

(8) This follows directly from the recursive nature of the definition. □

The importance of the Tarski conditions is that they give us the ability to say that a formula if true if and only if it evaluates to true. This is best stated in the propositional proof systems. Consider the formula $A \models_i F(\vec{x})$, where A codes an assignment and F is a formula over the variables $\vec{x}$. If we take the propositional translation of this formula, then we get a formula with free variables for the bits of the string A and F. In this case we are dealing with a specific formula, we can replace the bits of the formula F by its actual encoding. As for A, it is supposed to be a string that codes the assignment to $\vec{x}$. We can replace the variables that are the bits of A by formulas that encode the assignment to $\vec{x}$ given $\vec{x}$. For example, we replace $p^A_{i,0}$ by $\neg x_i$ (if x_i is assigned false, then $A(i,0)$ is true). Using this, we get the following lemma.

Lemma 2.3.6. *V^1 proves that there are polynomial-size G^*_i proofs of*

$$B(\vec{x}) \leftrightarrow C(\vec{x}),$$

where $\vec{B}(\vec{x})$ is the propositional translation of $A \models_i F(\vec{x})$ with the variables for A replaced by formulas coding $\vec{x}$ as an assignment and the variables for F are replaced by the constants that code C.

Proof. Structural induction on C using the fact that V^1 proves the Tarski condition for the true definition (which translates into polynomial-size G^*_1 proofs). See Lemma 9.3.15 in [22] for a similar lemma. □

Valid formulas (or tautologies) are defined as

$$TAUT_i(F) \equiv \forall A, (\text{“A is an assignment to the variables of } F\text{”} \supset A \models_i F)$$

This truth definition can be extended to define the truth of a sequent. So, if $\Gamma \rightarrow \Delta$ is a sequent of $\Sigma_i^q \cup \Pi_i^q$ formulas, then

$$\begin{aligned}(A \models_i \ \Gamma \rightarrow \Delta) \equiv &\text{“there exists a formula in } \Gamma \text{ that } A \text{ does not satisfy”} \\ &\vee \ \text{“there exists a formula in } \Delta \text{ that } A \text{ satisfies”}\end{aligned}$$

Another important formula we will use is the reflection principle for a proof system. We define the Σ_i^q reflection principle for a proof system P as

$$\Sigma_i^q\text{-RFN}(P) \equiv \forall F \forall \pi, (\text{“}\pi \text{ is a } P \text{ proof of } F\text{”} \wedge F \in \Sigma_i^q) \supset TAUT_i(F)$$

This formula essentially says that, if there exists a P proof of a Σ_i^q formula F, then F is valid. Another way of putting it is to say that P is sound when proving Σ_i^q formulas.

The usefulness of the reflection principles is found in the connection with bounded arithmetic. This is demonstrated in the following theorem.

Theorem 2.3.7.
- *V^i proves Σ_i^q-RFN(G_i^*) [26].*
- *The Σ_i^B consequences of V^i can be axiomatized by $V^1 + \Sigma_i^q$-RFN(G_i^*) [26].*

This tells us a lot about the complexity of the witnessing problem for G_i^*. Essentially the complexity of witnessing a G_i^* proof of a Σ_i^q formula is no harder than witnessing a Σ_i^B theorem of V^i. This is because the existential part of the reflection principle is a witness to the formula that is being proved.

Part II

The Fragments of G

Chapter 3

Proof Theory for G

3.1 Basic Constructions

In this section, we want to give a few simple results that hopefully will allow the reader to become familiar with the proof systems. These constructions are not new. They have come up in many proof theoretic proofs in different settings. The first result shows how we can remove a connective from the formula being proved. Note that we only mention connectives on the right, but there are corresponding connectives for the formulas on the left side of the sequent.

Lemma 3.1.1. *Let π be a G_i^* proof of a sequent*

$$\Gamma \rightarrow \Delta, A. \tag{3.1.1}$$

1. *If $A \equiv B \wedge C$ and D be B or C, then there exists a G_i^* proof π', where $|\pi'| < |\pi|$, of the sequent*

$$\Gamma \rightarrow \Delta, D.$$

2. *If $A \equiv B \vee C$, then there exists a G_i^* proof π', where $|\pi'| < |\pi|$, of the sequent*

$$\Gamma \to \Delta, B, C.$$

3. *If $A \equiv \neg B$, then there exists a G_i^* proof π', where $|\pi'| < |\pi|$, of the sequent*

$$B, \Gamma \to \Delta$$

4. *If $A \equiv \forall x B(x)$, then there exists a G_i^* proof π', where $|\pi'| < |\pi|$, of the sequent*

$$\Gamma \to \Delta, B(q)$$

where q is any variable that does not appear in Γ or Δ.

Proof. Each of these is proved the same way. They can be proved by induction on the depth of the proof. For example, consider 4 above. If the last inference in π does not have A as the principal formula, then π can be constructed using the same rule of inference with the sequent obtained from the induction hypothesis. If A was the principal formula, then the upper sequent in the last inference is

$$\Gamma \to \Delta, B(q).$$

This is the sequent we want. If we want q to be a different variable, we can replace every instance of q in the proof by the desired variable. □

Observe that in the above proof we did not include a case for $\exists$. This is because this case is actually not as simple. Consider an $\exists$-right inference:

$$\frac{\Gamma \to \Delta, B(C)}{\Gamma \to \Delta, \exists x B(x)}$$

The formula C could contain variables that are later used as eigenvariables. If we ignore this rule–which is essentially what was done in the proof above–the later $\forall$-right rules or $\exists$-left rules may no longer meet the eigenvariable condition. This is an important issue in our adaptation of the Herbrand theorem we prove later.

Note that we do not add to the proof. For example, in the $\forall$ case, we do not add a proof of $\forall x B(x) \to B(q)$ and cut $\forall x B(x)$ because it does not work. This is because it is possible that this formula is not Σ_i^q and cannot be cut in G_i^*. This is an important point because certain constructions that work in settings where every formula can be cut may not work here.

The next construction that is very often used is the substitution of formulas for free variables.

Lemma 3.1.2. *Let π be a G_i^* proof of a sequent*

$$\Gamma(x) \to \Delta(x),$$

and let C be any Σ_0^q formula that does not mention any of the eigenvariables π. Then there is a G_i^* proof of

$$\Gamma(C) \to \Delta(C)$$

those size is polynomial in the size of C and π.

Proof. Again this is done by induction on the depth of the proof. For the base case, we have a proof of $x \to x$, which becomes a proof of $C \to C$. Every other case follows easily by induction. □

Note that this proof may not work if C is not Σ_0^q, for, if π has a Σ_i^q cut formula that contains x, then, when x is replaced by C, the cut formula may not longer be Σ_i^q.

The final construction has to do with witnessing a single variable in a formula.

Lemma 3.1.3. *Let $B(q)$ be a Σ_i^q or Π_i^q formula. Then there exists polynomial-size G_i^* proofs of the sequents*

$$e \leftrightarrow B(\bot), B(e) \rightarrow \forall q B(q)$$

$$e \leftrightarrow B(\top), \exists q B(q) \rightarrow B(e)$$

Proof. The proof for the two sequents are essentially the same, so we only give the construction for the first one. Informally, the reason the first sequent is true is that we are picking a value for e that makes $B(e)$ false if possible. So, if $B(\bot)$ is false, we make e false, otherwise we make e true, which is the only other possible value.

First, it is possible to get cut-free proofs of the following four sequents.

$$e, B(e) \rightarrow B(\top)$$

$$B(e) \rightarrow B(\bot), e$$

$$e, B(\top) \rightarrow B(e)$$

$$B(\bot) \rightarrow B(e), e$$

This can be shown by simultaneous structural induction on the formula $B(e)$.

We use this in the following derivation:

$$\text{Cut } q \, \dfrac{q, B(\top) \rightarrow B(q) \qquad B(\bot) \rightarrow B(q), q}{\forall\text{-right} \, \dfrac{B(\bot), B(\top) \rightarrow B(q)}{B(\bot), B(\top) \rightarrow \forall q B(q)}}$$

So to finish proving this lemma, all we need are proofs of the sequents

$$e \leftrightarrow B(\bot), B(e) \rightarrow B(\bot)$$

$$e \leftrightarrow B(\bot), B(e) \rightarrow B(\top)$$

We prove the first one as follows:

$$\text{Cut e} \, \dfrac{e, e \supset B(\bot) \rightarrow B(\bot) \qquad B(e) \rightarrow B(\bot), e}{\text{Weakening and } \vee\text{-left} \, \dfrac{e \supset B(\bot), B(e) \rightarrow B(\bot)}{e \leftrightarrow B(\bot), B(e) \rightarrow B(\bot)}}$$

The second one is proved as follows:

$$\dfrac{\dfrac{B(e) \to e, B(\bot) \qquad B(\bot) \supset e, B(\bot) \to e}{B(\bot) \supset e, B(e) \to e}\,\text{Cut } B(\bot) \qquad e, B(e) \to B(\top)}{B(\bot) \supset e, B(e) \to B(\top)}\,\text{Cut } e$$

Note that the only cut formulas are e, $B(\bot)$, and $B(\top)$; therefore, the proof is a G_i^* proof. □

3.2 Herbrand Theorem for G_i^*

In bounded arithmetic, a useful tool has been the KPT witnessing theorem [24]. In the simplest case, the KPT witnessing theorem describes how to witness the Σ_2^B theorems of VPV. The original theorem was more general, but we state it here for the simplest case.

Theorem 3.2.1 (KPT Witnessing [24]). *Suppose VPV proves*

$$\forall X \exists Y \forall Z \phi(X, Y, Z),$$

where ϕ is a Σ_0^B formula. Then there exists a finite sequence of $\mathcal{L}_{FP}$ function symbols $F_1, F_2, \ldots, F_k$ such that

$$\begin{aligned} VPV \vdash \forall X \forall W \quad & \phi(X, F_1(X), W^{[1]}) \\ & \vee\ \phi(X, F_2(X, W^{[1]}), W^{[2]}) \\ & \quad \vdots \\ & \vee\ \phi(X, F_k(X, W^{[1]}, W^{[2]}, \ldots, W^{[k-1]}), W^{[k]}) \end{aligned}$$

Informally, this can be viewed as an interactive computation between a student, who runs in polynomial time, and an all-knowing teacher. Given a value for X, the student's goal is the find a witness for $\exists Y \forall Z \phi(X, Y, Z)$. The student starts by computing $F_1(X)$. If that is not a witness, the teacher responds with a counter example $W^{[1]}$. Using that the student makes a second guess by computing F_2. The teacher responds with $W^{[2]}$, and this process continues.

Our goal is to get a similar theorem for G_1^*, and to extend this to G_i^*. The rest of this section is organized as follows. We start by stating the analog of the above theorem for G_1^*. Using this as a starting point, we then define the concepts needed to prove this theorem. Our presentation will be based on a proof of the Herbrand Theorem. We then prove an analog of the Herbrand Theorem for G_1^*, and as a corollary we get a proof of the KPT Witnessing Theorem for G_1^*. In the second subsection, we explain how to generalize the Herbrand Theorem for G_1^* so it works for G_i^*.

3.2.1 Witnessing for G_1^*

In adapting the KPT Witnessing Theorem for G_1^*, the first obstacle comes in the statement of the theorem. The theory VPV has access to function symbols that correspond to the polynomial-time functions, but, in G_1^*, there are no function symbols. To fix this, we use the idea of an extension cedent from [11].

Definition 3.2.2. An *extension cedent* is a series of formulas of the form

$$e_1 \leftrightarrow E_1, e_2 \leftrightarrow E_2, \ldots, e_n \leftrightarrow E_n$$

such that E_i is a Σ_0^q formula that does not mention the variables $e_i, e_{i+1}, \ldots, e_n$. We say that e_i depends on a variable q if E_i mentions q or E_i mentions a variable that depends on q.

Observe that an extension cedent is really a description of a circuit, and that polynomial-size circuits are the nonuniform version of polynomial-time functions. So extension cedents replace the functions.

Theorem 3.2.3 (KPT Witnessing for G_1^*)**.** *There exists a $\mathcal{L}_{FP}$ (polynomial-time) function F such that VPV proves the following. Let π be a G_1^* proof of a prenex Σ_2^q formula*

$A(\vec{p}) \equiv \exists \vec{x} \forall \vec{y} B(\vec{x}, \vec{y}, \vec{p})$, where $B(\vec{x}, \vec{y}, \vec{p})$ is a Σ_0^q formula with all free variables shown. Then, given π, F outputs a G_0^* proof of a sequent $\Lambda \rightarrow \Theta$ where

1. *Θ is a series of n formulas of the form $B(\vec{e^i}, \vec{q^i}, \vec{p})$, where $\vec{e^i} \in E$ and $0 \leq i < n$*
2. *Λ is an extension cedent defining a new set of variables E in terms of $\vec{q}^0, \ldots, \vec{q}^{n-1}$ and $\vec{p}$,*
3. *$\vec{e^i}$ does not depend on $\vec{q^j}$, for $j \geq i$, and*
4. *$\vec{q^i}$ and $\vec{q^j}$ are disjoint when $i \neq j$.*

This theorem will be proved later as Theorem 3.2.10. Before we prove this theorem, notice that this is similar to the KPT Witnessing theorem for VPV. The row $W^{[i]}$ corresponds to $\vec{q^i}$, and F_i corresponds to the circuits defining $\vec{e^i}$. The major difference is that the number of rounds in the student-teacher game is not constant; it can grow polynomially in the size of the proof.

One way of proving the KPT Witnessing Theorem is to observe that it is a corollary to the Herbrand Theorem. So the idea behind our proof is to adjust the proof-theoretic proof of the Herbrand Theorem. See [5] Section 3 for an outline of the proof we use as a model. The main difference between our proof and that proof is that cut elimination cannot be used since it causes an exponential increase in the size of the proof. To get around this problem, we use the idea in [11] to prove that extended-Frege p-simulates G_1^*. The Σ_1^q cut formulas are turned into Σ_0^q cut formulas by witnessing the existential quantifiers with extension variables.

We prove the Herbrand Theorem for all Σ_i^q formulas, but before we can state the general theorem, we need a few definitions. The first one has more to do with notation. The q variables come from the eigenvariables in the G_1^* proof. To make it easier to refer to these variables, we use the following notation:

Notation. Let π be a G^* proof. Then the set Q_π will be the set of variables that are used as eigenvariables in π. If S is a sequent in π, then $Q_{\pi,S}$ will be the set of variables that are used as eigenvariables in the subproof of π ending with S. We will refer to $Q_{\pi,S}$ as Q_S when π is understood.

Note that π is treelike, and, if it is in free-variable normal form and S is derived from S_1 and S_2, then $Q_S = Q_{S_1} \cup Q_{S_2}$, and $Q_{S_1} \cap Q_{S_2} = \emptyset$.

The general witnessing theorem will be for G_1^* proofs of any formula A. In the end, we want a G_0^* proof of a sequent $\Lambda \longrightarrow A^*$, where A^* is an instance of of an $\vee$-expansion of A defined below.

From now on we assume quantifiers do not appear in the scope of a $\neg$. If we did not assume this, we would have to add a separate cases for when quantifiers appear in the scope of an odd number of quantifiers and an even number.

Definition 3.2.4 ($\vee$-expansion)**.** An $\vee$-expansion of a formula A is any formula that can be obtained from A by a finite number of applications of the following rule:

> If A^* is an $\vee$-expansion of A and B is a non-Σ_0^q subformula of A^*, then replacing B by $B \vee B'$, where B' is B with renamed quantified variables, in A^* yields another $\vee$-expansion of A. (α)

Note that A is an $\vee$-expansion of A.

An $\vee$-expansion is used as to handle formulas that are not prenex. The definition comes from [5].

Definition 3.2.5 ((Q, E)-instance)**.** Let Q and E be disjoint sets of variables. A (Q, E)-*instance* of a formula A is a quantifier-free formula A' obtained from A by replacing universally-quantified variables by distinct variables in Q and existentially-quantified variables by distinct variables in E, and deleting the quantifiers.

Example 3.2.6. If

$$A \equiv B_1 \wedge \exists x[B_2(x) \wedge \forall y B_3(x, y)]$$

and $B_2(x)$ is not a Σ_0^q formula, then

$$A^* \equiv B_1 \wedge \exists x[(B_2(x) \vee B_2'(x)) \wedge (\forall y B_3(x,y) \vee \forall y' B_3(x,y'))],$$

where $B_2'(x)$ is $B_2(x)$ with the quantifier variables renamed, is an $\vee$-expansion of A. This can be seen by replacing $B_2(x)$ and $\forall y B_3(x,y)$. We renamed the copy of y to y' to emphasis it is now a different quantified variable. If $q_1, q_2 \in Q$ and $e \in E$, then the formula

$$B_1 \wedge [(B_2(e) \vee B_2(e)) \wedge (B_3(e,q_1) \vee B_3(e,q_2))]$$

is a (Q,E)-instance of A^*, but

$$B_1 \wedge [(B_2(e) \vee B_2(e)) \wedge (B_3(e,q_1) \vee B_3(e,q_1))]$$

is not because y and y' were replaced by the same variable.

For another example, consider a prenex formula

$$\exists \vec{x}^1 \forall \vec{y}^1 \ldots \exists \vec{x}^n \forall \vec{y}^n B(\vec{x}^1, \vec{y}^1, \ldots, \vec{x}^n, \vec{y}^n),$$

where B is a Σ_0^q formula. Then an instance of an $\vee$-expansion of this formula is a formula of the form

$$B(\vec{e}^{1,1}, \vec{q}^{1,1}, \ldots, \vec{e}^{1,n}, \vec{q}^{1,n}) \vee \ldots \vee B(\vec{e}^{m,1}, \vec{q}^{m,1}, \ldots, \vec{e}^{m,n}, \vec{q}^{m,n}).$$

So in Theorem 3.2.3, the disjunction of the formulas in Θ is a (Q_π, E)-instance of A. Because of this, Theorem 3.2.3 is simply a special case of the Herbrand Theorem for G_1^* below.

Observe that in Theorem 3.2.3, there is an ordering on the variables. Namely the variables $\vec{q}^i$ come before the variables $\vec{q}^{i+1}$. We could also extend this ordering to include the extension variables. An extension variable would have to be larger than every variable it depends on. For the general case, we want something similar. To make the proof simpler, we will use $\prec$ to refer to this ordering. The ordering $\prec$ orders the eigenvariables Q and the extension variables E. Then A^* will be more than a (Q, E)-instance; it will be a $(Q, E, \prec)$-instance.

Definition 3.2.7. $(Q, E, \prec)$-instance Let B be a (Q, E)-instance of a formula A, and let $\prec$ be an ordering on $Q \cup E$. Then B is a $(Q, E, \prec)$-instance of A if $z_1 \prec z_2$ whenever z_2 replaces a quantified variable that is in the scope of the quantified variable that z_1 replaced.

Example 3.2.8. Take A^* from the previous example. Then

$$B \equiv B_1 \wedge [(B_2(e) \vee B_2(e)) \wedge (B_3(e, q_1) \vee B_3(e, q_2))]$$

is a (Q, E)-instance of A^*. If B is a $(Q, E, \prec)$-instance, then we know that $e \prec q_1$ and $e \prec q_2$ since $\forall y$ and $\forall y'$ are in the scope of the $\exists x$. Note that it does not matter if $q_1 \prec q_2$ or if $q_2 \prec q_1$ since $\forall y$ is not in the scope of $\forall y'$ and vice versa.

The idea of an instance is essentially the witnessing substitution from [5]. Now we are prepared to state the general theorem.

Theorem 3.2.9 (Herbrand Theorem for G_1^*)**.** *There exists a $\mathcal{L}_{FP}$ function F such that VPV proves the following. Let π be a G_1^* proof of A. Then, given π, F outputs a G_0^* proof of a sequent $\Lambda \longrightarrow A^*$ and a partial ordering $\prec$ of the variables $Q_\pi \cup E$, where E is a set of variables that do not appear in π, with the following properties:*

- *Λ is an extension cedent defining the variables in E in terms of Q_π and the free variables of A;*

- *for $e \in E$, if e depends on a variable $p \in Q_\pi \cup E$, then $p \prec e$; and*

- *A^* is a $(Q_\pi, E, \prec)$-instance of an $\vee$-expansion of A*

Proof. The G_0^* proof that we are looking for will be constructed by changing π one sequent at a time starting with the initial sequents and working our way down. To simplify this construction, we use the "multiplicative" form of two hypothesis rules instead of the "additive" form. For example, the multiplicative form of $\wedge$-right is

$$\frac{\Gamma_1 \rightarrow \Delta_1, A \qquad \Gamma_2 \rightarrow \Delta_2, B}{\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2, A \wedge B}$$

We use this form instead of the more standard form

$$\frac{\Gamma \rightarrow \Delta, A \qquad \Gamma \rightarrow \Delta, B}{\Gamma \rightarrow \Delta, A \wedge B}$$

This is something that was also done in [5]. The advantage of the multiplicative form is that, except for the principal formula, each formula in the bottom sequent has a single parent in the upper sequents. So, in essence, we have removed implicit contractions. We also ignore the order of the formulas in the sequents. So a sequent is a pair of multi-sets. One set for the left side of the sequent, and one set for the right side.

Let S be any sequent in π. By the subformula property of G_1^*, S is of the form

$$\Gamma \rightarrow \Delta, \Omega,$$

where Γ and Δ are possibly empty sets of Σ_1^q formulas that are not ancestors of the final formula and Ω is a possibly empty set of formulas that are ancestors of the final formula. Recall that we are assuming there are no quantifiers in the scope of any $\neg$. We want to define a $\mathcal{L}_{FP}$ function that outputs a G_0^* proof of a sequent

$$S' \equiv \Lambda, \Gamma' \rightarrow \Delta', \Omega',$$

and a partial ordering $\prec$ on $Q_S \cup E$ where

1. Γ' is obtained from Γ by replacing each formula $\exists\vec{z}D(\vec{z})$ by $D(\vec{q})$, where D is Σ_0^q and $\vec{q} \in Q_S$. (We use different $\vec{q}$ for different formulas.)

2. Δ' is obtained from Δ be replacing each formula $\exists\vec{z}D(\vec{z})$ by $D(\vec{e})$, where D is Σ_0^q and $\vec{e} \in E$. (We use different $\vec{e}$ for different formulas.)

3. Λ is an extension cedent defining E in terms of Q_S and the free variables of S;

4. for $e \in E$, if e depends on a variable $p \in Q_S \cup E$, then $p \prec e$;

5. Ω' is obtained from Ω by replacing each formula B by a $(Q_S, E, \prec)$-instance of an $\vee$-expansion of B; and

6. each $q \in Q_S$ appears in at most one formula in Γ', Δ', and Ω'.

Note that $\prec$ is only defined on the extension variables and eigenvariables used so far. Initially, $\prec$ is an ordering where nothing is comparable. As we move down the proof, we order the variables.

The proof is done by induction on the depth of S in the proof π. If we let S be the final sequent, we get a proof of the theorem since $Q_\pi = Q_S$, and conditions 3-5 are the conditions we need for the theorem. Also, note that the induction hypothesis can be stated as a $\Sigma_0^B(\mathcal{L}_{FP})$ formula (is a polynomial-time predicate) by saying that the output of the function F on the first i sequents of π meets all of the conditions. This means the induction can be carried out in VPV.

The description of F is done in cases. There is a separate case for each rule of inference. The construction is similar to the proof that extended-Frege p-simulates G_1^* (Theorem 7.48 of [11]). The difference is that the variables need to be ordered.

When the last inference is the $\vee$, $\wedge$, $\neg$ introduction rules, the same rule can be applied in the G_0^* proof we are constructing. In the case of the single parent rules, the ordering stays the same. For the two parent rules, the new ordering is the union of the two ordering obtained from the induction hypothesis with the parents. Note that if there are two parents, then eigenvariables and the extension variables in the two dijoint subproofs are disjoint since π is in free-variable normal form. From this, it is a simple exercise to check that the induction hypothesis still holds. The same would go for weakening. The other cases are more involved, and are given below.

Inductive Case 1: S is inferred using cut

Suppose

$$S \equiv \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2, \Omega_1, \Omega_2$$

and that it is derived from

$$S_1 \equiv \exists\vec{x}D(\vec{x}), \Gamma_1 \to \Delta_1, \Omega_1$$

and

$$S_2 \equiv \Gamma_2 \to \Delta_2, \exists\vec{x}D(\vec{x}), \Omega_2,$$

where $D(\vec{x})$ is a Σ_0^q formula. By induction, we have a G_0^* proof of the sequents

$$S_1' \equiv \Lambda_1(\vec{q}), D(\vec{q}), \Gamma_1' \to \Delta_1', \Omega_1'$$

with an ordering $\prec_1$ on $Q_{S_1} \cup E_1$ and

$$S_2' \equiv \Lambda_2, \Gamma_2' \to \Delta_2', D(\vec{e}), \Omega_2'$$

with an ordering $\prec_2$ on $Q_{S_2} \cup E_2$. In this case, we let

$$S' \equiv \Lambda_2, \Lambda_1(\vec{e}), \Gamma_1', \Gamma_2' \to \Delta_1', \Delta_2', \Omega_1', \Omega_2'.$$

Since π is in free-variable normal form, the eigenvariables and extension variables from the proofs of S_1 and S_2 are disjoint. The new ordering is done by extending the old orderings and making the variables from the proof of S_2 smaller than those from the proof of S_2. That is, $p_1 \prec p_2$ is true if any of the following conditions hold:

- $p_1 \prec_1 p_2$,
- $p_1 \prec_2 p_2$, or
- $p_1 \in Q_{S_2} \cup E_2$ and $p_2 \in Q_{S_1} \cup E_1$.

We can prove S' by taking the proof of S'_1 and replacing $\vec{q}$ with $\vec{e}$. Because the proofs are treelike, the substitution does not cause any problems. We can then do the cut with S'_1 and S'_2.

We now look at each part of the induction hypothesis to be sure it still holds. Properties 1 to 3 are obvious.

Let us prove property 4. Suppose $e \in E = E_1 \cup E_2$ depends on $p \in Q_S \cup E$. If $e \in E_2$, then e also depends on p in S'_2 since Λ_2 did not change. So, by induction with S_2, $p \prec_2 e$ and, therefore, $p \prec e$. Now suppose $e \in E_1$. If $p \in Q_{S_2} \cup E_2$, then $p \prec e$. If $p \notin Q_{S_2} \cup E_2$, then e depends on p in $\Lambda_1(\vec{q})$ since $\vec{e} \in E_2$. So, by induction with S_1, we get $p \prec_1 e$, which implies $p \prec e$.

Property 5 follows directly form the induction hypothesis. Property 6 follows from the induction hypothesis and the fact that Q_{S_1} and Q_{S_2} are disjoint.

Inductive Case 2: S is inferred using $\forall$-right (or $\exists$-left)

Suppose

$$S \equiv \Gamma \to \Delta, \forall y D(y), \Omega$$

and it is derived from

$$S_1 \equiv \Gamma \to \Delta, D(q), \Omega.$$

By induction with S_1, there exists a sequent S'_1 and an ordering $\prec_1$ satisfying the induction hypothesis. In S'_1, there is a formula $D^*(q)$ that is a $(Q_{S_1}, E, \prec_1)$-instance of an $\vee$-expansion of $D(q)$. That same formula is also a $(Q_S, E, \prec)$-instance of an $\vee$-expansion of $\forall y D(y)$ now that q is part of Q_S.

The construction for this case is fairly simple. We let S' be the same as S_1', and the ordering $\prec$ is the same as $\prec_1$ except q is now smaller than everything in $Q_{S_1} \cup E$. That is, q is the smallest variable of all variables ordered so far.

For *exists*-left, the construction is the same.

Inductive Case 3: S is inferred using $\exists$-right

Suppose

$$S \equiv \Gamma \rightarrow \Delta, \exists x D(x), \Omega$$

and it is derived from

$$S_1 \equiv \Gamma, \rightarrow \Delta, D(F), \Omega,$$

where F is a Σ_0^q formula.

By induction, we have

$$S_1' \equiv \Lambda, \Gamma' \rightarrow \Delta', D^*(F), \Theta$$

and the ordering $\prec_1$ on $Q_{S_1} \cup E$. Note that we use $D^*(F)$ in place of $D(F)$ because it is possible that $D(F)$ had quantifiers that have already been removed. Since F is a Σ_0^q formula, it would still be intact since (α) does not change Σ_0^q formulas. There are two cases to consider. If $D(F)$ is an ancestor of the final formula then $D^*(F)$ is an instance of an $\vee$-expansion of $D(F)$. If $D(F)$ is not an ancestor of the final formula, then $D^*(F)$ is $D(F)$ with the existential quantifiers replaced by extension variables. In either case, $D^*(F)$ is a Σ_0^q formula. The sequent S' will be the sequent

$$e \leftrightarrow F, \Lambda, \Gamma' \rightarrow \Delta', D^*(e), \Theta,$$

where e is a new extension variable. As for the ordering $\prec$, it is defined by extending $\prec_1$ by making e the minimum element. Note that $Q_S = Q_{S_1}$.

Let S_2' be

$$e \leftrightarrow F, D'(F) \rightarrow D^*(e).$$

It is easy to derive S_2'. Then it is possible to derive S' from S_1' and S_2' by cutting $D^*(F)$, which is a Σ_0^q formula.

We now look at each part of the induction hypothesis to be sure it still holds. It is easy to see that properties 1 to 3 and property 6 still hold.

For property 4, if e, the new extension variable, depends on p, then p must appear in F, which is part of S_1. This means that $p \notin Q_S \cup E$. So, property 4 holds for e. For other variables, it holds directly from the induction hypothesis.

For property 5, the only instance that changed is $D^*(F)$, assuming $\exists x D(x)$ is not a Σ_1^q formula. Since e replaced the outermost quantifier, e does not have to be larger than any variable, and it is smaller than every variable that replaced inside variables. Therefore $D^*(e)$ is a $(Q_S, E, \prec)$-instance of an $\vee$-expansion of $\exists x D(x)$.

Inductive Case 4: S is inferred using contraction-right

Suppose

$$S \equiv \Gamma \to \Delta, D, \Omega$$

and it is derived from

$$S_1 \equiv \Gamma \to \Delta, D, D, \Omega.$$

We look at two different cases: D is a Σ_1^q formula, D is not Σ_1^q.

For the first case, let D be $\exists \vec{x} D'(\vec{x})$, where $D'(\vec{x})$ is a Σ_0^q formula, then, by induction with S_1, we have

$$S_1' \equiv \Lambda, \Gamma' \to \Delta', D'(\vec{e}^1), D'(\vec{e}^2), \Omega'$$

with the ordering $\prec_1$ on $Q_{S_1} \cup E$. We now have two witnesses for D, and we need to pick the one that works. So, in this case, we let

$$S' \equiv \ldots, e_i^3 \leftrightarrow [(D'(\vec{e}^1) \wedge e_i^1) \vee (\neg D'(\vec{e}^1) \wedge e_i^2)], \ldots, \Lambda, \Gamma' \to \Delta', D'(\vec{e}^3), \Omega',$$

where $\vec{e}^3$ are new extension variables. The ordering $\prec$ is defined as $\prec_1$ with $\vec{e}^3$ added as the maximum elements so far.

Now we look at each part of the induction hypothesis to be sure it still holds. For properties 1 to 3, notice that the initial part of S' is part of the extension cedent and it defines the new variables $\vec{e}^3$. With this observation, it is easy to see that properties 1 to 3 still hold. Now to look at property 4. Since $\vec{e}^3$ are the largest elements in the ordering $\prec$, anything they depend on must be incomparable or smaller. So property 4 holds for $\vec{e}^3$. For other extension variables, it holds directly from the induction hypothesis. Property 5 follows directly from the induction hypothesis since Ω did not change. Property 6 follows directly from the induction hypothesis.

For the second possibility, assume that D is not a Σ_1^q formula. Then by induction with S_1 we get

$$S_1' \equiv \Lambda, \Gamma' \to \Delta', D_1^*, D_2^*, \Omega'$$

where D_1^* and D_2^* are $(Q_{S_1}, E, \prec_1)$-instances of an $\vee$-expansion of D. Then we let

$$S' \equiv \Lambda, \Gamma' \to \Delta', D_1^* \vee D_2^*, \Omega'$$

which can be obtained from S_1' using $\vee$-right. Notice that $D_1^* \vee D_2^*$ is also a $(Q_S, E, \prec)$-instance of an $\vee$-expansion of D, and, since the ordering is not changed, the induction hypothesis still holds.

Inductive Case 5: S is inferred using contraction-left

Suppose

$$S \equiv D, \Gamma \to \Delta, \Omega$$

and it is derived from

$$S_1 \equiv D, D, \Gamma, \to \Delta, \Omega.$$

The formula D must be a Σ_1^q formula and an ancestor of a cut formula. Let it be of the form $\exists x F(x)$, where $F(x)$ is a Σ_0^q formula. For now, we assume that D has a single existential quantifier, but the construction easily generalizes. By induction with S_1, there exists a sequent S_1' of the form

$$\Lambda(q_1, q_2), F(q_1), F(q_2), \Gamma' \to \Delta', \Omega'$$

with an ordering $\prec$. Without loss of generality, assume that $q_1 \prec q_2$. Then, we let

$$S' \equiv \Lambda(q_1, q_1), F(q_1), \Gamma' \to \Delta', \Omega'$$

The ordering will remain the same.

To prove S', we take the proof of S_1', replace every instance of q_2 by q_1, and then contract the two copies of $F(q_1)$. The substitution can be done because the proof is treelike.

We now look at each part of the induction hypothesis to be sure it still holds. It is easy to see that properties 1 to 3 hold.

Property 4 follows from the induction hypothesis. Note that, if a variable depended on q_2, it now depends on q_1, but that is fine since q_1 is smaller.

Property 5 holds since Ω did not change. $\square$

From this, we are able to prove the witnessing theorem (Theorem 3.2.3). We will restate it here.

Theorem 3.2.10 (KPT Witnessing for G_1^*)**.** *There exists a $\mathcal{L}_{FP}$ (polynomial-time) function F such that VPV proves the following. Let π be a G_1^* proof of a prenex Σ_2^q formula $A(\vec{p}) \equiv \exists\vec{x}\forall\vec{y}B(\vec{x}, \vec{y}, \vec{p})$, where $B(\vec{x}, \vec{y}, \vec{p})$ is a Σ_0^q formula with all free variables shown. Then, given π, F outputs a G_0^* proof of a sequent $\Lambda \to \Theta$ where*

1. *Θ is a series of n formulas of the form $B(\vec{e}^{\,i}, \vec{q}^{\,i}, \vec{p})$, where $\vec{e}^{\,i} \in E$ and $0 \le i < n$*
2. *Λ is an extension cedent defining a new set of variables E in terms of $\vec{q}^{\,0}, \ldots, \vec{q}^{\,n-1}$ and $\vec{p}$,*

3. $\vec{e^i}$ does not depend on $\vec{q^j}$, for $j \geq i$, and

4. $\vec{q^i}$ and $\vec{q^j}$ are disjoint when $i \neq j$.

Proof. By the Herbrand theorem above, there is a proof of $\Lambda \longrightarrow A^*$, where A^* is a $(Q_\pi, E, \prec)$-instance of an $\vee$-expansion of A. We need to show how to get Θ from A^*.

The first observation we make is that A^* is of the from

$$B(\vec{e^1}, \vec{q^1}, \vec{p}) \vee \ldots \vee B(\vec{e^n}, \vec{q^n}, \vec{p}).$$

This is true because the rule (α) gives multiple copies of B, which all remain in tact since B is a Σ_0^q formula, combined using $\vee$. This means that A^* is essentially the Θ we want.

Without loss of generality, we can assume that, if $i < j$, then the smallest variable in $\vec{q^i}$ is smaller than the smallest variable in $\vec{q^j}$. This implies that $\vec{e^i}$ does not depend on $\vec{q^j}$ for $j \geq i$. This is because every variable in $\vec{e^i}$ is smaller than every variable in $\vec{q^i}$ since we have a $(Q_\pi, E, \prec)$-instance.

The final observation is that if $\vec{q^i}$ and $\vec{q^j}$ contain a common variable then $\vec{e^i}$ and $\vec{e^j}$ must be the same. Otherwise, if $\vec{e^i}$ and $\vec{e^j}$ are different, then an application of (α) must have occurred that would make part of $\vec{e^i}$ and $\vec{e^j}$ correspond to different existential variables. Since the universal variables are in the scope of these existential variables, $\vec{q^i}$ and $\vec{q^j}$ would correspond to different quantifiers making them disjoint.

If we have that $\vec{q^i}$ and $\vec{q^j}$ are not disjoint, we are able to replace $\vec{q^j}$ by $\vec{q^i}$ everywhere in the proof. Then we can contract the two copies of $B(\vec{e^i}, \vec{q^i})$. $\square$

3.2.2 Witnessing for G_i^*

In the statement of the original KPT witnessing theorem for VPV, polynomial-time functions are used to find the possible witnesses; however, for TV^i, the KPT witnessing theorem uses functions in $FP^{\Sigma_i^p}$. Corresponding to that we will generalize the definition of an extension cedent to allow oracles from the other levels of the polynomial hierarchy.

Definition 3.2.11 (i-extension cedent)**.** An *i-extension cedent* is a series of formulas of the form

$$e_1 \leftrightarrow E_1, e_2 \leftrightarrow E_2, \ldots, e_n \leftrightarrow E_n$$

such that E_m is a $\Sigma^q_i \cup \Pi^q_i$ formula that does not mention the variables $e_m, e_{m+1}, \ldots, e_n$.

Note that an extension cedent is the same as a 0-extension cedent.

As with G^*_i, we are able to prove the analogue of the KPT witnessing theorem for G^*_i.

Theorem 3.2.12 (KPT Witnessing for G^*_i)**.** *There exists a $\mathcal{L}_{FP}$ (polynomial-time) function F such that VPV proves the following. Let π be a G^*_i proof of a prenex Σ^q_{i+1} formula $A(\vec{p}) \equiv \exists\vec{x}\forall\vec{y}B(\vec{x},\vec{y},\vec{p})$, where $B(\vec{x},\vec{y},\vec{p})$ is a Σ^q_{i-1} formula with all free variables shown. Then, given π, F outputs a G^*_{i-1} proof of a sequent $\Lambda \rightarrow \Theta$ where*

1. *Θ is a series of formulas of the form $B(\vec{e^i},\vec{q^i},\vec{p})$, where $\vec{e^i} \in E$*
2. *Λ is an $(i-1)$-extension cedent defining a new set of variables E in terms of $\vec{q^1},\ldots,\vec{q^n}$ and $\vec{p}$,*
3. *$\vec{e^i}$ does not depend on $\vec{q^j}$, for $j \geq i$, and*
4. *$\vec{q^i}$ and $\vec{q^j}$ are disjoint.*

Again, we will prove a more general theorem. To be able to the handle formulas that are not prenex, we use the idea of an i-expansion.

Definition 3.2.13 (i-expansion)**.** The same as an $\vee$-expansion except that B must be a non-($\Sigma^q_i \cup \Pi^q_i$) formula instead of a non-Σ^q_0 formula.

Definition 3.2.14 ($(i,Q,E,\prec)$-instance)**.** An *$(i,Q,E,\prec)$-instance* of a formula A is the same as a $(Q,E,\prec)$-instance of A except that $\Sigma^q_i \cup \Pi^q_i$ subformulas of A are not changed. That is, the quantifiers that appear in the i innermost blocks of quantifiers are not replaced.

Note that a $(i, Q, E, \prec)$-instance of a formula will always be a $\Sigma_0^q(\Sigma_i^q)$ formula.

Example 3.2.15. Let A be the formula $\exists\vec{x}\forall\vec{y}\exists\vec{z}B(\vec{x},\vec{y},\vec{z})$, where B is a Σ_0^q formula. Then a $(1, Q, E, \prec)$-instance of A would be $\exists\vec{z}B(\vec{e},\vec{q},\vec{z})$ and a $(2, Q, E, \prec)$-instance would be $\forall\vec{y}\exists\vec{z}B(\vec{e},\vec{y},\vec{z})$.

Theorem 3.2.16 (Herbrand Theorem for G_i^*)**.** *For each $i > 0$, there is a $\mathcal{L}_{FP}$ function F such that VPV proves the following. Let π be a G_i^* proof of A. Then, given π, F outputs a G_{i-1}^* proof of a sequent $\Lambda \longrightarrow A^*$ and a partial ordering $\prec$ of the variables $Q_\pi \cup E$, where E is a set of variables that do not appear in π, with the following properties:*

- *Λ is a $(i-1)$-extension cedent defining the variables in E in terms of Q_π and the free variables of A;*
- *for $e \in E$, if e depends on a variable $p \in Q_\pi \cup E$, then $p \prec e$; and*
- *A^* is an $(i-1, Q_\pi, E, \prec)$-instance of an $(i-1)$-expansion of A.*

Before we prove this theorem, we should note that it does not seem like we can improve the complexity of the extension cedent. For, if the $(i-1)$-extension cedent could be replaced by a $(i-2)$-extension cedent, this could be used to show that G_{i-1}^* p-simulates G_i^* for prenex formulas. The proof would be similar to the proof of Theorem 4.1.2.

Proof of Theorem 3.2.16. The proof is almost the same as in the G_1^* case. The quantifier complexity of the cut formulas is reduced by witnessing the outermost block of existential quantifiers with extension variables. The only difference is that we can no longer skip all of the quantifier introduction rules. Some will have to be added. For example, if we have a cut formula $\exists\vec{x}\forall\vec{y}C(\vec{x},\vec{y})$, then we will replace $\vec{x}$ by extension variables, but we still add $\forall\vec{y}$ to the formula.

This construction can be described more formally. As before, each sequent S in π can be divided into three parts: Γ which contains all of the formulas on the left-hand side; Δ, which contains the formulas on the right-hand side that are ancestors of cut formulas; and Ω, which contains the ancestors of the final formula on the right-hand side. Note that by the subformula property, we know that Γ and Δ contain only Σ_i^q formulas. For each sequent $S \equiv \Gamma \rightarrow \Delta, \Omega$ in π, we construct a G_{i-1}^* proof of a sequent

$$S' \equiv \Lambda, \Gamma' \rightarrow \Delta', \Omega',$$

and a partial ordering $\prec$ on $Q_S \cup E$ where

1. Γ' is obtained from Γ by replacing each non-Σ_{i-1}^q formula $\exists\vec{z}D(\vec{z})$ by $D(\vec{q})$, where D is Π_{i-1}^q and $\vec{q} \in Q_S$. (We use different $\vec{q}$ for different formulas.)

2. Δ' is obtained from Δ by replacing each non-Σ_{i-1}^q formula $\exists\vec{z}D(\vec{z})$ by $D(\vec{e})$, where D is Π_{i-1}^q and $\vec{e} \in E$. (We use different $\vec{e}$ for different formulas.)

3. Λ is an $(i-1)$-extension cedent defining E;

4. for $e \in E$, if e depends on a variable $p \in Q_S \cup E$, then $p \prec e$;

5. Ω' is obtained from Ω by replacing each formula B by an $(i-1, Q_S, E, \prec)$-instance of an $(i-1)$-expansion of B; and

6. each $q \in Q_S$ appears in at most 1 formula in Γ', Δ', and Ω'.

The construction is the same as in Theorem 3.2.9 except for the need to add a few new cases. If $\exists$-right is applied with a Σ_{i-1}^q principal formula or $\forall$-left is applied with a Π_{i-1}^q principal formula, the same inference can be used in the G_{i-1}^* proof we are constructing. We must also consider when S is derived using $\exists$-left with a Σ_{i-1}^q principal formula and when S is derived using $\forall$-right with a Π_{i-1}^q principal formula. Both cases are handled in the same way, so we only describe the latter.

Suppose

$$S \equiv \Gamma \to \Delta, \forall q D(q), \Omega$$

and it is derived from

$$S_1 \equiv \Gamma \to \Delta, D(q), \Omega$$

where $D(q)$ is Π^q_{i-1}. By induction, we have a G^*_{i-1} proof of S'_1, where

$$S'_1 \equiv \Lambda(q), \Gamma' \to \Delta', \Omega', D(q).$$

We know q does not appear in Γ' or Δ' since it was used as an eigenvariable, but it is still possible that the extension variables depend on it, in which case it would appear in Λ.

The first step is to replace q by a new extension variable e. This gives

$$\Lambda(e), \Gamma' \to \Delta', \Omega', D(e). \tag{3.2.1}$$

We then derive

$$e \leftrightarrow D(\bot), D(e) \to \forall q D(q). \tag{3.2.2}$$

See Lemma 3.1.3 in the previous section for this construction. We finish by deriving

$$e \leftrightarrow D(\bot), \Lambda(e), \Gamma' \to \Delta', \Omega', \forall q D(q)$$

by cut with sequents (3.2.1) and (3.2.2) and cut formula $D(e)$.

The ordering is changed by adding e as the smallest element. Note that, since $D(q)$ is in S_1, $D(\bot)$ does not contain any extension variables or eigenvariables in Q_S. With this fact in mind, we can see all of the conditions in the induction hypothesis follow. $\square$

Chapter 4

Polynomial Simulations

4.1 GPV_i^* And G_i^*

We now move on to applications of the Herbrand Theorem for G_1^*. The first application deals with a seemingly weaker proof system.

Definition 4.1.1. For $i \geq 0$, the proof system GPV_{i+1}^* is G_{i+1}^* in which cut formulas are restricted to Σ_i^q formulas or formulas of the form $\exists x[x \leftrightarrow A]$, where A is a Σ_i^q formula that does not mention x. The proof system GPV^* will refer to GPV_1^*.

At first glance, it seems like GPV^* would be a weaker proof system than G_1^* because the cut formulas are less expressive. The cut formulas in GPV^* can be trivially witnessed, but the cut formulas in G_1^* are NP-hard. Nevertheless, it can be shown that GPV^* and G_1^* are p-equivalent for prenex formulas. One direction is easy since every GPV^* proof is a G_1^* proof, so all that is left is to prove the other direction.

Theorem 4.1.2. *VPV proves that GPV_i^* p-simulates G_i^* for prenex formulas.*

Proof. We will prove the theorem for the case when $i = 1$. The general case is essentially the same. Let π be a G_1^* proof of a formula A of the form

$$\forall \vec{y}^0 \exists \vec{x}^1 \forall \vec{y}^1 \ldots \exists \vec{x}^n \forall \vec{y}^n B(\vec{y}^0, \vec{x}^1, \vec{y}^1, \ldots, \vec{x}^n, \vec{y}^n),$$

where B is a Σ_0^q formula. By the Theorem 3.2.9, VPV proves that there exists a G_0^* proof π' of a sequent $\Lambda \to A^*$ and a total ordering $\prec$ of the variables $Q_\pi \cup E$ meeting the conditions of the theorem. Since A is in prenex form, we know that A^* is of the form $\bigvee_{i=0}^{m} B(\vec{q}^{i,0}, \vec{e}^{i,1}, \vec{q}^{i,2}, \ldots, \vec{e}^{i,n}, \vec{q}^{i,n})$. From this we are able to get a proof of $\Lambda \to \Theta$ where

$$\Theta \equiv B(\vec{q}^{0,0}, \vec{e}^{0,1}, \vec{q}^{0,2}, \ldots, \vec{e}^{0,n}, \vec{q}^{0,n}), \ldots, B(\vec{q}^{m,0}, \vec{e}^{m,1}, \vec{q}^{m,2}, \ldots, \vec{e}^{m,n}, \vec{q}^{m,n})$$

by deriving $A^* \to \Theta$ and cutting A^*.

We describe an algorithm that takes as input π' and $\prec$. The algorithm extends π' into a GPV^* proof of A. At any stage, π' will be a proof of a sequent $\Lambda' \to \Theta'$, where Λ' is a subsequence of Λ and Θ' is a sub-series of Θ with some quantifiers added. The algorithm has four steps:

```
Step 1: Add as many existential quantifiers to the formulas
        in $\Theta'$ as possible using $\exists$-right rules such that the formula is still a subformula of $A$.
Step 2: Use contraction to combine as many formulas in
        $\Theta'$ as possible.
Step 3: Find the largest variable that is mentioned in
        $\Lambda'$ or $\Theta'$.
        Step 3a:    If it is an extension variable $e$,
                    apply $\exists$-left to the formula $e \leftrightarrow E$
                    with $e$ as the eigenvariable. Then
                    cut the formula $\exists e[e \leftrightarrow E]$ after
                    deriving $\to \exists e[e \leftrightarrow E]$.
        Step 3b:    If it was an eigenvariable $q$ in $\pi$,
                    then apply $\forall$-right with $q$ as the
                    eigenvariable.
Step 4: Repeat steps 1 to 3 until there is no change.
```

At first, it may not be obvious that this algorithm works. For example, it is not obvious that the eigenvariable restriction for $\exists$-left or $\forall$-right rules in Step 3 is met. To show that the eigenvariable restriction is met, we make two observations. First, if p is the largest variable in Λ' and Θ', then no extension variable can depend on p. Otherwise, that variable would be larger than p. Second, if we are in Step 3 and p is the largest variable in Λ' and Θ', then p cannot be mentioned in Θ' unless it is in Q_π; otherwise p would be an extension variable and have been used as the target formula in an $\exists$-right rule in Step 1. If this is not the case, an eigenvariable that appears to the right of p is still present, and this variable must be larger than p. For the same reason, we know that there cannot be two formulas in Θ with p replacing a universal variable that have not been contracted yet. This means the eigenvariable restriction is met in Step 3.

When the algorithm is done, we will have a proof of the formula we want. Notice that Λ' would be empty because every extension variable has been removed. Also, Θ' would be the single formula A since every formula in Θ would have every quantifier added by now, and would have been contracted to a single formula. We know the algorithm eventually stops because we continually reduce the number of variables in π'. □

4.2 G_i And G_i^*

As has already been mentioned, for $i > 0$, G_i is commonly connected with the theory TV^i and G_{i+1}^* is commonly connected with V^{i+1}. Since the two theories have the same Σ_{i+1}^B theorems, it was natural that the two proof systems are p-equivalent when proving Σ_{i+1}^q formulas. However, we want to extend this to more general formulas. In [30], it was shown that one direction is probably not possible. Namely that, under an appropriate complexity assumption, there is a family of Σ_{i+2}^q formulas for which G_{i+1}^* does not p-simulate G_i. Here we prove that G_i p-simulates G_{i+1}^* for all formulas.

The proof is based on the proof of Krajícek that depth d, DAG-like PK can p-simulate depth $d+1$, treelike PK (Proposition 1.4 in [19]). The observation of the similarity between the two theorems is due to Toni Pitassi.

Definition 4.2.1 (The i-Substitution Rule)**.** The i-substitution rule is

$$\frac{A_1(p), \ldots, A_m(p), \Gamma \rightarrow \Delta, B_1(p), \ldots, B_n(p)}{A_1(C), \ldots, A_m(C), \Gamma \rightarrow \Delta, B_1(C), \ldots, B_n(C)}$$

where C is a quantifier-free formula, $A_1, \ldots, A_m, B_1, \ldots, B_n$ are $\Sigma_i^q \cup \Pi_i^q$ formulas, and p does not appear in the bottom sequent.

Lemma 4.2.2. *For $i > 0$, G_i^* p-simulates the i-substitution rule.*

Proof. The proof is the same as the proof of Lemma 2.1 in [23]. We will describe how to do the simulation for the case where there is one A and B. The general case is done the same way.

Suppose we have a derivation of

$$A(p), \Gamma \rightarrow \Delta, B(p). \tag{4.2.1}$$

We want to derive

$$A(C), \Gamma \rightarrow \Delta, B(C).$$

First we derive

$$p \leftrightarrow C, A(C) \rightarrow A(p),$$

and cut this with (4.2.1), where $A(p)$ is the cut formula. This gives

$$p \leftrightarrow C, A(C), \Gamma \rightarrow \Delta, B(p). \tag{4.2.2}$$

Then we derive

$$p \leftrightarrow C, B(p) \rightarrow B(C),$$

and cut this with (4.2.2), where $B(p)$ is the cut formula. This gives

$$p \leftrightarrow C, A(C), \Gamma \rightarrow \Delta, B(C). \tag{4.2.3}$$

We then apply $\exists$-left to this sequent with p as the eigenvariable, and then cut $\exists p[p \leftrightarrow C]$ after deriving $\rightarrow \exists p[p \leftrightarrow C]$. □

Theorem 4.2.3. *For $i > 0$, G_i p-simulates G^*_{i+1}.*

Proof. At a high level, this proof is done by carefully applying one step of cut-elimination to each cut formula. The increase in the size of the proof in the cut-elimination theorem comes from repeating part of the proof multiple times. We avoid this increase by creating a DAG-like proof.

Let π be a G^*_{i+1} proof. The reason π is not a G_i proof is that it would contain cut formulas that are not Σ^q_i or Π^q_i. We can assume these formulas are Σ^q_{i+1} and are of the form

$$\exists x_1 \ldots \exists x_n C(x_1, \ldots, x_n),$$

where C is Π^q_i. We can assume this because, in [29], Morioka proved that any G^*_{i+1} proof can be transformed into a G^*_{i+1} proof where the cut formulas are prenex. We need to turn these cut formulas into Π^q_i cut formulas. To do this, we change all of the non-$(\Sigma^q_i \cup \Pi^q_i)$ formulas that are ancestors of these cut formulas. These formulas are of the form

$$\exists x_l \ldots \exists x_n C(D_1, \ldots, D_{l-1}, x_l, \ldots, x_n), \tag{4.2.4}$$

where D_j is a Σ^q_0 formula for $j < l$, and $C(\vec{x})$ is a Π^q_i formula. Note that, if this formula is on the left side of a sequent, then the formula D_i is a single variable, and that variable eventually get used as an eigenvariable in an $\exists$-left rule. From now on, we will assume all formulas of the form (4.2.4) are ancestors of cut formulas. Those that are not are simply ignored.

The construction will be done inductively. We start with the first sequent in π and work our way down the proof. For each sequent $S \equiv \Gamma \to \Delta$ in π, we give a G_i proof π' of a sequent $S' \equiv \Gamma' \to \Delta'$ where

1. Γ' is obtained from Γ by replacing every formula of the form (4.2.4) by

$$C(D_1, \ldots, D_{l-1}, x_l^C, \ldots, x_n^C),$$

2. Δ' is obtained from Δ by removing every formula of the form (4.2.4),

3. the sequent

$$C(D_1, \ldots, D_{l-1}, x_l^C, \ldots, x_n^C) \to$$

   can be used as an axiom if and only if Δ contains a formula of the form (4.2.4).

For example, if S is the sequent

$$\exists x_2, x_3 C_1(q_1, x_2, x_3), \Gamma \to \Delta, \exists x_3, x_4 C_2(D_1, D_2, x_3, x_4),$$

S' would be

$$C_1(q_1, x_2^{C_1}, x_3^{C_1}), \Gamma \to \Delta,$$

and when we prove S', we are allowed to use

$$C_2(D_1, D_2, x_3^{C_2}, x_4^{C_2}) \to$$

as an axiom. In essence, we are saying, if we can derive

$$C_2(D_1, D_2, x_3^{C_2}, x_4^{C_2}) \to,$$

we can prove S'. Note that, when we get to the final sequent, no formula is an ancestor of a cut formula. Therefore, if S is the final formula in π, $S' = S$ and the only initial sequents are of the form $x \to x$. So this will give us a proof of the theorem.

The construction of π' is given inductively. There is a separate case for each rule of inference. Most cases are simple and are left to the reader. The only cases we will give are cut, $\exists$-left, and $\exists$-right.

Cut: Suppose $S \equiv \Gamma \to \Delta$ is derived from S_1 and S_2 using cut. Let the cut formula be $\exists\vec{x}C(\vec{x})$. By induction with S_1, we have a G_i proof π'_1 of

$$S'_1 \equiv C(\vec{x}^C), \Gamma' \to \Delta'.$$

By induction with S_2, we have a G_i proof π'_2 of $\Gamma' \to \Delta'$ using the axiom $C(\vec{x}^C) \to$. Notice that π'_2 is a proof of the sequent we want, but it uses an axiom we are no longer able to use. However, π'_1 gives us a derivation of this axiom, with a few extra formulas.

The first step in the construction of π' is to add Γ' to the left and Δ' to the right of every sequent in π'_2. This makes the axiom we want to remove $C_i(\vec{x}^i), \Gamma' \to \Delta'$, which is the final sequent π'_1. So, π' is π'_1 followed by the new π'_2. Note that the axiom would have been used once for every time $\exists x_n$ was introduced in the original proof. Each of these formulas would later be contracted into the single cut formula. However, since we are constructing a DAG-like proof, we do not need to repeat π'_1 multiple times. This gives a proof of $\Gamma', \Gamma' \to \Delta', \Delta'$, from which we can derive $\Gamma' \to \Delta'$ using contraction.

$\exists$-left: Suppose S is

$$\exists x_j \ldots \exists x_n C(q_1, \ldots, q_{j-1}, x_j, x_{j+1}, \ldots, x_n), \Gamma \to \Delta,$$

and it was derived from S_1

$$\exists x_{j+1} \ldots \exists x_n C(q_1, \ldots, q_{j-1}, q_j, x_{j+1}, \ldots, x_n), \Gamma \to \Delta.$$

By induction with S_1, we get a G_i proof of

$$C(q_1, \dots, q_{j-1}, q_j, x^C_{j+1}, \dots, x^C_n), \Gamma' \to \Delta'.$$

Since q_j was used as an eigenvariable, it only appears in that one formula. Therefore we can replace q_j by x^C_j using the i-substitution rule. This gives us π'.

$\exists$-right: Suppose S is

$$\Gamma \to \Delta, \exists x_j \dots \exists x_n C(D_1, \dots, D_{j-1}, x_j, x_{j+1}, \dots, x_n),$$

and it was derived from S_1

$$\Gamma \to \Delta, \exists x_{j+1} \dots \exists x_n C(D_1, \dots, D_{j-1}, D_j, x_{j+1}, \dots, x_n).$$

First assume $j < n$. That is we had at least one quantifier already. By induction with S_1, we get a G_i proof of $\Gamma' \to \Delta'$ using the axiom

$$C(\dots, D_j, \dots) \to . \tag{4.2.5}$$

We cannot use this axiom anymore. Instead, we use the axiom

$$C(\dots, x^C_j, \dots) \to$$

and derive (4.2.5) using the i-substitution rule.

If $j = n$, the construction is a little different. By induction with S_1, we get a G_i proof of

$$\Gamma' \to \Delta', C(\dots, D_{n-1}, D_n). \tag{4.2.6}$$

To construct π', we take the axiom we can now use,

$$C(\ldots, D_{n-1}, x_n^C) \rightarrow,$$

and derive

$$C(\ldots, D_{n-1}, D_n) \rightarrow$$

using the i-substitution rule. Then we cut with (4.2.6).

Contraction: Suppose that S is

$$\Gamma \rightarrow \Delta, A,$$

and that it was derived from

$$\Gamma \rightarrow \Delta, A, A.$$

Then by induction with the upper sequent, there is a G_i proof of

$$\Gamma' \rightarrow \Delta', A', A'.$$

Here A' is the formula that is obtained from A, and, since this transformation is deterministic, both A's will be transformed into the same A'. When we can obtain S' by applying contraction to this sequent.

Upper bound on the size: We will find the upper bound on the size of the proof in two steps. First we will count the number of sequents in the new proof. Then we will get an upper bound on the size of each sequent. To keep the analysis simple, we will consider the i-substitution as a single step. We have already shown that G_i p-simulates G_i with the i-substitution rule.

To count the number of sequents, we will count the number of sequents we add in each step of the induction. In all of the rules except for cut and $\exists$-right, there is one new sequent. For $\exists$-right, there are at most two new sequents. For cut, we introduce sequents when doing the contractions at the end of that step. This would correspond to at most $|\Gamma| + |\Delta|$ new sequents. Totaling this we get that the number of formulas in π is an upper bound on the number of sequents in the new proof.

To find the maximum size of each sequent, the key observation is that each formula in a sequent corresponds to a distinct formula in the original proof (not necessarily in the corresponding sequent). So if a formula appears twice in a squent that is because that formula appears at least twice in the original proof. So again, the upper bound on the number of formulas in a sequent is the number of formulas that appear in the original proof. $\square$

4.3 Collapse of Bounded Arithmetic and Quantified Proof Systems

In this section, we explore some consequences of the collapse of V^∞ for the fragments of G. The results come from looking at the relations between bounded arithmetic and quantified propositional proof systems.

Some results of this type have already been obtained. One example is the following.

Theorem 4.3.1 (Infered From Theorem 9.3.17 [22]). *If TV^{i+1} is Σ_i^B conservative over TV^i, then G_i p-simulates G_{i+1} with respect to Σ_i^q formulas.*

This result is a corollary to the translation theorem and the fact that TV^{i+1} proves Σ_i^q-RFN(G_{i+1}) (in fact the Σ_{i+1}^q-RFN(G_{i+1})). The construction relies on G_i cutting the translation of Σ_i^q-RFN(G_{i+1}). Because of this, the proof cannot be generalized to show that G_i p-simulates G_{i+1} for more complex formulas.

The result we prove finds a way around this. By increasing the complexity of the cut formulas, we can show that a simulation holds for all quantified formulas. Our result is the following:

Theorem 4.3.2. *For $i > 0$, if $TV^i = V^{i+1}$, then, for all $j \geq 0$, G^*_{i+3} p-simulates G^*_j with respect to all quantified propositional formulas.*

The significance of this result should be understood as a way of expanding the search for a theorem that separates TV^i from V^{i+1}. Currently, if we want to separate the theories, we must find a theorem that can be stated as a formula in the language of bounded arithmetic. This means it can be stated as a relation in the PH. If we translate this into a family of quantified propositional formulas, then we are looking for a family of polynomial-size Σ^q_j formulas from some fixed j. However, Theorem 4.3.2 says any family of quantified propositional formulas that shows that G^*_i does not p-simulate G^*_{i+1} can be used to separate TV^i and V^{i+1}. The formulas need not be polynomial-size or have a fixed number of quantifier alternations. This opens the door to a family of tautologies that express a $PSPACE$ property. For example, we can consider formulas that talk about winning strategies in the games that are $PSPACE$-complete.

The proof of the theorem uses ideas in the proof of the following.

Theorem 4.3.3 (Corollary 7.4 [23]). *For $i > 0$, if TV^i proves that $NP = coNP$, then G_i is a polynomially-bounded proof system relative to Σ^q_i formulas.*

The idea is that, if $NP = coNP$, then every quantified boolean formula is equivalent to a Σ^q_1 sentence, which has a short proof if it evaluates to true.

In our case, if $TV^i = V^{i+1}$, then TV^i proves that $\Sigma^p_{i+3} = \Pi^p_{i+3}$ [6]. This means that TV^i proves that every Π^B_{i+3} formula is equivalent to a Σ^B_{i+3} formula. In particular, TV^i would prove that the truth definition for Π^q_{i+3} formulas is equivalent to a Σ^B_3 formula. Using the translation theorem, with appropriate extra work, we then get polynomial-size G^*_{i+3} proofs that every Π^q_{i+3} formula is equivalent to a Σ^q_{i+3} formula, and vice versa.

Now given a G^*_j proof, for $j > i+3$, we can reduce the quantifier complexity of the cut formulas by replacing every maximal Σ^q_{i+3} (Π^q_{i+3}) formula by its equivalent Π^q_{i+3} (Σ^q_{i+3}) formula. This can be done by cutting Σ^q_{i+3} or Π^q_{i+3} formulas.

Recall the truth definition given in Section 2.3.2.

$$(A \models_i F) \equiv \text{“}F \text{ is a } \Sigma_i^q \text{ formula that is satisfied by the assignment } A\text{”}.$$

When F is a Σ_i^q formula, this is a Σ_i^B formula. We want to show that, if $TV^i = V^{i+1}$, then $\models_{i+3}$ can be defined by a Π_{i+3}^B formula. The main tool in proving this is the following.

Theorem 4.3.4 (Lemma 7 [6]). *If $TV^i = V^{i+1}$ and F codes a Σ_{i+3}^q (Π_{i+3}^q) formula, there is a Π_{i+3}^B (Σ_{i+3}^B) formula $\psi(A, F)$ such that TV^i proves*

$$(A \models_{i+3} F) \leftrightarrow \psi(A, F)$$

The idea is that, if $TV^i = V^{i+1}$, then PH collapses to $\Sigma_{i+3}^p = \Pi_{i+3}^p$ (Lemma 7 [6]). This collapse is provable in TV^i. The theorem above is simply a special case where we are using the Σ_{i+3}^p relation $(A \models_{i+3} F)$ and finding an equivalent Π_{i+3}^p definition. We can now use this result and the translation theorem to construct polynomial-size G_{i+3}^* proofs that every Σ_{i+3}^q is equivalent to a Π_{i+3}^q formula, and vice versa.

Lemma 4.3.5. *If $TV^i = V^{i+1}$, then V^1 proves that there is a polynomial $p(n)$ such that for every Σ_{i+3}^q (Π_{i+3}^q) formula $F(\vec{x})$ there is a Π_{i+3}^q (Σ_{i+3}^q) formula $F'(\vec{x})$ and a G_{i+3}^* proof of $F(\vec{x}) \leftrightarrow F'(\vec{x})$ of size less than $p(|F|)$.*

Proof. Assume $TV^i = V^{i+1}$, and fix F. Suppose F is a Σ_{i+3}^q formula. The proof is essentially the same when F is a Π_{i+3}^q formula. By Lemma 4.3.4, there is a TV^i proof of

$$(A \models_{i+3} F) \leftrightarrow \psi(A, F)$$

where ψ is a Π_{i+3}^B formula. Then by the translation theorem (Theorem 2.3.3), there is a polynomial-time constructable G_{i+1}^* proof of

$$B(\vec{x}) \leftrightarrow F'(\vec{x}) \tag{4.3.1}$$

where $B(\vec{x})$ is the translation of $(A \models_{i+3} F)$ as in Lemma 2.3.6, and $F'(x)$ is the translation of $\psi(A, F)$. Since we are dealing with a specific F, the only free variables in the formulas are those for the assignment A, which are replaced by formulas that code x as an assignment. Note that $F'(x)$ is a Π^q_{i+3} formula since ψ is a Π^B_{i+3} formula.

By Lemma 2.3.6, there is a polynomial-size G^*_{i+3} proof of

$$F(\vec{x}) \leftrightarrow B(\vec{x}) \tag{4.3.2}$$

Combining the proofs of (4.3.1) and (4.3.2), with appropriate manipulation, we get a proof of

$$F(\vec{x}) \leftrightarrow F'(\vec{x}).$$

See Section 3.1 for how these manipulations are done. □

We can now prove the main theorem of this section

Theorem 4.3.6 (Theorem 4.3.2)**.** *For $i > 0$, if $TV^i = V^{i+1}$, then, for all $j \geq 0$, G^*_{i+3} p-simulates G^*_j with respect to all quantified propositional formulas.*

Proof. Let $j \geq i + 3$. We will show that G^*_j p-simulates G^*_{j+1}. Then we get a proof of the theorem by composing these simulations.

Let π be a G^*_{j+1} proof. We can assume that all of the cut formulas are prenex Σ^q_{j+1} [29]. We will show how to reduce the quantifier complexity of a non-Σ^q_j cut formula by one. We then get a proof of the theorem by repeating this process for every non-Σ^q_j cut formula.

Let one of the cut formulas in π be $\exists\vec{x}\forall y C(\vec{x}, y)$, where C is a Π^q_j formula. By Lemma 4.3.5, there is a G^*_j proof $\pi_1(\vec{x})$ of

$$\forall y C(\vec{x}, y) \leftrightarrow C'(\vec{x}),$$

where C' is a Σ^q_j formula. Note that $TV^i = V^i$ implies $TV^j = V^j$, so the conditions of the lemma apply.

Now we start to change π. First replace $\exists\vec{x}\forall y C(\vec{x}, y)$ by $\exists\vec{x}C'(\vec{x})$. This turns the cut formula into a Σ_j^q formula. As well, replace every ancestor of this cut formula of the form

$$\exists x_l \ldots \exists x_m \forall y C(B_1, \ldots B_{l-1}, x_l, \ldots, x_m, y)$$

by

$$\exists x_l \ldots \exists x_m C'(B_1, \ldots B_{l-1}, x_l, \ldots, x_m).$$

If there is an ancestor that does not have the $\forall y$ quantifier added yet, that formula is not replaced. This leaves a valid proof except for inferences of the form

$$\frac{\Gamma \to \Delta, C(\vec{B}, q)}{\Gamma \to \Delta, C'(\vec{B})}$$

or

$$\frac{C(\vec{q}, B), \Gamma \to \Delta}{C'(\vec{q}), \Gamma \to \Delta}$$

These are the inferences where the universal quantifier $\forall y$ is added. This can be turned into a valid proof as follows:

$$\frac{\dfrac{\Gamma \to \Delta, C(\vec{B}, q)}{\Gamma \to \Delta, \forall y C(\vec{B}, y)} \qquad \begin{array}{c} \vdots\ \pi_1(\vec{B}) \\ \forall y C(\vec{B}, y) \to C'(\vec{B}) \end{array}}{\Gamma \to \Delta, C'(\vec{B})}$$

Note that we implicitly used some transformations to $\pi_1(\vec{B})$ that are described in Section 3.1. The other invalid inferences can be fixed in the same manner. $\square$

Chapter 5

Reflection Principles

5.1 Reflection Principles For the Fragments of G

We can also use the Herbrand Theorem to prove reflection principles. Proving reflection principles is the standard method of assessing the strength of a proof system relative to a theory. For example, the Σ_1^q reasoning of G_1^* is not stronger than the Σ_1^B reasoning of V^1 because V^1 proves Σ_1^q-RFN(G_1^*) [22]. Our goal is to find the weakest fragment of V^∞ that proves Σ_i^q-RFN(G_1^*). In [29], it was shown that TV^0 does not prove Σ_2^q-RFN(G_1^*) unless the polynomial-time hierarchy collapses. Using the same ideas, it is possible to show that TV^i does not prove Σ_{i+2}^q-RFN(G_1^*), for $i \geq 0$, unless the polynomial-time hierarchy collapses. This still leaves open whether or not V^i proves Σ_{i+1}^q-RFN(G_1^*) for $i \geq 1$. We prove that, in fact, it does.

We first prove the simplest case. Namely, that V^1 proves (prenex Σ_2^q)-RFN(G_1^*). The proof serves as a template for the general case, which we prove right after.

Theorem 5.1.1. *V^1 proves (prenex Σ_2^q)-RFN(G_1^*).*

Proof. Let π be a G_1^* proof of a prenex Σ_2^q formula A. So A is of the form

$$\exists\vec{x}\forall\vec{y}B(\vec{x},\vec{y},\vec{p}),$$

where B is a Σ_0^q formula. In this formula, $\vec{p}$ is all of the free variables in A, and should be understood as being implicitly universally quantified. We want to prove in V^1 that, given values for $\vec{p}$, there exist values for $\vec{x}$ that witness the formula.

By the KPT witnessing theorem for G_1^* (Theorem 3.2.3), V^1 proves that there is a G_0^* proof of a sequent

$$S \equiv \Lambda \to \Theta,$$

meeting the conditions of the theorem.

Let

$$\begin{aligned}
&\psi(m, \Lambda, \Theta, P) \equiv \\
&\exists E\ \exists Q\ \text{“E is a truth assignment to the extension variables”} \\
&\qquad \wedge \text{“Q is a truth assignment to the eigenvariables”} \\
&\qquad \wedge \forall i < m\ (P \cup E \cup Q) \models_0 \neg B(\vec{e^i}, \vec{q^i}, \vec{p}) \\
&\qquad \wedge (P \cup E \cup Q) \models_0 \Lambda
\end{aligned}$$

This formula says that there exists assignments E and Q that satisfy Λ and make the first m formulas in Θ false. It is easy to bound the size of E and Q. This means that ψ is equivalent to a Σ_1^B formula.

Using Σ_1^B-MAX, we find the maximum value m_0 for m that satisfies ψ given values for Λ, Θ, and P. Then $\vec{e^{m_0+1}}$ are the witnesses we are looking for, which we now prove.

First note that $\psi(0)$ is true. We can set Q to the assignment that sets everything to false, and compute E that satisfies Λ. Also note that $m_0 < n$ since it is not possible the falsify all of the formulas in Θ. This would violate the Σ_0^q-RFN(G_0^*), which is provable in V^1. This means that $\vec{e^{m_0+1}}$ exists.

Let E and Q be witnesses for $\psi(m_0)$. For the sake of contradiction assume $\vec{e^{m_0+1}}$ is not a witness for $\exists\vec{x}\forall y B(\vec{x}, \vec{y}, \vec{p})$. Change Q so that $\vec{q^{m_0+1}}$ are assigned values falsifying $B(\vec{e^{m_0+1}}, \vec{q^{m_0+1}}, \vec{p})$. We can then change E so that Λ is satisfied. Since $\vec{e^j}$, for $j \leq m_0+1$, does not depend on $\vec{q}^{m_0+1}$, their values stay the same. This means we now have E and Q making the first $m_0 + 1$ formula in Θ false, violating our choice of m_0. □

Theorem 5.1.2. *$V^i \vdash \Sigma^q_{i+1}$-RFN$(G^*_i)$.*

Proof. Suppose we have a G^*_i proof π of a Σ^q_{i+1} formula A. By Theorem 3.2.16 (Herbrand Theorem), we can find a G^*_{i-1} proof of an instance of an $\vee$-expansion of A, with an ordering $\prec$. The ordering will not be just any ordering that meets the conditions of the theorem, but will be the specific ordering constructed in the proof of that theorem. We choose this ordering because we want to be able to infer something about π from the order of the variable.

In particular, let q_1 and q_2 be two distinct eigenvariables in π. Start at the inferences where these variables are used as eigenvariables and move down the proof to the first place where they meet. We can ask which rule was used in that inference. There are three possibilities. The first possibility is a $\forall$-right (or $\exists$-left). In this case, the eigenvariable for this $\forall$-right inference is either q_1 or q_2, and one variable precedes the other in the ordering. This can be easily observed from the $\forall$-right case in the proof of Theorem 3.2.16.

The second possibility is the cut rule. In this case, one of the variables is less than the other as well. The final possibility is $\wedge$-right (or $\vee$-left). In this case, q_1 and q_2 are incomparable.

There is one final observation to make. Suppose we have a proof of $\forall y_1 A(y_1) \wedge \forall y_2 B(y_2)$, and the instance we get from the construction in the Herbrand Theorem is $A(q_1) \wedge B(q_2)$. Then we know that q_1 and q_2 are incomparable. This is because the meeting place for these two eigenvariables is the rule that introduced the $\wedge$ between A and B.

At this point, we are prepared to move on to the proof of this theorem. Let A^* be the $\vee$-expansion of A, and let A' be the instance of A^*. Then, by Lemma 5.1.3 below, all we need to do is prove A^* is valid in order to prove the reflection principle.

To explain how this is done, we start with a simplified example. Suppose A^* is of the form

$$\bigvee_{i=0}^{n} \exists \vec{x}_i \bigwedge_{j=0}^{m_i} \forall \vec{y}_{i,j} B_{i,j}(\vec{x}_i, \vec{y}_{i,j})$$

and A' is

$$\bigvee_{i=0}^{n} \bigwedge_{j=0}^{m_i} B_{i,j}(\vec{e}_i, \vec{q}_{i,j}).$$

The first thing to note is that $\vec{q}_{i,j}$ is incomparable with $\vec{q}_{i,j'}$ for $j \neq j'$. The second thing to note is that we can assume that if $i < i'$ then $\vec{q}_{i',j'} \not< \vec{q}_{i,j}$. If this were not true, the first observation would be violated. To show that A^* is valid, we give a student-teacher algorithm that witnesses the formula. The student starts by assigning false to every q-variable. Using Λ, the student finds an assignment for the extension variables, and presents this to the teacher as a possible witness to A^*. If it is, we are done. If not, the teacher replies with an assignment to the q-variables that prove it is false. In particular, $\bigwedge_{j=0}^{m_0} B_{0,j}(\vec{e}_0, \vec{q}_{0,j})$ is false. This means there exists a j where $B_{0,j}$ is false. The student the adjust his assignment to $\vec{q}_{0,j}$ to match the teachers response. At this point, the student has values for $\vec{e}_0$ and $\vec{q}_{0,j}$ that make $B_{0,j}$ false. From now on, we will be careful not make changes to the q-variable that will cause $\vec{e}_0$ to change value (we will explain why this is true later). So what the student has done is *fix $B_{0,j}$ to false.* As well, since $B_{0,j}$ will always be false from now on, $\bigwedge_{j=0}^{m_0} B_{0,j}(\vec{e}_0, \vec{q}_{0,j})$ will always be false as well. We can say this has also be fixed to false. Note that the value of $B_{0,j'}$, for $j' \neq j$, does not matter anymore. So we can now call it *irrelevant.* The student now uses this new assignment to the q-variables to get a new assignment to the extension variables. The teacher responds, and the student does the same thing except 1 is used in place of 0. This continues until either the student has a witness for A^* or an assignment satisfying Λ and falsifying A', but the latter is not possible. So A^* must be valid. Note that the prenex case given above is a special case of this where $m_i = 1$ for all i.

The general case is essentially the same except there is more than one $\bigvee$ in the formula. However, the general idea is the same. In each round, the student will make progress on one of the $\bigvee$. The difficult part is to formalize this algorithm in such a way that the proof of correctness can be carried out in the theory. The algorithm is formalized as follows. We define a Σ_i^B formula $\phi(l)$ that says the student-teacher game can last for l rounds. Then assuming that A^* is not valid, we show by induction on l that the game can last long enough for the student to have an assignment to the variables that satisfy Λ and falsify A'. However, this violates the Σ_{i-1}^q reflection principle for G_{i-1}^*, which is provable in V^i.

To define $\phi(l)$, we must describe how the student picks the block in each round. We start with a little notation. In the previous case, a block of quantifiers was $\vec{q^i}$. In this theorem, we put all of the universal quantifiers that are in the scope of the same existential quantifiers in one group. For example, if

$$A^* \equiv (\exists x_1(\forall y_1 B \vee \forall y_2 B)) \wedge \exists x_2(\forall y_3 C \wedge \exists x_3 \forall y_4 D)$$

then there are three groups of quantifiers. The variables y_1 and y_2 form one group since they are both in the scope of x_1 and no other variables. The variable y_3 forms the second group. It cannot be in the same group as y_4 because it is not in the scope of x_3. The final group is y_4. We order the groups of universal variables by a minimal variable according to $\prec$ in the group. The smallest variable would be one corresponding to the outermost quantifiers. It is possible that there is more than one minimal, but that does not matter. Pick any one.

If B^* is a subformula of A^*, we will use B' to be the corresponding subformula in A'. Let $B_n^*(\vec{x})$ be the minimal subformula of A^* that contains all of the universal quantifiers corresponding the nth block of universal quantifiers. The $\vec{x}$ are the existentially quantified variables where B_n^* is in the scope of that quantifier. We will use $\vec{e}_n$ to refer to the extension variables that replace $\vec{x}$. Then $B_n'(\vec{e}_n, \vec{q}_n)$ is the corresponding subformula in

A'.

In each round, the student starts with an assignment to the eigenvariables. In the first round, any assignment will do. The student uses this assignment to get an assignment to the extension variables that satisfies Λ. This latter assignment is given to the teacher as a possible witness to A^*. The teacher responds with a counter example. Under the assumption that A^* is false, the teacher can always respond.

Let $i_0, \ldots, i_n$ be the list of blocks that were chosen in the first n rounds. The idea is that the student has values for $\vec{e}_{i_m}$ and $\vec{q}_{i_m}$ that make B'_{i_m} false, and the student will not change $\vec{e}_{i_m}$. So, as before, it is possible to say that B'_{i_m} has been fixed to false. We can then extend this idea of being fixed to false to other subformulas of A'. If $B' \equiv D_1 \vee D_2$ and both D_1 and D_2 are fixed to false, then B' is fixed to false. If $B' \equiv D_1 \wedge D_2$ and D_1 is fixed to false, then B' is fixed to false and D_2 and all of its subformulas are now irrelevant.

In round $n+1$, the student picks i_{n+1} to be the smallest value such that

- $B'_{i_{n+1}}$ is neither fixed to false nor irrelevant,
- the counter-example provided by the teacher says $B'_{i_{n+1}}$ is false, and
- if $\vec{q}_j \prec \vec{e}_{i_{n+1}}$, then B'_j is either fixed to false or irrelevant.

The first condition guarantees that the student is making progress on one of the $\bigwedge$. The second condition is there because we want to fix $B'_{i_{n+1}}$ to false, so the student must choose a formula he knows to be false. The third condition guarantees that $\vec{e}_{i_{n+1}}$ will not be changed by a choice is an later round.

Then $\psi(n)$ can be defined as $\psi(n) \equiv \ \exists i_0, \ldots, i_n \exists E \ \exists Q \ \exists C$

1. $\forall m \leq n$ "$Q^{[m]}$ is a truth assignment to the eigenvariables."
2. $\wedge \forall m \leq n$ "$Q^{[m]}$ differs from $Q^{[m+1]}$ only in the assignment to the i_m block."

3. $\wedge \forall m \leq n$ "$E^{[m]}$ is a truth assignment to the extension variables."

4. $\wedge$ "$E^{[i]}$ and $Q^{[i]}$ satisfy Λ."

5. $\wedge \forall m \leq n$ "$C^{[m]}$ is a counter-example proving $E^{[m]}$ is not a witness for A^*"

6. $\wedge \forall m \leq n$ "i_m meets the conditions described above."

All that remains is to prove the induction step. Given $i_0, \ldots, i_n$, how do we know there is always an appropriate i_{n+1}?

First consider a tree where the root of the tree is labeled with A'. The leaves are labeled with B'_n. The internal nodes are labeled with subformulas of A'. The edge are done in the obvious way. We say a formula B' is involved in making A' false if every node in the path from A' to B' in this tree is labeled with a false formula.

To prove that an appropriate i_{n+1} always exists we construct a sequence $j_1, j'_1, j_2, j'_2, \ldots$ as follows. The value of j_1 is the minimum value such that B'_{j_1} meets the first two conditions that i_{n+1} must meet and is involved in making A' false. If no such value exists, then A' would would be fixed to false, but that is not possible.

Then, given j_m, j'_m is a value such that $\vec{q}_{j'_m} \prec \vec{e}_{j_m}$, $B'_{j'_m}$ is not fixed to false or irrelevant, and the teacher did not prove that $B'_{j'_m}$ is false. This value must exist otherwise j_m would be an appropriate value for i_{n+1}.

Given j'_m, the value j_{m+1} is obtained by finding the minimal superformula of $B'_{j'_m}$ that is involved in making A^* false. That is, go up the tree from $B'_{j'_m}$ until you find the last possible formula that is true, and go up one more. This formula must be of the form $D_1 \wedge D_2$ where $B'_{j'_m}$ is a subformula of say D_1. Then choose j_{m+1} to be the minimum value such that $B'_{j_{m+1}}$ is a subformula of D_2 that is involved in making A^* false, and is neither fixed to false or irrelevant. If no such value exists, D_2 would be fixed to false and $B'_{j'_m}$ would be irrelevant, so j_{m+1} exists.

Since this sequence continues forever, it must repeat at some point. Let $j_1, j'_1, j_2, j'_2, \ldots, j_{l+1}$ such a sequence up to the point where is begins to repeat (i.e. $j_{l+1} = j_1$). Further assume this sequence is of minimum length. To get a contradiction, we look at the original proof π of A. For each j_m (j'_m), there is a subproof of π that is the minimal proof that contains all of the $\forall$-right rules that introduced $\vec{q}_{j_m}$ ($\vec{q}_{j'_m}$). This gives us $2l$ subproofs. Since $\vec{e}_{j_m}$ depends on $\vec{q}_{j'_m}$, the corresponding subproofs must meet with a cut or one is a subproof of the other. See the comments at the beginning of this proof. The subproof that corresponds to j'_m and j_{m+1} must be joined by an $\wedge$-right inference. This would be the $\wedge$-right inference that introduced the $\wedge$ in $D_1 \wedge D_2$ used in the construction above.

If we used the same inference to join two different pairs, we would be able to shorten the sequence. For example, if the same $\wedge$-right rule is used to join the proof of the pairs (j'_m,j_{m+1}) and (j'_l,j_{l+1}). Start with j'_m. Find the $D_1 \wedge D_2$ as the algorithm describes. Then j'_m is in D_1 and j_{m+1} is in D_2. Since j'_l and j_{l+1} meet at the same connective, j'_l is in either D_1 or D_2 and j_{l+1} is in the other.

If j_{l+1} is in D_2, then $j_{m+1} = j_{l+1}$ since in both cases we chose the minimum values meeting some conditions. Since we have a repeated value, it is obvious how to reduce the size of the sequence.

If j'_l is in D_2, then j'_l meets j_{m+1} at some point in D_2. Since j_{m+1} is involved in making A^* false, this formula is false. However, this contradicts the assumption that D_2 is the maximal superformula of $B_{j'_l}$ that is true. This means the two pairs cannot meet at the same connective.

This gives $2l$ proofs that are joined in $2l$ different places, but that is not possible in a tree-like proof. □

Lemma 5.1.3. *V^i proves that, if A^* is an $\vee$-expansion of a Σ^q_{i+1} formula A, then*

$$\sigma \models_{i+1} A^* \leftrightarrow \sigma \models_{i+1} A.$$

Proof. Done by induction on the number of applications of (α) (Definition 3.2.4) used to obtain A^* from A. The tricky part is the setup the induction so that is can be carried out in V^i.

Let

$$A = A_1^*, A_2^*, \ldots, A_n^* = A^*$$

be a series of formulas such that A_{i+1}^* is obtained from A_i with one application of (α). Assume A is true. From the truth definition, there is a Π_i^B formula that says that X is a witness for A and E is the evaluation of A that proves it. Assuming A is true, such an X and E exists. Define a polynomial-time function that given X and E and i will produce a witness and evaluation for A_i^*. We prove this function is correct by induction on i. When $i = n$, this function gives us a witness and evaluation for A^*, which means A^* is true.

The other direction is done in a similar manner. $\square$

5.2 New Axiomatization of V^∞

In this section, we will strengthen a result from [23]. In that paper, Krajícek and Pudlák showed that V^∞ can be axiomatized by $V^1 + \{\Sigma_i^q\text{-RFN}(G_i) \mid i \in \mathbb{N}\}$. A similar proof can be used to prove that V^∞ can be axiomatized by $V^1 + \{\Sigma_i^q\text{-RFN}(G_i^*) \mid i \in \mathbb{N}\}$. In this section, we show that V^∞ can also be axiomatized by $V^1 + \{\Sigma_i^q\text{-RFN}(CFG^*) \mid i \in \mathbb{N}\}$, where CFG^* is the cut-free version of G^*. Note that CFG^* is a weaker proof system than any of the other fragments of G including G_0^*.

Just a bit of notation. If A is a formula with free variables $\vec{p}$, then $\exists A$, called the existential closure of A, is the formula $\exists \vec{p} A$.

Lemma 5.2.1. V^1 *proves*

$$\Sigma_{i+1}^q\text{-}RFN(CFG^*) \leftrightarrow \Sigma_{i+1}^q\text{-}RFN(G_i^*).$$

Proof. The if direction is easy since a CFG^* proof is also a G_i^* proof. The only if direction is not as easy. Assume Σ^q_{i+1}-RFN(CFG^*), and argue in V^1. Given a G^*_i proof π of a Σ^q_{i+1} formula A, we change it into a CFG^* proof of a formula

$$B \equiv A \vee \bigvee_{j=1}^{n} \exists(C_j \wedge \neg C_j),$$

where $C_1, \ldots, C_n$ are all of the cut formulas in π.

This is done by first replacing each cut by

$$\dfrac{\dfrac{\Gamma \to \Delta, C \qquad \dfrac{C, \Gamma \to \Delta}{\Gamma \to \Delta, \neg C}}{\Gamma \to \Delta, C \wedge \neg C}}{\Gamma \to \Delta, \exists(C \wedge \neg C)}$$

The sequents in the rest of the proof are changed to include $\exists(C_i \wedge \neg C_i)$. Note that none of the inferences are affected by adding this formula. The only problem could be the eigenvariable restriction in $\exists$-left and $\forall$-right inferences; however, since the new formula does not have any free variables, there is no problem. At the end of the proof, the A is combined with the new formulas using $\vee$-right inferences.

Since the cut formulas are Σ^q_i formulas, B is a Σ^q_{i+1} formula. By Σ^q_{i+1}-RFN(CFG^*), B is true, and, since $\exists(C_i \wedge \neg C_i)$ cannot be true, A must be true. This can be done in V^1 since it proves the Tarski conditions for the truth definition. □

Corollary 5.2.2. *$V^\infty = V^1 + \{\Sigma^q_i$-RFN$(CFG^*) \mid i \in \mathbb{N}\}$.*

Proof. Follows from the lemma above, Krajícek and Pudlák's axiomatization of V^∞, and the fact that Σ^q_{i+1}-RFN(G^*_i) implies Σ^q_i-RFN(G^*_i) □

On a similar note, we are able to axiomatize TV^i and V^i in terms of the reflection principles.

Theorem 5.2.3. *Let $i > 0$. Then*

- *$V^i = V^1 + \Sigma^q_{i+1}$-RFN($G^*_i$) $= V^1 + \Sigma^q_{i+1}$-RFN(CFG^*), and*

- *$TV^i = V^1 + \Sigma^q_{i+1}$-RFN($G^*_{i+1}$).*

Proof. Let $\mathcal{T} = V^1 + \Sigma^q_{i+1}$-RFN$(G^*_i)$ First we want to show that $V^i \subseteq \mathcal{T}$. This follows from the fact that V^i proves Σ^q_{i+1}-RFN(G^*_i) (Theorem 5.1.2) and it is an extension of V^1.

We also need to show that $\mathcal{T} \subseteq V^i$. Since V^i can is axiomatized by Σ^B_{i+1} formulas, we need to show that every Σ^B_{i+1} theorem of V^i is provable in $\mathcal{T}$. From the translation theorem (Theorem 2.3.3), we have that if V^i proves ϕ then G^*_i has polynomial-size proofs of the translation of ϕ. Then the fact that $\mathcal{T}$ proves ϕ follows from the following claim:

Claim 5.2.4 (Lemma 3.3 [23]). $\mathcal{T}$ proves the following: If ϕ is a Σ^B_{i+1} formula and for all $\vec{m}, \vec{n}$ there is G^*_i proof of $||\phi(\vec{x}, \vec{X})||[\vec{m}; \vec{n}]$, then $\mathcal{T}$ proves $\forall \vec{X} < \vec{n} \phi(\vec{m}, \vec{X})$.

$V^1 + \Sigma^q_{i+1}$-RFN$(G^*_i) = V^1 + \Sigma^q_{i+1}$-RFN$(CFG^*)$ follows from Lemma 5.2.1.

$TV^i = V^1 + \Sigma^q_i$-RFN(G^*_i) is proved in the same way as the first equality. □

Chapter 6

Computational Complexity and G

One of the most informative ways of understanding the strength of a theory of bounded arithmetic is to look at it from a computational complexity perspective. As was mentioned earlier, many theories of bounded arithmetic have a corresponding complexity class. This correspondence was made using the witnessing theorems. If two theories are the same, then their corresponding complexity classes are the same. In our work, we will do the same, but for the fragments of G. The specific problem is to find the complexity class for which the following problem is complete: Given a G_i^* proof of a Σ_j^q formula, find a witness for the outermost existential quantifiers. We call this problem $\mathrm{Wit}[G_i^*, \Sigma_j^q]$. Note that this is not necessarily a function, but it is a total search problem. The reason is that there is not always a unique witness.

To make things easier, we will use $\Sigma_j^B(T)$ to refer to the set of search problems that are Σ_j^B definable in the theory T.

An initial guess would be that $\mathrm{Wit}[G_i^*, \Sigma_j^q]$ is closely related to the $\Sigma_j^B(V^i)$. This is because we typically view G_i^* as the non-uniform version of V^i. As a matter of fact, it is true for $j \leq i + 1$.

Theorem 6.0.5. *Let $j \leq i+1$. Then $Wit[G_i^*, \Sigma_j^q]$ is complete for $\Sigma_j^B(V_i^*)$.*

Proof. First let π be a G_i^* proof of a Σ_j^q formula. Then to find a witness for the formula, find a witness for the Σ_j^B formula Σ_j^q-RFN(G_i^*), which is a theorem of V^i (Theorem 5.1.2). This shows that $\mathrm{Wit}[G_i^*, \Sigma_j^q] \in \Sigma_i^B(V^i)$.

Let $\phi(X)$ be a Σ_j^B theorem of V^i. Then, given X, ϕ can be witnessed by witnessing $||\phi||[|X|]$. We can get a G_i^* proof of $||\phi||[|X|]$ in polynomial time by the translation theorem (Theorem 2.3.3). This shows that $\mathrm{Wit}[G_i^*, \Sigma_j^q]$ is hard for $\Sigma_j^B(V^i)$. $\square$

You may now be wondering if this also holds for $j > i + 1$, and, in fact, Morioka was able to show that it is unlikely to hold.

Theorem 6.0.6 (Theorem 2.24, Theorem 8.8 in [29]). *Let $j > i + 1$. If $Wit[G_i^*, \Sigma_j^q] \in \Sigma_j^B(V^i)$, then PH collapses.*

The remainder of this chapter is divided into two sections: one for when $j > i + 1$ and one for when $j \leq i + 1$.

6.1 Witnessing Complex Formulas

In this section, we want to determine the exact complexity of $\mathrm{Wit}[G_i^*, \Sigma_j^q]$ when $j > i+1$. We look at these problems because they have been the main tool used to get conditional separations in bounded arithmetic and these proof systems. For example, the KPT witnessing theorem is used to show that V^∞ does not collapse unless PH collapses as well.

We can look at this problem from two perspectives. First we will show a negative result. We show that the complexity of witnessing complex formulas, using the standard notion of oracle Turing Machines, does not depend on the proof system being used but only on the complexity of the formula being proved. As a result, witnessing problems for these proof systems cannot be used to separate them when proving complex formulas.

With this in mind, we change our model of computation to a two player system. The players are a student and a teacher who responds to questions from the student. Then we can characterize the witnessing problem by the complexity of the student and the complexity of the questions asked.

In the standard model of computation, Morioka did some work in this area already [29]. He was able to show that $\mathrm{Wit}[G_i^*, \Sigma_j^q] \in FP^{\Sigma_{j-1}^p}[Wit, \log]$ and that $\mathrm{Wit}[G_i^*, \Sigma_j^q]$ is hard for $FP^{\Sigma_{j-1}^p}[Wit, O(1)] = \Sigma_j^B(V^i)$. This gives a range for the complexity.

Previous results in this thesis make it easy to prove that $\mathrm{Wit}[G_i^*, \Sigma_j^q]$ is in fact complete for the upper-end of this range.

Theorem 6.1.1. *When $j > i+1$, Wit$[G_i^*, \Sigma_j^q]$ is complete for $FP^{\Sigma_{j-1}^p}[Wit, \log]$.*

The proof of this theorem is divided into two two parts. First we must show that $\mathrm{Wit}[G_i^*, \Sigma_j^q]$ is in $FP^{\Sigma_{j-1}^p}[Wit, \log]$. It was already mentioned that Morioka showed the problem is in the class, but we can prove it a different way. Using the Herbrand theorem for G_i^*, we get a different proof.

Lemma 6.1.2. *When $j > i+1$, Wit$[G_i^*, \Sigma_j^q]$ is in $FP^{\Sigma_{j-1}^p}[Wit, \log]$*

Proof. Let π be a G_i^* proof of a Σ_j^q formula $\exists \vec{x} A(\vec{x}, \vec{p})$. By the Herbrand Theorem for G_i^* (Theorem 3.2.16), there is a proof π' of a sequent of the form

$$\Lambda \rightarrow A^*$$

meeting the conditions of that theorem. Informally, this gives us an interaction between a student and a teacher. The number of rounds is polynomial in the size of the proof. Our algorithm asks queries of the from "Can the teacher respond to the first i queries of the student?" (a Σ_{j-1}^p query). Using binary search, the algorithm finds the first round in which the teacher cannot respond. The use of binary search ensures we only make a logarithmic number of queries. Since these are witness queries, the oracle responds to the final query of the student with the witness we want. See Theorem 5.1.2 for the reason this is true. $\square$

The other part of the problem is to show that the problem is hard for the class.

Lemma 6.1.3. *When $j > i+1$, $Wit[G_i^*, \Sigma_j^q]$ is hard for $FP^{\Sigma_{j-1}^p}[Wit, \log]$ with respect to polynomial-time many-one reductions.*

Proof. We use a reduction similar to the reduction in Lemma 5.2.1 to reduce $\mathrm{Wit}[G_{j-1}^*, \Sigma_j^q]$ to $\mathrm{Wit}[G_i^*, \Sigma_j^q]$. From the translation theorem (Theorem 2.3.3), we already know that $\mathrm{Wit}[G_{j-1}^*, \Sigma_j^q]$ is hard for $FP^{\Sigma_{j-1}^p}[Wit, \log]$. Given a G_{j-1}^* proof of a Σ_j^q formula B, we construct a cut-free proof of $B \bigwedge \forall(C_i \wedge \neg C_i)$, where $C_0, \ldots, C_n$ are all of the cut formulas in the original proof (see Lemma 5.2.1 for this construction). Then witnessing this proof gives us a witness for B. □

In this last proof, we see why the complexity of the witnessing problem does not change as long as $i < j$. The work of the student can be incorporated in the oracles. When $i = j-1$, the student will determine if a cut formula is true or false on his own. However, the construction in the proof shows how the student can pass that work on to the teacher. Informally, this can be thought of as the student not doing the work, but simply getting an answer from the teacher. In order to distinguish between G_i^* and G_{i+1}^*, we need a model of computation where the teacher will not answer any question, but only certain pointed questions. This leads to the following definition.

Definition 6.1.4. Let C_1 and C_2 be two complexity classes such that $C_1 \subset C_2$. A STUDENT-TEACHER(C_1, Π_i^q) Turing Machine is a polynomial-time Turing Machine with two oracle tapes. The input is a quantified propositional sentence A. One oracle tape answers yes/no queries in C_1. The other takes as input a partial instance of A that is a Π_j^q formula, $j \leq i$, and returns a counter-example to the outermost universal quantifiers if the formula is false. This idea comes from [20].

Then a family of quantified propositional formulas Φ is in STUDENT-TEACHER(C_1, Π_i^q) if there is a STUDENT-TEACHER(C_1, Π_i^q) Turing Machine that witnesses every formula in Φ.

Informally, this can be viewed as student-teacher computations. The complexity class C_1 corresponds to the intelligence of the student, and the second complexity class refers to the questions to the teacher. In this model, a student cannot pass off the work to the teacher by asking the teacher for a counter-example to a different formula. We believe this model corresponds better to the witnessing problem for G_i^*.

Theorem 6.1.5. *Let Φ be a family of Σ_j^q formulas. Then if G_i^* has polynomial-size proofs of the formulas in Φ, then Φ is in $STUDENT\text{-}TEACHER(\Sigma_{i-1}^P, \Pi_{j-1}^q)$.*

Proof. Follows directly from the Herbrand Theorem for G_i^* (Theorem 3.2.16). See Theorem 5.1.2 for how to set up the queries. □

6.2 Witnessing Simple Formulas

As has already been mentioned, witnessing G_i^* proofs of simple formulas is complete for the set of problems definable in V^i. This gives us one characterization of Wit$[G_i^*, \Sigma_j^q]$ for $j < i$. On its own, this is not very informative. However, if we combine this with characterizations of the search problems that are definable in V^i, we get some insight into the proof systems. For example, in [25], the Σ_1^B theorems of V^3 are characterized using an extension of polynomial local search (PLS). More recently in [39], the Σ_1^B theorems of V^i are characterized using a statement about winning strategies in games. As well, in [37], the Σ_1^B theorems of V^i are characterized by a different type of principle about playing a game on multiple boards at the same time.

In this section, we reprove the result from [25]. The proof we use is different and is based on the Herbrand Theorem for G_i^*. The idea is that this proof may generalize to witnessing any class of formulas in V^i.

The first step is to define the complexity class we are using. The complexity class is an extension of the class PLS, which stands for polynomial local search. Generally, you can think of the input to a PLS problem is a large directed, acyclic graph. The graph is given by a polynomial time function that given a node will output a neighbor if one exists. The problem is to find a sink in this graph. One way of finding a sink is to start with a particular node, and keep following the neighbor function until you reach a sink. However, since the size of the graph is exponential in the size of the input, this is an exponential time and polynomial space algorithm. On the other hand, we could non-deterministically guess a node and check if it is a sink, and, if it is, output that node. The complexity comes from the difficulty of telling whether this problem has at least one output for every input.

In [25], PLS was generalized to $CPLS$, colour PLS, in order to characterize the Σ_1^B consequences of TV^2. This is the class we use except we allow the use of an oracle. Informally, the input to a $CPLS$ is a large directed, acyclic graph where each node is colored with a number of colours. You are told that, as you follow the neighbor function, you do not reach any new colours. You can easily find a colour for the sinks, but not the other node. The problem is to find a colour for a given node. One way of doing this is to start at the given node, and follow the neighbor function until we reach a sink. Then the colour of that sink will also be a colour of the original node. Again this is exponential time. You can also non-deterministically guess a colour.

Definition 6.2.1 ($CPLS^A$)**.** Let w be the input to the problem. Then a problem in $CPLS^A$ is defined by

- a neighbor function $N(v, w)$ (given as a P TM),
- a leaf relation $L(v, w)$ (given as a P TM),
- a colour relation $C(v, c, w)$ (given as a P^A TM),
- a function $e(v, w)$ that colours leaves (given as a P^A TM), and
- a source node a.

The output is one of the following

1. a non-leaf node v with a neighbor that is not smaller than v ($\neg L(v,w) \wedge N(v,w) \geq v$),

2. a node whose neighbor has a colour that the nodes does not have ($\forall c\, \neg C(v,c,w) \wedge \exists c\, C(N(v,w),c,w)$),

3. a leaf that is not properly colored ($L(v,w) \wedge \neg C(v,e(v,w),w)$), or

4. a colour for the source node a ($C(a,c,w)$).

We then get the following characterization of the theorems of TV^i.

Theorem 6.2.2. *For $i \geq 1$, a problem is Σ_i^B definable in TV^{i+1} if and only if the problem is in $CPLS^{\Sigma_{i-1}^p}$.*

Proof of the if direction. Suppose we have a problem in $CPLS^{\Sigma_{i-1}^p}$. This problem is defined by saying

$$\exists V \exists C[\text{“one of conditions 1-4 is satisfied”}].$$

This is a Σ_i^B formula since all of the functions and relations are in $P^{\Sigma_{i-1}^p}$. So now we must prove that V and C always exist. In TV^{i+1}, suppose the source node does not have a colour (i.e. assume 4 is not possible). Find the minimum node V that does not have a colour. This is Π_i^B-STRING-MIN. If V is a leaf, then we have a leaf that is not properly colored (condition 3 is true). If it is not a leaf, then either $N(V)$ is not smaller than V (condition 1) or $N(V)$ has a colour that V does not have (condition 2). □

Proof of the only if direction. This direction is more difficult; however, with the tools we have available, it is not too difficult. We will give the proof for the special case when $i = 1$, but the proof easily generalizes. Suppose TV^2 proves a Σ_1^B formula $\phi(X)$. Then, by the translation theorem (Theorem 2.3.3), there are polynomial-size G_3^* proofs of the translation of this formula. So to witness ϕ, we must find a witness for these G_3^* proofs.

Let π be a G_3^* proof of a Σ_1^q formula $\exists\vec{x}A(\vec{x})$, where A is Σ_0^q. By the Herbrand Theorem for G_3^* (Theorem 3.2.16), there is a G_2^* proof π' of a sequent

$$\Lambda_1' \rightarrow \exists\vec{x}A(\vec{x})$$

meeting the conditions of that theorem. Let E_1 be the set of extension variables defined by Λ_1'. Recall that Λ_1' is a series of formulas of the form

$$e \leftrightarrow \forall\vec{x}\exists\vec{y}E(\vec{x},\vec{y}) \tag{6.2.1}$$

where E is a Σ_0^q formula. Note that the Herbrand Theorem says that the formula defining e is a Σ_2^B formula, but it is a simple task to change this to a Π_2^B, which we use in this case.

Now apply the Herbrand Theorem to π'. This gives us a G_1^* proof of a sequent

$$\Lambda_1, \Lambda_2 \rightarrow \exists\vec{x}A(\vec{x}),$$

where Λ_2 is the new extension cedent and Λ_1 is obtained from Λ_1'. Let E_2 be the set of extension variables defined by Λ_2 and let $\prec$ be the ordering of the variables. Now Λ_1 is now a series of formulas of the form

$$\neg e \supset \forall\vec{y}\neg E(\vec{q},\vec{y}) \wedge e \supset \exists\vec{y}E(\vec{e}',\vec{y}),$$

where $\vec{e}' \in E_2$. This formula was obtained from (6.2.1) by witnessing the outermost $\forall$ quantifiers by new extension variables, and replacing the outermost $\exists$ quantifies by eigenvariables. The formulas in Λ_2 are of the form

$$e \leftrightarrow \exists\vec{y}E(\vec{y})$$

where E is a Σ_0^q formula. The algorithm searches for values for the eigenvariables and extension variables that satisfy Λ_1 and Λ_2. Once this is done, we can get a witness for $\exists\vec{x}A(\vec{x})$ using the G_1^* witnessing algorithm.

In our $CPLS$ algorithm, a node is given by a tuple $(\sigma_{E_1}, \sigma_{E_2}, \sigma_\exists, \sigma_Q)$. The assignment σ_{E_1} is an assignment to E_1, σ_{E_2} is an assignment to E_2, and σ_Q is an assignment to the eigenvariables. Finally, $\sigma_\exists$ is an assignment to the existential quantifier in Λ_1 and Λ_2. The nodes can be ordered using a lexicographic order for the assignments, where smaller variables, according to $\prec$, are written first. So now to define the different parts of the $CPLS$ problem that witnesses this proof.

We define three different types of colours. First, a colour is a witness for $\exists\vec{x}A(\vec{x})$. Second, for variables $e \in E_2$, consider the defining formula

$$e \leftrightarrow \exists\vec{y}E(\vec{y})$$

from Λ_2. This is equivalent to

$$(e \supset \exists\vec{y}E(\vec{y})) \wedge (\neg e \supset \forall\vec{y}\neg E(\vec{y})).$$

Setting e to false implies $\forall\vec{y}\neg E(\vec{y})$, so, when e is assigned false, a colour is an assignment to $\vec{y}$ that proves this is false. The assignment $\sigma_\exists$ assigns values $\vec{v}$ to the existentially quantified $\vec{y}$. When e is assigned true, a colour is an assignment $\vec{v}'$ larger than $\vec{v}$ such that $\neg E(\vec{v}')$. Informally, if we assign $\vec{y}$ the value $\vec{v}$, then we are saying that, for all values larger than $\vec{v}$, E is false. A colour is a counter-example that proves this is not true.

Third, there are colours for variables $e \in E_1$. Consider the defining formula for e in Λ_1:

$$(e \supset \exists\vec{y}E(\vec{e}', \vec{y})) \wedge (\neg e \supset \forall\vec{y}\neg E(\vec{q}, \vec{y})),$$

where $\vec{e}' \in E_2$. A colour is essentially the same as before. If e is true, then a colour is proof that $\forall\vec{y}\neg E(\vec{q},\vec{y})$ is false. If e is false, then a colour is an assignment $\vec{v}'$ larger than the current assignment to $\vec{y}$ such that $E(\vec{e}',\vec{v}')$.

The neighbor function finds an extension variable that does not have an appropriate value yet, and makes some progress in the search for an appropriate value. So, the first thing this function does is find, if it can, the smallest eigenvariable $e \in E_2$ such that $e \supset E(\vec{v})$ is not satisfied by the current set of assignments, where $\exists\vec{y}E(\vec{y})$ is the formula defining e in Λ_2 and $\vec{v}$ is the assignment $\sigma_\exists$ assigns to $\vec{y}$. If $\vec{v}$ is not the lexicographically smallest assignment, the neighbor is the assignment where $\vec{v}$ is given the next smallest assignment. It is a simple task to verify that this does not add any colours. If $\vec{v}$ is the lexicographically smallest assignment, the neighbor will

- assign e the value false,
- assign true to every $e' \in E_2$ larger than e,
- $\sigma_\exists$ will assign true to every variable.

The idea is that if $\vec{v}$ is the smallest assignment, then we must have checked every possible assignment to $\vec{y}$. This means that $\forall\vec{y}\neg E(\vec{y})$ is true. We reset the value for e' because it may depend on e, in which case, we may need to change its value. For the same reason, we reset the $\sigma_\exists$ to restart our searches. Note that we do not need to change σ_{E_1}. If $e'' \in E_1$ is true, then we reset the search, so no new colours could exists. If e'' is false, then the defining formula for e'', $\forall\vec{y}\neg E(\vec{q},\vec{y})$ does not mention any variables in E_2 and will therefore not change.

If the algorithm is unable to find an appropriate $e \in E_2$, then it searches for an $e \in E_1$ such that $e \supset E(\vec{e}',\vec{v})$ is not satisfied by the current assignments. If $\vec{v}$, is not the lexicographically smallest assignment, then the function sets it to the next smallest. Otherwise, the neighbor will

- assign e the value false,

- set $\vec{q}$ to the current value of $\vec{e}'$,
- assign true to every variable in E_1 larger than e,
- assign true to every variable in E_2, and
- $\sigma_\exists$ will assign true to every variable.

The idea is similar to the first case. The main difference is the change to $\vec{q}$. This is done because, if $\vec{v}$ is the smallest assignment, then we must have checked every possible assignment to $\vec{y}$. This means that $\forall\vec{y}\neg E(\vec{e}',\vec{y})$ is true. So we set $\vec{q}$ to $\vec{e}'$ so that $\forall\vec{y}\neg E(\vec{q},\vec{y})$ will be true as well.

If we cannot find an appropriate $e \in E_1$, then this assignment is a leaf. For leaves, we must be able to find a colour. This is done in polynomial time using the Σ_1^q witnessing algorithm for G_1^* (Theorem 2.3.7 and 2.2.4). The assignments we use are given by $\sigma_\exists$, σ_{E_1}, σ_{E_2}, and σ_Q. This covers all of the free variables as well as all of the existentially quantified variables on the left side of the sequent. Since the current node is a leaf, we know we have appropriate witnesses for these variables. The output of the algorithm is a witness for $\exists\vec{x}A(\vec{x})$ or a counter example for $\neg e \supset \forall\vec{y}\neg E$ for some extension variables e. In either case, it is a colour.

The source node is the lexicographically largest assignment. So, all of the extension variables, eigenvariables, and existential quantifiers are assigned true. Note that the only possible colours for the source node are witnesses for $\exists\vec{x}A(\vec{x})$. The output to the $CPLS$ algorithm has to be a colour for the source, which gives us a witness for $\exists\vec{x}A(\vec{x})$. $\square$

Part III

Defining New Proof Systems

Chapter 7

A Proof System for L

In this part of the thesis, we will be defining new proof systems. The idea is that the restrictions are meant to show where the strength of the proof systems actually comes from. The first example is GL^*. This was first defined in [34]. In that work, the translation theorem was proved, but the reflection principles were not proved. That will be done now. We begin by giving the definition necessary for this section, then we prove the main result.

7.1 A Universal Theory For L Reasoning

One way to get a theory for L is to define a universal theory with a language that contains a function symbol for every function in FL. Then, we get a theory for L by taking the defining axioms for these functions. This is the idea behind the theory PV and $\overline{V^0}$. In our case, we characterize the FL functions using Lind's characterization [27]. We will only give the definitions. For a more thorough exposition see Chapter 9 in [11].

In the next definition, we define the set of function symbols in $\mathcal{L}_{FL}$ and give their intended meaning.

Definition 7.1.1. The language $\mathcal{L}_{FL}$ is the smallest language satisfying

1. $\mathcal{L}_A^2 \cup \{pd, \min\}$ is a subset of $\mathcal{L}_{FL}$ and have defining axioms 2BASIC, and the axioms

$$pd(0) = 0 \tag{7.1.1}$$

$$pd(x+1) = x \tag{7.1.2}$$

$$min(x, y) = z \leftrightarrow (z = x \wedge x \leq y) \vee (z = y \wedge y \leq x) \tag{7.1.3}$$

2. For every open formula $\alpha(i, \vec{x}, \vec{X})$ over $\mathcal{L}_{FL}$ and term $t(\vec{x}, \vec{X})$ over $\mathcal{L}_A^2$, there is a string function $F_{\alpha,t}$ in $\mathcal{L}_{FL}$ with bit defining axiom

$$F_{\alpha,t}(\vec{x}, \vec{X})(i) \leftrightarrow i < t(\vec{x}, \vec{X}) \wedge \alpha(i, \vec{x}, \vec{X}) \tag{7.1.4}$$

3. For every open formula $\alpha(z, \vec{x}, \vec{X})$ over $\mathcal{L}_{FL}$ and term $t(\vec{x}, \vec{X})$ over $\mathcal{L}_A^2$, there is a number function $f_{\alpha,t}$ in $\mathcal{L}_{FL}$ with defining axioms

$$f_{\alpha,t}(\vec{x}, \vec{X}) \leq t(\vec{x}, \vec{X}) \tag{7.1.5}$$

$$z < t(\vec{x}, \vec{X}) \wedge \alpha(z, \vec{x}, \vec{X}) \supset \alpha(f_{\alpha,t}(\vec{x}, \vec{X}), \vec{x}, \vec{X}) \tag{7.1.6}$$

$$z < f_{\alpha,t}(\vec{x}, \vec{X}) \supset \neg\alpha(z, \vec{x}, \vec{X}) \tag{7.1.7}$$

4. For every number function $g(\vec{x}, \vec{X})$ and $h(y, \vec{x}, \vec{X}, p)$ in $\mathcal{L}_{FL}$ and term $t(y, \vec{x}, \vec{X})$ over $\mathcal{L}_A^2$, there is a number function $f_{g,h,t}(y, \vec{x}, \vec{X})$ with defining axioms

$$f_{g,h,t}(0, \vec{x}, \vec{X}) = \min(g(\vec{x}, \vec{X}), t(\vec{x}, \vec{X})) \tag{7.1.8}$$

$$f_{g,h,t}(y+1, \vec{x}, \vec{X}) = \min(h(y, \vec{x}, \vec{X}, f(y, \vec{x}, \vec{X})), t(\vec{x}, \vec{X})) \tag{7.1.9}$$

The last scheme is called p-bounded number recursion. The p-bounded number recursion is equivalent to the *log*-bounded string recursion given in [27]. The other schemes come from the definition of $\mathcal{L}_{FAC^0}$ in [8].

It is not difficult to see every function in $\mathcal{L}_{FL}$ is in FL. The only point we should note is that the intermediate values in the recursion are bounded by a polynomial in the size of the input. This means, if we store intermediate values in binary, the space used is bounded by the log of the size of the input. So the recursion can be simulated in L. To show that every FL function has a corresponding function symbol in $\mathcal{L}_{FL}$, note that the p-bounded number recursion can be used to traverse a graph where every node has out-degree at most one.

$\overline{VL}$ is defined over the language $\mathcal{L}_{FL}$. This ensures there is a function symbol for every function in FL. As for the axioms, $\overline{VL}$ has the defining axioms for every function in FL, and a modified version of the 2BASIC axioms. $B14$ is removed because of the existential quantifier. It is replaced with the two defining axioms 7.1.1 and 7.1.2, which can be used to prove $B14$. Now we define $\overline{VL}$.

Definition 7.1.2. $\overline{VL}$ is the theory over the language $\mathcal{L}_{FL}$ with B1-B13 plus 7.1.1; 7.1.2; axiom 7.1.4 for each string function $F_{\alpha,t}$ in $\mathcal{L}_{FL}$; axioms 7.1.5, 7.1.6, and 7.1.7 for each number function $f_{\alpha,t}$ in $\mathcal{L}_{FL}$; axioms 7.1.8 and 7.1.9 for each number function $f_{g,h,t}$ in $\mathcal{L}_{FL}$; and open($\mathcal{L}_{FL}$)-IND.

An open formula is a formula that does not have any quantifiers.

The important part of this theory is the it really is a universal version of VL.

Theorem 7.1.3. *$\overline{VL}$ is a conservative extension of VL.*

7.2 Definition of GL^*

In this section, we will define the proof system we wish to explore. This proof system is be defined by restricting cut formulas to a set of formulas that can be evaluated in L.

Alone that is not enough to change the strength of the proof system, so we also restrict the use of eigenvariables.

The first step is to define a set of formula that can be evaluated in L. These formula will be bases on $CNF(2)$ formulas. A $CNF(2)$ formula is a CNF formula where no variable has more than two occurrences in the entire formula. It was shown in [17] that determining whether or not a given $CNF(2)$ formula is satisfiable is complete for L. Based on this we get the following definition:

Definition 7.2.1. The set of formulas $\Sigma CNF(2)$ is the smallest set

1. containing Σ_0^q,
2. containing every formula $\exists\vec{z}, \phi(\vec{z}, \vec{x})$ where (1) ϕ is a quantifier-free CNF formula $\bigwedge_{i=1}^m C_i$ and (2) existence of a z-literal l in C_i and C_j, $i \neq j$, implies existence of an x-variable x such that $x \in C_i$ and $\neg x \in C_j$ or vice versa, and
3. closed under substitution of Σ_0^q formulas that contain only x-variables for x-variables.

Definition 7.2.2. GL^* is the propositional proof system G_1^* with cuts restricted to $\Sigma CNF(2)$ formulas in which every free variable in a non-Σ_0^q formula is a parameter variable.

The restriction on the free variables in the cut formula might seems strange, but it is necessary. If we did not have this restriction, then the proof system would be as strong as G_1^*. This is demonstrated by the connection between GPV^* and G_1^* (Theorem 4.1.2).

7.3 Proving The Reflection Principles

In this section, we show that GL^* does not capture reasoning for a higher complexity class. This is done by proving, in VL, that GL^* is sound. This idea comes from [7], where Cook showed that PV proves extended-Frege is sound, and [23], where Krajícek and Pudlák showed T_2^i proves G_i is sound for $i > 0$.

We will actually show that $\overline{VL}$ proves GL^* is sound. The idea behind the proof is to give an $\mathcal{L}_{FL}$ function that witnesses the quantifiers in the proof. Then we prove, by $\Sigma_0^B(\mathcal{L}_{FL})$-IND, that this functions witness every sequent, including the final sequent. Therefore the formula is true.

We start by giving an algorithm that witnesses $\Sigma CNF(2)$ formulas in L when the formula is true. This algorithm is the algorithm given in [17] with a few additions to find the satisfying assignment. We describe an $\mathcal{L}_{FL}$ function that corresponds to this algorithm and prove it correct in $\overline{VL}$. We then use this function to find an $\mathcal{L}_{FL}$ function that witnesses GL^* proofs, and prove it correct in $\overline{VL}$.

7.3.1 Witnessing $\Sigma CNF(2)$ Formulas

Let $\exists\vec{z}A(\vec{x},\vec{z})$ be a $\Sigma CNF(2)$ formula. We will describe how to find a witness for this formula. We assume that A is a CNF formula. That is, the substitution of the Σ_0^q formulas has not happened. The general case is essentially the same.

The first thing to take care of is the encoding of A. We will not go through this in detail. Suffice it to say that parsing a formula can be done in TC^0 [10], and, as long as we are working in a theory that extends TC^0 reasoning, we can use any reasonable encoding. We will refer to the ith clause of A as C_i^A. A clause will be viewed as a set of literals. A *literal* is either a variable or its negation. So we will write $l \in C_i^A$ to mean that the literal l is in the ith clause of A. Since the parsing can be done in TC^0, these formulas can be defined by $\Sigma_0^B(\mathcal{L}_{FL})$ formulas. An assignment will also be viewed as a set of literals. If a literal is in the set, then that literal is true. So an assignment X satisfies a clause C if and only in $X \cap C \neq \emptyset$.

Given values for $\vec{x}$, we first simplify A to get a $CNF(2)$ formula. We will refer to the simplified formula as F. This can be done using the $\mathcal{L}_{FL}$ function defined by the following formula:

$$l \in C_i^F \leftrightarrow l \in C_i^A \wedge X \cap C_i^A = \emptyset,$$

where X is the assignment to the free variables. From the definition of a $\Sigma CNF(2)$ formula, $\overline{VL}$ can easily prove that F now encodes a $CNF(2)$ formula. In fact, it can be shown that no literal appears more than once. A satisfying assignment to this formula is the witness we want. Mark Braverman gave an algorithm for finding this assignment [1], but we use a different algorithm that is easier to formalize.

Before we describe the algorithm that finds this assignment, we go through a couple of definitions. First, a *pure literal* is a literal that appears in the formula, but its negation does not. Next the formula imposes an order on the literals. We say a literal l_1 *follows* a literal l_2 if the clause that contains l_1 also contains l_2, and l_1 is immediately to the right of l_2, circling to the beginning if l_2 is the last literal. More formally:

$$\begin{aligned} follows(l_1, l_2, F) \leftrightarrow \exists i, l_1 \in C_i^F \wedge l_2 \in C_i^F &\wedge \forall l_3(l_2 < l_3 < l_1 \supset l_3 \notin C_i^F) \\ &\wedge \forall l_3(l_3 < l_1 < l_2 \supset l_3 \notin C_i^F) \\ &\wedge \forall l_3(l_1 < l_2 < l_3 \supset l_3 \notin C_i^F) \end{aligned}$$

Note that if a clause contains a single literal then that literal follows itself. Also, note that literals are coded by numbers and $l_1 < l_2$ means the number coding l_1 is less then the number coding l_2.

To find the assignment to F, we will go through the literals in the formula in a very specific order. Starting with a literal l that is not a pure literal, the *next literal* is the literal that follows $\bar{l}$:

$$next(l_1, F) = l_2 \leftrightarrow follows(l_2, \bar{l}_1, F).$$

Note that if l_1 is a pure literal, then there is no next literal, so we simply define it to be itself. The important distinction is that *next* gives an ordering of the literals in a formula, and $follows$ orders the literal in a clause. When F is understood, we will not mention F in *next* and $follows$.

The algorithm that finds the assignment works in stages. At the beginning of stage

i, we have an assignment that satisfies the first $i-1$ clauses. Then, in the ith stage, we make local changes to this assignment to satisfy the ith clause as well. At a high level, to satisfy the ith clause, we start with the first literal in the ith clause, and assign that literal to true. The clause that contains this literal's negation may have gone from being satisfied to being unsatisfied. So we now go to the next literal, which is in this other clause. We continue this until we get to a point where we know the other clause is satisfied. We need to be able to do this in L. Algorithm **??** shows how to do this. At any

Algorithm 1 Algorithm for Stage i

```
Set $l_1$ to the first literal in clause $i$.
repeat
  Assign true to $l_1$.
  set $l_2 := next(l_1)$
  while $l_2$ is not the complement of $l_1$ do
    Assign true to $l_2$
    set $l_2 := next(l_2)$
    If $l_2$ is a pure literal, assign true to $l_2$, and stage $i$ is done.
    If $l_1$ and $l_2$ are in the same clause, stage $i$ is done.
  end while
  Assign true to $l_1$. {This statement is redundant, but it is included to emphasis that
  $l_1$ is true.}
  set $l_1 := next(l_1)$
until $l_1$ is the first literal in clause $i$
At this point we know the formula is unsatisfiable.
```

point in the algorithm, the only information we need are the values of l_1 and l_2, so this is in L. Note that we do not store the assignment on the work tape, but on a write-only, output tape. What is not obvious is why this algorithm works.

The next lemma can be used to show that the both loops will eventually finish.

Lemma 7.3.1. *For all literals l, there exists a $t > 0$ such that after t applications of next to l, we get to l or a pure literal.*

Proof. Let $next^0(l) = l$ and $next^{t+1}(l) = next(next^t(l))$. Since $next$ has a finite range, there exist a minimum i and t such that $next^i(l) = next^{i+t}(l)$. Suppose this is not a pure literal. If $i > 0$, then $next(next^{i-1}(l)) = next(next^{i+t-1}(l))$. However, this implies

$next^{i-1}(l) = next^{i+t-1}(l)$ since $next$ is one-to-one when not dealing with pure literals. This violates our choice of i. Therefore $i = 0$, and $l = next^0(l) = next^t(l)$. □

The implies the inner loop will halt, because, if it does not end earlier, l_2 will eventually equal l_1 which both will be in the same clause. For the outer loop, if the algorithm does not halt for any other reason, l_1 will eventually return to the first literal in the ith clause.

The next lemma plays a small role in the proof of correctness.

Lemma 7.3.2. *Suppose the algorithm fails at stage i and that $next^t(l') = l$, where l' is the first literal in clause i. Then, for every literal in the same clause as l, there is a t' such that $next^{t'}(l')$ equals that literal.*

Proof. To prove this lemma, we will show that there exists a t' that equals the literal that follows l. Then by continually applying this argument, you get that every literal in the clause is visited.

Let l' be the first literal in the ith clause. Then, after going through the outer loop t times, $l_1 = l$. Since the algorithm fails, the inner loop will finish because $l_2 = \bar{l}_1$. This means there is a t' such that $next^{t'}(l') = \bar{l}$. Then $next^{t'+1}(l')$ is the literal that follows l. □

Theorem 7.3.3. *If the algorithm fails, the formula is unsatisfiable.*

Proof. This is proved by contradiction. Let F be a $CNF(2)$ formula and A be an assignment that satisfies it. Assume that the algorithm fails. From this we can defined a function from the set of variables to the set of clauses as follows:

$$f(i) = j \leftrightarrow (x_i \in C_j^F \wedge x_i \in A) \vee (\neg x_i \in C_j^F \wedge \neg x_j \in A).$$

Informally, if $f(i) = j$ then clause C_j is true because of the variable x_i. Since the formula is satisfied, this function is onto the set of clauses. Also, since F is $CNF(2)$, no literal appear more than once. So f is indeed a function because if $f(i) = j$ and $f(i) = j'$ then the literal x_i or $\neg x_i$ is in both C_j^F and $C_{j'}^F$.

Now we will use the assumption that the algorithm fails to find a way to restrict f so that it violates the PHP. Suppose the algorithm fails at stage i. Let l be first literal in clause i. We then define sets of variables V^a as follows:

$$V^a = \left\{x_n : \exists b < a\ next^b(l) = x_n \vee next^b(l) = \neg x_n\right\}.$$

We also defined sets of clauses W^t as follows:

$$W^a = \{C_n : \exists x \in V^a(x \in C_n \vee \neg x \in C_n)\}.$$

Note that for a large enough a, say $|F|$, if C_n is in W^a, then every variable that appears in C_n is in W^a by 7.3.2. We show by induction on a that $|V^a| < |W^a|$.

For $a = 1$, $|V^a| = 1$. If l is a pure literal or l and $\neg l$ are in the same clause, then the algorithm would succeed. Otherwise $|W^a| = 2$.

For the inductive case, suppose $|V^a| < |W^a|$. Let $l' = next^{a+1}(l)$. If l' is not a new variable, then $|V^{a+1}| = |V^a| < |W^a| = |W^{a+1}|$. If l' is a new variable, then $\bar{l}'$ must be in a new clause. For, if this was not the case, the algorithm would succeed. To see this, let l_1 be the most recent literal in the same clause as l'. We know l_1 is not $\bar{l}'$ since l' is a new variable. Then eventually l_2 will become $next(l')$, which is in the same clause as l_1. The inner loop will not end because l_2 becomes the complement of l_1 since that would mean $next(\bar{l}_1)$ is more recent.

This gives $|V^{a+1}| = |V^a| + 1 < |W^a| + 1 = |W^{a+1}|$.

If we restrict f to $V^{|F|}$, then f is a function from $V^{|F|}$ that is onto $W^{|F|}$ violating the PHP. □

Theorem 7.3.4. *If the algorithm succeeds, then, for all i, the assignment given at the end of stage i satisfies the first i clauses of F.*

Proof. The proof is done by induction on i. For $i = 0$, the statement holds since there are no clauses to satisfy. As an induction hypothesis, suppose the statement holds for i. Then we will show if the algorithm ever visits one of the literals in clause n, then that clause is satisfied.

Consider clause n, where $n \leq i+1$. Find the last point in the algorithm that either l_1 or l_2 was in clause n, and let l be that literal. First, it is possible that when the algorithm ends l_2 is in clause n. If l_2 is a pure literal, then l_2 is set to true, satisfying the clause. Otherwise, l_1 and l_2 are in the same clause. In this case, l_1 is true since it was assigned true. If l_2 ever became $\bar{l}_1$, the algorithm would exit the inner loop, so $\bar{l}_1$ could never have been assigned true.

Second, we consider the possibility that l_2 was not in clause n when the algorithm ended. Then we claim that l is true, and, therefore, clause n is satisfied. Suppose for a contradiction that it is not. Then at some later point $\bar{l}$ was assigned true. This could happen in one of three places. First is if $l_1 = l$ and we are at the beginning of the outer loop. However, l_2 would be set to $next(\bar{l})$ right after, which is in clause n. This means we did not find the last occurrence of a literal in clause n as we should have. A similar argument can be used in the other two places. □

We now turn to formalizing this algorithm. For this, we define an $\mathcal{L}_{FL}$ function $f(i,t)$ that will return the value of l_1 and l_2 after t steps in stage i. This is done using number recursion. In the following let $f(c,t) = \langle l_3, l_4 \rangle$:

$$f(i,0) = \langle l_1, l_2 \rangle \leftrightarrow l_1 = \min_{l} l \in C_i^F \wedge l_2 = next(l_1)$$

$$\begin{aligned} f(c,t+1) = \langle l_1, l_2 \rangle \leftrightarrow & \phi_1 \supset l_1 = next(l_3) \wedge l_2 = next(l_1) \\ & \wedge \neg\phi_1 \wedge \phi_2 \supset (l_1 = l_3 \wedge l_2 = l_4) \\ & \wedge \neg\phi_1 \wedge \neg\phi_2 \supset (l_1 = l_3 \wedge l_2 = next(l_4)) \end{aligned}$$

where

$$\phi_1 \equiv l_3 = \bar{l}_4$$

$$\phi_2 \equiv (\mathit{sameClause}(l_3, l_4) \vee \mathit{pureLiteral}(l_4))$$

The formulas ϕ_1 and ϕ_2 are the conditions that are used to recognize when the inner loop ends. The first formula is when the loop ends and we have to continue with the outer loop. The second formula is when the stage is finished. In the formula version, we do not stop if the algorithm fails. Instead we view the algorithm as failing if after $|F|^2$ steps, ϕ_2 was never true. We use this value since $|F|$ is an upper bound on the number of literals in F and current state of the algorithm is determined by a pair of literal. In the following, any reference to time has the implicit bound of $|F|^2$.

The final step is to extract the assignment. The assignment is done by finding the last time a variable is assigned a value. This means we must be able to determine when a variable is assigned a value. To do this, observe that a literal is assigned true just before the *next* function is applied to that literal. With this is mind we get the following:

$$\mathit{Assigned}(i, t, l) \leftrightarrow \exists l', f(i, t) = \langle \mathit{next}(l), l' \rangle \vee f(i, t) = \langle l', \mathit{next}(l) \rangle$$

So $\mathit{Assigned}(i, t, l)$ means that l was assigned true during the tth step of stage i. Then we can get the assignment as follows:

$$\begin{aligned} l \in \mathit{Assignment}(i, F) \leftrightarrow & c = \max_c \exists t \ \mathit{Assigned}(c, t, l) \\ & \wedge t = \max_t \mathit{Assigned}(c, t, l) \\ & \wedge c' = \max_{c'} \exists t' \ \mathit{Assigned}(c, t', \bar{l}) \\ & \wedge t' = \max_{t'} \mathit{Assigned}(c', t', \bar{l}) \\ & \wedge (c > c' \vee (c = c' \wedge t > t')) \end{aligned}$$

The idea is the value of a variable is the last value that was assigned to it.

The $\overline{VL}$ proof that this algorithm is correct is the essentially the same as the proofs of Theorem 7.3.3 and Theorem 7.3.4, which can be formalized in $\overline{VL}$. This gives the following.

Theorem 7.3.5. *$\overline{VL}$ proves that, if the algorithm fails, the formula is unsatisfiable.*

Theorem 7.3.6. *$\overline{VL}$ proves that, if the algorithm succeeds, then, for all i, $Assignment(i, F)$ gives a satisfying assignment to the first i clauses of F.*

7.3.2 Witnessing GL^* Proofs

Let π be a GL^* proof of a Σ_1^q formula $\exists\vec{z}P(\vec{x}, \vec{z})$, and let A be an assignment to the parameter variables. We assume π is in free variable normal form (Definition 2.3.2).

Let $\Gamma_i \to \Delta_i$ be the ith sequent in π. We will prove by induction that for any assignment to all of the free variables of Γ_i and Δ_i, a function $Wit(i, \pi, A)$ will find at least one formula that satisfies the sequent.

There are two things to note. By the subformula property, every formula in Γ_i is $\Sigma CNF(2)$, which means it can be evaluated. Also, we need an assignment that gives appropriate values to the non-parameter free variables that could appear. To take care of this second point, we extend A to an assignment A' as follows:

1: Given a non-parameter free variable y, find the $\exists$-left inference in π that uses y as an eigenvariable. Let z be the new bound variable and let F be the principal formula.

2: Find the descendant of F that is used as a cut formula. Let F' be the cut formula. Note that F is a subformula of F', and, because of the variable restriction on cut formulas, every free variable in F' is a parameter variable.

3: Assign y the value that $Assignment(F', A)$ assigns z.

The reason for this particular assignment will become evident in the proof of Lemma 7.3.7.

We can now define $Wit(i, \pi, A')$, which witnesses $\Gamma_i \to \Delta_i$. Wit will go through each formula in the sequent to find a formula that satisfies the sequent. $\Sigma CNF(2)$ formulas are evaluated using the algorithm described in the previous section. We will now focus our attention on other Σ_1^q formulas, which must appear in Δ_i. Each Σ_1^q formula $F =_{syn} \exists \vec{z} F^*(\vec{z})$ in Δ is evaluated by finding a witness to the quantifiers as follows:

1: Find a formula F' in π that is an ancestor of F, is satisfied by A', and is a Σ_0^q formula of the form $F^*(z_1/B_1, \ldots, z_n/B_n)$, where each B_i is Σ_0^q

2: z_i is assigned $\top$ if A' satisfies B_i, otherwise it is assigned $\bot$

3: if no such F' exists, then every bound variable is assigned $\bot$.

Lemma 7.3.7. *For every sequent $\Gamma_i \to \Delta_i$ in π, $Wit(i, \pi, A')$ finds a false formula in Γ_i or a witness for a formula in Δ_i.*

Proof. We prove the theorem by induction on the depth of the sequent. For the base case, the sequent is an axiom, and the theorem obviously holds. For the inductive step, we need to look at each rule. We can ignore $\forall$-left and $\forall$-right since universal quantifiers do not appear in π.

We will now assume all formulas in Γ_i are true and all $\Sigma CNF(2)$ formulas in Δ_i as false. So we need to find a Σ_1^q formula in Δ_i that is true.

Consider cut. Suppose the inference is

$$\frac{F, \Gamma \to \Delta \qquad \Gamma \to \Delta, F}{\Gamma \to \Delta}$$

First suppose F is true. By induction, with the upper left sequent, Wit witnesses one of the formulas in Δ. Then the corresponding formula in the bottom sequent is witnessed by Wit. This is because the ancestor of the formula in the upper sequent that gives the witness is also an ancestor of the corresponding formula in the lower sequent. If F is false, it cannot be the formula that was witnessed in the upper right sequent, and a similar argument can be made.

Consider $\exists$-right. Suppose the inference is

$$\frac{\Gamma \rightarrow \Delta, F(B)}{\Gamma \rightarrow \Delta, \exists z F(z)}$$

First suppose $F(B)$ is Σ_0^q. If it is false, we can apply the inductive hypothesis, and, by an argument similar to the previous case, prove one of the formulas in Δ must be witnessed. If $F(B)$ is true, then Wit will witness $\exists z F(z)$ since $F(B)$ is the ancestor that gives the witness. If $F(B)$ is not Σ_0^q, then we can apply the inductive hypothesis, and, by the same argument, find a formula that is witnessed.

The last rule we will look at is $\exists$-left. Suppose the inference is

$$\frac{F(y), \Gamma \rightarrow \Delta}{\exists z F(z), \Gamma \rightarrow \Delta}$$

To be able to apply the inductive hypothesis, we need to be sure that $F(y)$ is satisfied. If $\exists z F(z)$ it true, then we know $F(y)$ is satisfied by the construction of A': the value assigned to y is chosen to satisfy $F(y)$ if it is possible. Otherwise, $\exists z F(z)$ is false, and we do not need induction.

For the other rules the inductive hypothesis can be applied directly and the witness found as in the previous cases. □

Theorem 7.3.8. *$\overline{VL}$ proves GL^* is sound for proofs of Σ_1^q formulas.*

Proof. The functions $Assignment$ and Wit are in FL and can be formalized in $\overline{VL}$. A function that finds A', given A, can also be formalized since it in $\overline{VL}$. The final thing to note is that the proof of Lemma 7.3.7 can be formalized in $\overline{VL}$ since the induction hypothesis can be express as a $\Sigma_0^B(\mathcal{L}_{FL})$ formula and the induction carried out. □

The reason this proof does not work for a larger proof system, say G_1^*, is because *Assignment* cannot be formalized for the larger class of cut formulas. Also, if the variable restriction was not present, we would not be able to find A' in L, and the proof would, once again, break down.

Chapter 8

A Proof System for NL

In the previous chapter, we defined a proof system that was meant to capture reasoning for L. The method that was used seems like it could be generalized to other complexity classes. In this chapter, we do the same type of construction for NL. The main contributions of this section include a formalization of the Immerman-Szelepcsenyi Theorem in bounded arithmetic and a formalization of the reduction from 2-SAT to the complement of directed graph reachability. Both of these have been done in [18], but we present them in a simpler way. As well, for the reduction, there was an error in [18] which we fix.

8.1 Definition of GNL^*

As with GL^*, we want to restrict the cut formulas to a class of formulas that can be witnessed in NL. The class of formulas we defined are called the $\Sigma Krom$. The definition is derived from Grädel's descriptive complexity characterization of NL in [15].

Definition 8.1.1. The set of $\Sigma Krom$ formulas are formulas of the form

$$\exists z_1, \ldots, \exists z_n \phi(\vec{z}, \vec{x}),$$

such that $\phi(\vec{z}, \vec{x})$ is a quantifier free formula of the form $\bigwedge_{i=0}^{n} C_i$, where each C_i is the disjunction of at most 2 z-literals and a Σ_0^q formula that does not mention any z-variable.

Definition 8.1.2. The proof system GNL^* is the proof system G^* with cuts restricted to $\Sigma Krom$ formulas, and no non-Σ_0^q cut formula contains a free variable that is an eigenvariable.

8.2 NL and Bounded Arithmetic

8.2.1 Theories for NL

As with GL^*, we start with a theory known to capture reasoning for NL. We look at two theories that have already been considered, and a third that has not been considered. The first theory we will look at was first studied in [31]. The idea was to take V^0 and add an axiom that says there exists an output to a function that is complete for NL under AC^0 reductions.

The function that Nguyen used takes as input a graph with nodes $\{0, \ldots, a\}$ and edge relation E. The output of the function is a string Z such that $Z(k, i)$ is true if and only if there is a path of length at most k from 0 to i in the input graph. This is stated in the LC axiom:

$$\exists Z \leq \langle a, a\rangle, \psi_{LC}(a, E, Z), \text{ where}$$

$$\begin{aligned}\psi_{LC} =_{syn} & \forall i \leq a(Z(0,i) \leftrightarrow i = 0)\\ & \wedge \forall w, x \leq a(Z(w+1,x) \leftrightarrow (Z(w,x) \vee \exists y \leq a(Z(w,k) \wedge E(y,x))))\\ & \wedge \forall w, x < |Z|(w > a \vee x > a \supset \neg Z(w,a))\end{aligned}$$

Definition 8.2.1. The theory VNL is axiomatized by V^0 plus every instance of the LC axiom where E is replaced by a Σ_0^B formula.

Originally, the axiomatization of VNL did not include the substitution of Σ_0^B formulas for E in the LC axiom; however, this axiomatization is obviously equivalent since V^0 includes Σ_0^B-comp, and, for our purposes, it is useful.

The second theory was first studied in [18]. The idea was to use the descriptive complexity characterization of NL to define the theory. In [15], NL was described as the set of problems that can be defined by Σ_1^B-Krom-formulas.

Definition 8.2.2. A formula is a Σ_1^B-Krom-formula if it is

1. of the form
$$\exists Z_1 \ldots \exists Z_n \forall x_1 < t_1 \ldots \forall x_m < t_m \phi(\vec{a}, \vec{x}, \vec{B}, \vec{Z}),$$
where ϕ is a CNF formula in which no clause contains more that two Z-literals (formulas of the form $Z_i(t)$ or $\neg Z_i(t)$) or mentions $|Z|_i$, or

2. is an instance of a formula of the form 1 in which Σ_0^B formulas have replaced the free string variables.

Definition 8.2.3. The theory $V - Krom$ is V^0 plus comprehension for every Σ_1^B-Krom-formula.

This axiomatization is again slightly different from that in [18]: we include Σ_0^B-comp, and we substitute Σ_0^B-formulas for free string variables in Σ_1^B-Krom-formula. However, since the original $V - Krom$ can prove Σ_0^B-comp (see Corollary 5.2.2 [18]), the two axiomatizations are equivalent.

The third theory VKI has not been considered. The idea for this theory comes from the fact that V^i, for $i > 0$, can be axiomatized with Σ_i^B-ind instead of Σ_i^B-comp. In certain circumstances, the axiomatization with induction is more useful. For example, in the $G_i^* - V^i$ translation theorem, the induction axiom is used in place of the comprehension axiom to keep the quantifier complexity of the formulas down.

Definition 8.2.4. VKI is the theory axiomatized by V^0 plus Σ_1^B-Krom-ind.

8.2.2 Starred Theories

When we are doing propositional translations, there is a very strong restriction on the free variables that can appear in the cut formulas. As it turns out, this restriction can be mimicked in the theory by restricting the use of the axioms.

Let $\mathcal{T}$ be a theory axiomatized by $V^0 + \mathcal{A}$, where $\mathcal{A}$ is a set of formulas. Suppose $\phi(X) \in \mathcal{A}$. It is understood that $\mathcal{T}$ proves $\forall X \phi(X)$. The free variable X is implicitly universally quantified. We "shrink" $\mathcal{T}$ to a theory $\mathcal{T}^*$ where there is no implicit quantifier on ϕ. Instead, we will assume $\phi(X)$ holds for certain strings in the universe, not all.

Definition 8.2.5. Let $\mathcal{T}$ be a theory over the language $\mathcal{L}$ that is axiomatized by $V^0 + \mathcal{A}$. Then

1. $\mathcal{L}^*$ is the language $\mathcal{L}$ with new string constants C_i for $i \geq 0$,
2. $\mathcal{A}^*$ is the set of formulas obtained from $\mathcal{A}$ be replacing the free variable X_i by C_i, for $i \geq 0$, in every formula in $\mathcal{A}$, and
3. $\mathcal{T}^*$ is the theory over the language $\mathcal{L}^*$ axiomatized by $V^0 + \mathcal{A}^*$.

Note that, in the above definition, we assume there is a countable number of string variables, which we identify as $X_0, X_1, X_2, \ldots$. For the purpose of this definition there are no other string variables. We also assume that $\mathcal{A}$ is closed under term substitution. So, if $\phi(X_i)$ is in $\mathcal{A}$, then so is $\phi(X_j)$, for every j. This assumption is not necessary for the definition, but it is needed for some of the results later on.

This restriction on the axioms is meant to correspond to the restriction on the free variables in cut formulas. For example, if we have an anchored VKI^* proof of a formula $\phi(C_0)$, then the cut formulas are instances of axioms and there for the only free string is the constant C_0. If any of the other constants appear, then it can be replace by C_0 and we still get a valid proof. So if we translate this proof, the only free variables in the cut formulas are those that correspond to C_0.

At this point, it is worth reminding the reader that all of the theories $V-Krom$, VKI, and VNL have axiom schemes where we can replace string variables by a Σ_0^B formula. This is important when we are using the starred version of these theories.

8.2.3 Equivalence of the Theories

In this section, we want to prove that the three theories defined in the previous section are the same. We must do this carefully because there are certain properties of the proof that we will use later. We divide the proof into the separate parts.

Theorem 8.2.6. *$VKI \subseteq V-Krom$, and $VKI^* \subseteq V-Krom^*$.*

Proof. We need to prove Σ_1^B-Krom-ind from Σ_1^B-Krom-comp. In [11], it was shown that V^0 proves X-ind. To prove induction for a Σ_1^B-Krom-formula $\phi(i)$, just note that there is an X such that $X(i) \leftrightarrow \phi(i)$ for all $i \leq b$, by Σ_1^B-Krom-comp. So induction on i for $X(i)$ implies induction on i for $\phi(i)$.

Note that we only use one instance of the comprehension axiom scheme, and that the free variables are the same as the free variables in the induction scheme we are proving. Therefore the same construction works for the starred theories as well. □

The proof that $VNL \subseteq VKI$ uses ideas from the Immerman-Szelepcsenyi Theorem ([16, 40]). The idea is that we will count the number of nodes reachable from node 0 in a graph. Once this is done in VKI, we can determine whether or not a node is reachable by checking if it is one of the nodes we counted.

To do this in VKI, we construct a Σ_1^B-Krom-formula $\phi(c, a, E)$ that is true if the number of reachable nodes in the graph of E with a nodes is at most c. Then by finding the minimum value for c that satisfies $\phi(c, a, E)$, we can pick our which nodes are reachable.

We now want to define $\phi(c, a, E)$ in such a way that it is a Σ_1^B-Krom-formula. The idea is to formalize the following algorithm. Initialize a counter to 0. For each node in the graph, check if it is reachable from node 0. This check is done by looking at a set of nodes that contain 0 and is closed under reachability. If the current node is not in this set, then the node is not reachable and we move on to the next node. If the current node is in the set, then the node may be reachable, so we increment the counter and move on to the next node. Notice that it is possible that we count a node that is not reachable, but it is not possible that we miss a node that is reachable. This means the value of the counter at the end of the algorithm is at least the number of reachable nodes.

To formalize this algorithm, we use a counting array $Z(c, i, j)$. The c is the counter, and i is the current node. The third index is used to code the set described in the algorithm. So $Z^{[c,i]} = \{j : Z(c, i, j)\}$ is a set of nodes. This set will either be empty (meaning we do not use this value for the counter), or will contain 0 and be closed under reachability. This done as follows:

$$\rho_1 \equiv \forall c_0 \forall i \leq a \forall j_1 \forall j_2 [Z(c_0, i, j_1) \wedge E(j_1, j_2) \supset Z(c_0, i, j_2)]$$
$$[Z(c_0, i, j, u, v) \supset Z(c_0, i, j, 0, 0)]$$

Now that we have checked if a particular node is reachable, we now must make sure that we count properly. To find the number of nodes we have counted so far we look for the largest c such that $Z^{[c,i]}$ contains 0. To make this value easier to spot, we force $Z^{[c_0,i]}$ to contain 0 whenever $Z^{[c_0+1,i]}$ contains 0:

$$\rho_2 \equiv \forall c_0 \forall i [Z(c_0 + 1, i, 0) \supset Z(c_0, i, 0)].$$

Next we want to be sure that the counter does not decrease when we move on to the next node. This is done as follows:

$$\rho_3 \equiv \forall c_0 \forall i \forall [Z(c_0, i, 0) \supset Z(c_0, i + 1, 0)]$$

If we find that the current node (i) is possibly reachable, we want to be sure that we increment the counter:

$$\rho_4 \equiv \forall c_0 \forall i [Z(c_0, i, i) \supset Z(c_0 + 1, i + 1, 0)]$$

We need to deal with the beginning and end. To initialize the algorithm, we want to make sure that $Z^{[0,0]}$ contains 0. As well, at the end, we want to make sure that we have counted c nodes. Note that the last node is $a - 1$. So to check the total number of nodes counted we check the value after we have checked this node. To accomplish this, we include

$$\rho_5 \equiv Z(0, 0, 0) \wedge Z(c, a, 0) \wedge \neg Z(c + 1, a, 0).$$

Putting all of this together we get

$$\phi(c, a, E) \equiv \exists Z, \rho_1 \wedge \rho_2 \wedge \rho_3 \wedge \rho_4 \wedge \rho_5$$

Note that we have not included the bounds for the quantifiers, but we because the graphs has a nodes there is an appropriate bound for every quantifier.

In order to count the exact number of reachable nodes, we must find the minimum value for c such that $\phi(c, a, E)$ is true. This follows by induction on $\neg\phi(c, a, E)$. For $c = 0$, $\neg\phi(0, a, E)$ is true, since the node 0 is always reachable. As well, $\phi(a, a, E)$ is true since there are only a nodes in the graph and we can claim that every node is reachable. So by induction with $\neg\phi(c, a, E)$ there is a value for c_0 such that $\phi(c_0, a, E)$ is false and $\phi(c_0+1, a, E)$ is true. For this to work we must be able to prove induction on the negation of Σ_1^B-Krom-formulas.

Lemma 8.2.7. *Let $\phi(i)$ be a Σ_1^B-Krom-formula. Then VKI proves*

$$\neg\phi(0) \wedge \forall i < b[\neg\phi(i) \supset \neg\phi(i + 1)] \supset \neg\phi(b).$$

Proof. This can be done the standard way. That is, induction on i for $\neg\phi(i)$ is equivalent to induction on i for $\phi(b-i)$, which is a Σ_1^B-Krom-formula. □

Theorem 8.2.8. *$VNL \subseteq VKI$, and $VNL^* \subseteq VKI^*$*

Proof. To prove the theorem, we must prove the LC axiom in VKI. So consider an a and E as in the axiom. The LC axiom says there exists a string that is the set of reachable nodes in the layered graph E', defined as

$$E'(\langle d, i\rangle, \langle d', j\rangle) \leftrightarrow (E(i,j) \vee i = j) \wedge d' = d + 1.$$

Let c_0 be the minimum value such that $\phi(c_0, n, E')$ is true, where n is the number of nodes in the graph of E'. Let Z be a witness for ϕ. Then define

$$Y(d,i) \leftrightarrow \exists c\, [Z(c, \langle d,i\rangle, 0) \wedge \neg Z(c+1, \langle d,i\rangle, 0) \wedge Z(c, \langle d,i\rangle, \langle d,i\rangle)].$$

The idea is that we are looking at the point where we checked if $\langle d, i\rangle$ is reachable. If the algorithm determined it is reachable, then we include it. If not, then we do not include it.

We now claim that Y is a witness for the LC axiom. If Y is not a witness, it is because we counted a node that is not reachable. We can then change Z to not count this node and show that $\phi(c_0 - 1, n, E')$ is true. However, this contradicts our choice of c_0.

It is possible to show this more formally, but we leave it to the reader.

For the starred theories, observe that the only time the induction is used is on the formula $\neg\phi(c_0, n, E')$. The only free string variable is E', which can be defined by a Σ_0^B formula whose free variables are the free variables of the instance of the LC axiom we are proving. In the case of the starred theories, E' can be defined in terms of the string constants and no other strings. Therefore, this induction axiom can be carried out in

VKI^*. $\square$

For the final inclusion, we show that $V-Krom \subseteq VNL$. This involves proving the Σ_1^B-Krom-comp axiom scheme in VNL. To do this, we reduce the evaluation of a Σ_1^B-Krom-formula to a question of reachability in undirected graphs. This was done in Theorem 5.5.1 [18], but there was a slight error, so we will give a new proof that fixes this mistake.

Lemma 8.2.9. *Let $c = \langle\langle 0,2\rangle, \langle 0,2\rangle\rangle$. For every Σ_1^B-Krom-formula $\phi(\vec{a}, \vec{A})$, there exists a Σ_0^B-formula $\phi'(i,j,\vec{a},\vec{A})$ such that*

$$VNL \vdash \phi(\vec{a}, \vec{A}) \leftrightarrow \exists Q[Cond(t, Q, \phi') \wedge \neg Q(0,c)],$$

where

$$Cond(b, Q, \phi') =_{syn} \forall i,j,k \leq b[Q(i,i) \wedge (Q(i,j) \wedge \phi'(j,k,\vec{a},\vec{A}) \supset Q(i,k))].$$

Proof. Let

$$\phi \equiv \exists Z \forall x(\psi(x,Z)),$$

where ψ is the CNF formula in which, for all i, clause i is of the form

1. $Z(t_1^i) \vee Z(t_2^i) \vee \psi_i$,
2. $Z(t_1^i) \vee \neg Z(t_2^i) \vee \psi_i$, or
3. $\neg Z(t_1^i) \vee \neg Z(t_2^i) \vee \psi_i$.

Note that we are assuming there is only one existentially quantified string variable and one universally quantified number variable. The case where there are more is essentially the same.

The reduction involves the construction of the implication graph for ϕ. The nodes in the graph are the literals that correspond to the bits of Z. So there is a node for $Z(i)$

and for $\neg Z(i)$. There is a (directed) edge from one literal to another if there is a clause that says the first literal implies the other.

As before, literals will be coded as pairs $\langle i, v\rangle$. The pair corresponds to $Z(i)$ when $v = 0$ and $\neg Z(i)$ when $v = 1$. We use l to refer to a literal, and $\bar{l}$ will be the negation of l. As well, $Positive(l)$ and $Negative(l)$ are predicates that are true when l is positive and negative, respectively. Then the clauses above can be viewed at sets. For example, if clause i has form 1, then

$$C(i,x) = \begin{cases} \{\langle t_1^i(x), 0\rangle, \langle t_2^i(x), 0\rangle\} & \text{if } \neg\psi_i(x) \\ \{\top\} & \text{otherwise} \end{cases}$$

Since ψ is a fixed formula, $C(i,x)$ can be defined by a Σ_0^B formula.

The set of edges for the implication graph can be defined as follows:

$$E(l_1, l_2) \leftrightarrow \exists i \exists x(\bar{l}_1 \in C(i,x) \wedge l_2 \in C(i,x))$$

Claim 8.2.10. VNL proves that ϕ is false if and only if there exists an l such that there is a path in E from l to $\bar{l}$ and then back to l.

Assuming this is true, we finish the proof by defining a graph ϕ' where there is a path from 0 to $c = \langle\langle 0, 2\rangle, \langle 0, 2\rangle\rangle$ if and only if ϕ is false. Each node in this graph is a pair of literals. To start the path, there is an edge from $\langle 0, 0\rangle$ to $\langle l_1, l_1\rangle$ for any positive literal l_1. This corresponds to guessing a literal to check. From this point on, we are looking for a path from l_1 to $\bar{l}_1$. The second node in the pair is used to search for this path. If this path is found (that is we reach $\langle l_1, \bar{l}_1\rangle$, then we go to $\langle \bar{l}_1, \bar{l}_1\rangle$ to start looking for a path back to l_1. If this path is found we go node c. This is formalized as follows:

$$\begin{aligned}\phi'(\langle l_1, l_1'\rangle, \langle l_2, l_2'\rangle) \equiv \quad & (l_1 = 0 \wedge l_1' = 0 \wedge l_2 = l_2' \wedge Positive(l_2)) \\ & \vee (l_1 = l_2 \wedge E(l_1', l_2')) \\ & \vee (l_1 = \bar{l}_1' \wedge Positive(l_1) \wedge l_2 = \bar{l}_1 \wedge l_2' = \bar{l}_1) \\ & \vee (l_1 = \bar{l}_1' \wedge Negative(l_1) \wedge \langle l_2, l_2'\rangle = c)\end{aligned}$$

It is a simple task to prove that the following claim is true.

Claim 8.2.11. V^0 proves that node c is reachable from node 0 in ϕ' if and only if there is a literal l such that there is a path from l to $\bar{l}$ and back to l in E.

The two claims together imply the theorem. □

To complete the proof of the theorem above, we still need to prove the first claim.

Proof of Claim 8.2.10. For one direction, let $l = \langle i, 0\rangle$ be a literal such that there is a path from l to $\bar{l}$ and then back to l in E. Let Z be a possible witness for ϕ. If $Z(i)$ is true, then, by induction on the length of the path from l to $\bar{l}$, there is an edge from a literal l_1 to l_2 such that l_1 is true and l_2 is false. This corresponds to a clause in ϕ that is false proving that Z is not a witness. Similarly, if $Z(i)$ is false, then, using the path from $\bar{l}$ to l, Z is not a witness for ϕ.

For the other direction, assume that no such l exists. Let $E^*(i,j)$ be the string that represents the transitive closure of E. VNL proves that E^* exists using an instance of the LC axiom with a string that is Σ_0^B definable in terms of E. The first observation to make is that for every literal l_1 and l_2,

$$E^*(l_1, l_2) \leftrightarrow E^*(\bar{l}_2, \bar{l}_1).$$

This can be easily proved from the fact that

$$E(l_1, l_2) \leftrightarrow E(\bar{l}_2, \bar{l}_1).$$

Now we need to define a string Z that is a witness for ϕ. To do this, we will define a set A of literals. Then $Z(i)$ is true if $\langle i, 0\rangle$ is in A. That is A will be the set of true literals.

To start the assignment, notice that if there is a path from l to $\bar{l}$, then $\bar{l}$ is forced to be true. We can identify literal that are forced to be true as follows:

$$Forced(l) \equiv E^*(\bar{l}, l).$$

Then an unforced literal is one where neither it nor it negation are forced:

$$Unforced(l) \equiv \neg Forced(l) \wedge \neg Forced(\bar{l}).$$

This gives us the assignment for certain variables, but there are still other variables that we must assign values to. For a literal l, we find the smallest literal l' that, if we assign false to l', then l is forced to have a value. Then we assign l that value. This can be formalized as follows:

$$\begin{aligned} l \in A \leftrightarrow Forced(l) \vee \exists l'[&\neg Forced(\bar{l}) \\ &\wedge l' = \min_{l'}(Unforced(l') \wedge (E^*(l, l') \vee E^*(\bar{l}', l))) \\ &\wedge E^*(\bar{l}', l)] \end{aligned}$$

First we want to show that A is a valid assignment. That is, exactly one of l and $\bar{l}$ is in A. For a contradiction, assume that $l \in A$ and $\bar{l} \in A$. Let

$$l_1 = \min_{l'}(Unforced(l') \wedge (E^*(l, l') \vee E^*(\bar{l}', l)))$$
$$l_2 = \min_{l'}(Unforced(l') \wedge (E^*(\bar{l}, l') \vee E^*(\bar{l}', \bar{l})))$$

Then we can show that $l_1 = l_2$ because a literal that forces l to be true forces $\bar{l}$ to be false (recall $E^*(l, l_1) \leftrightarrow E^*(\bar{l}_1, \bar{l})$). Since both l and $\bar{l}$ are in A, we know that $E^*(\bar{l}_1, l)$ and $E^*(\bar{l}_1, \bar{l})$. The latter implies $E^*(l, l_1)$. However, this means that l_1 is forced to be true ($E^*(\bar{l}_1, l)$ and $E^*(l, l_1)$), contradicting the choice of l_1. Similarly, if neither l nor $\bar{l}$ are in A, there is a contradiction.

Now to show that this assignment indeed witnesses ϕ. Suppose, for the sake of contradiction, it does not. Let $C(i, x)$ be the clause that is not satisfied. That is, $C(i, x) = \{l_1, l_2\}$ where $l_1 \notin A$ and $l_2 \notin A$. Let

$$l'_1 = \min_{l'}(\neg Forced(l') \wedge (E^*(\bar{l}_1, l') \vee E^*(\bar{l}', \bar{l}_1)))$$
$$l'_2 = \min_{l'}(\neg Forced(l') \wedge (E^*(\bar{l}_2, l') \vee E^*(\bar{l}', \bar{l}_2)))$$

Since $\bar{l}_1$ is true, we have that l'_1 implies $\bar{l}_1$. Since l_1 and l_2 are in the same clause $\bar{l}_1$ implies l_2. Together this means that l'_1 forces l_2 to be false. So by the choice of l'_2, we have that $l'_1 \geq l'_2$. Using a similar argument, $l'_2 \geq l'_1$. This means that $l'_1 = l'_2$. However, as before, we are able to show that l'_1 is forced, which contradicts the choice of l'_1. $\square$

Lemma 8.2.12. *Let $c = \langle\langle 0, 2\rangle, \langle 0, 2\rangle\rangle$. For every Σ_1^B-Krom-formula $\phi(\vec{a}, \vec{C})$ with no free string variable (only the string constants $\vec{C}$), there exists a Σ_0^B-formula $\phi'(i, j, \vec{a}, \vec{C})$ such that*

$$VNL^* \vdash \phi(\vec{a}, \vec{C}) \leftrightarrow \exists Q[Cond(t, Q, \phi') \wedge \neg Q(0, c)],$$

where

$$Cond(b, Q, \phi') =_{syn} \forall i, j, k \leq b[Q(i, i) \wedge (Q(i, j) \wedge \phi'(j, k, \vec{a}, \vec{C}) \supset Q(i, k))].$$

Proof. The proof is the same as the proof of the previous lemma. The only time the LC axiom is used in that proof is in the proof of claim 8.2.10, but the free string variables can be defined by Σ_0^B formulas with no free string variables. □

Theorem 8.2.13. *$V - Krom \subseteq VNL$, and $V - Krom^* \subseteq VNL^*$.*

Proof. Because of Lemma 8.2.9, we only need to prove comprehension for Σ_1^B-Krom-formulas of the form

$$\psi(x) =_{syn} \exists Q[Cond(t, Q, \phi(x)) \wedge \neg Q(0, c)],$$

where the term t may contain x and $\phi(x)$ is a Σ_0^B-formula. Note that ϕ contains two free variables other than x which are used in $Cond$ for the edge relation. These two variables will be referred to as i and j. The formula $\psi(x)$ says that there is no path from 0 to c in the graph with nodes $\{0, \ldots, t\}$ and edge relation $\phi(i, j, x)$. We need to prove

$$\exists Y \leq b \forall x < b[Y(x) \leftrightarrow \psi(x)]$$

in VNL.

We begin by defining a graph $E(i, j)$ by

$$\begin{aligned} E(\langle i_1, j_1\rangle, \langle i_2, j_2\rangle) \leftrightarrow &(i_1 = 0 \wedge j_1 = 0 \wedge j_2 = 0 \wedge i_2 \leq t(b)) \\ &\wedge(i_1 = i_2 \wedge j_2 \leq t(i_1) \wedge \phi(j_1, j_2, i_1)). \end{aligned}$$

In this formula, E is a graph with a copy of the graph of ϕ for the different values of x. Node $\langle i, j\rangle$ belongs to the copy of the graph of ϕ when $x = i$. Let Z be the string that exists by the LC axiom with E. Then we can define Y as

$$Y(x) \leftrightarrow Z(\langle b, t(b)\rangle, \langle x, c\rangle).$$

The idea is that the node $\langle x, c\rangle$ is reachable from 0 in E if and only if there is a path from 0 to c in the graph of $\phi(i, j, x)$.

For the starred theories, the proof is essentially the same. We use Lemma 8.2.12 in place of Lemma 8.2.9. The only other use of the LC axiom is with E, which is Σ_0^B definable in terms of the free variables of ψ. □

Immerman-Szelepcsenyi Theorem in VKI^*

We still need to establish the connection between VKI and VKI^*. An important step in this connection will be a proof that FNL is closed under composition. In order to prove this, we must be able to show that NL is closed under complement. This is the Immerman-Szelepcsenyi theorem that first appeared in [16, 40]. Later, Kolokolova showed how this theorem could be formalized in $V-Krom$ [18]. We should like to do something similar to what she did; however, we want to formalize it in the starred theory VKI^*. In a model of VKI^*, there are certain strings that cannot be used in the induction axiom. Because of this, we cannot prove the theorem for all strings, but we will prove it for certain strings.

Theorem 8.2.14. *Let $\mathcal{M}$ be a model of VKI^*, and let E be a string such that*

$$\forall i, j \leq a E(i, j) \leftrightarrow \phi(i, j),$$

where ϕ is a Σ_0^B formula with no free string variables but may contain the string constants. Then there exists a string E' and a number c_0 such that

$$\forall l, i, j < a' E'(i, j)' \leftrightarrow \phi'(i, j),$$

where ϕ' has no free string variables, and there is a path from 0 *to* n *in the graph of* E *if and only if there is a path from* 0 *to* $\langle n, c_0, a+1, 0\rangle$ *in the graph of* E'.

Proof. The proof of Theorem 8.2.8, and the preceding text, explains how to find a c_0 that is the number of nodes reachable from 0 in the graph of $\phi(i,j)$. This was done in VKI, but the same construction works in VKI^* when ϕ does not have any free string variables (only the string constants). To check if node n is reachable, we simply need to find c_0 nodes other than n that are reachable, and we want to do this for all n at the same time.

A node in ϕ' will be a tuple $\langle n, c, i_1, i_2\rangle$. The n will refer to the nodes we want to avoid, the c is the number of nodes we have counted so far, i_1 is the current node we are checking, and i_2 is the current node in the path from 0 to i_1. So now to describe ϕ'.

Starting at node 0 the first step is to pick the node we want to avoid:

$$\alpha_1(\langle n, c, i_1, i_2\rangle, \langle n', c', i_1', i_2'\rangle) \equiv (n = c = i_2 = i_2 = 0 \wedge c' = i_1' = i_2' = 0).$$

For each n, c and i_1, we will try to guess a path in ϕ from 0 to i_1:

$$\alpha_2(\langle n, c, i_1, i_2\rangle, \langle n', c', i_1', i_2'\rangle) \equiv (n = n' \wedge c = c' \wedge i_1 = i_1' \wedge \phi(i_2, i_2')).$$

If a path is found and the current node is not n, we want to move on to the next node and increase the counter:

$$\alpha_3(\langle n, c, i_1, i_2\rangle, \langle n', c', i_1', i_2'\rangle) \equiv (i_2 = i_1 \neq n \wedge n = n' \wedge c+1 = c' \wedge i_1 + 1 = i_1' \wedge i_2' = 0).$$

At any time, we want to be able to move on to the next node without increasing the counter:

$$\alpha_4(\langle n, c, i_1, i_2\rangle, \langle n', c', i_1', i_2'\rangle) \equiv (n = n' \wedge c = c' \wedge i_1 + 1 = i_1' \wedge i_2' = 0).$$

Then we let

$$\phi'(\langle n, c, i_1, i_2\rangle, \langle n', c', i'_1, i'_2\rangle) \equiv \alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge \alpha_4.$$

In this graph, there is a path from 0 to $\langle n, c_0, a+1, 0\rangle$ if and only if node n is not reachable in the graph of ϕ. If this was not true, then, in a similar manner to the proof of Theorem 8.2.8, we could show that c_0 is not the number of reachable nodes as it is supposed to be. □

8.2.4 Connection Between VNL and VNL^*

We will now show why the starred theories can be useful. We do this by showing the connection between VNL and VNL^*.

Theorem 8.2.15. *Let $\phi(X_0, \ldots, X_n)$ be a Σ_1^B formula. Then VNL proves $\phi(X_0, \ldots, X_n)$ if and only if VNL^* proves $\phi(C_0, \ldots, C_n)$.*

Proof. First to prove the $\Longleftarrow$ direction. This is the easy direction. We prove the contrapositive. Suppose VNL does not prove $\phi(X_0, \ldots, X_n)$. Then there exists a model $\mathcal{M}$ of VNL such that $\phi(X_0, \ldots, X_n)$ is false. Expand $\mathcal{M}$ to a model of VNL^* by interpreting the constants C_i as the string assigned to X_i in an assignment that falsifies ϕ. Call the expanded model $\mathcal{M}^*$. Then $\mathcal{M}^*$ is a model of VNL^* that shows that VNL^* does not prove $\phi(C_0, \ldots, C_n)$.

The $\Longrightarrow$ direction is more difficult. Once again we will prove the contrapositive. Suppose VNL^* does not prove $\phi(C_0, \ldots, C_n)$. Then let $\mathcal{M}^*$ be a model of VNL^* where $\phi(C_0, \ldots, C_n)$ is false. Let M^* be the string universe of $\mathcal{M}^*$. We will construct a model of VNL with the same number universe as $\mathcal{M}^*$ and whose string universe M is such that $C_i \in M$, for all $i \geq 0$, and $M \subseteq M^*$. Then, if we assign X_i the string C_i, the formula $\phi(X_0, \ldots, X_n)$ is false in $\mathcal{M}$. This is because ϕ is a Σ_1^B formula: If there are strings that witness ϕ in M, then there are strings that witness ϕ in M^*, which, by assumption, is not true. So now to actually construct M.

The set of strings M is defined in three stages. First

$$M_1 = \left\{ Y : \forall i < m(Y(i) \leftrightarrow \phi(i, \vec{n}, \vec{S}),\text{ where } \phi \in \Sigma_0^B \text{ and } m, n \in M^* \text{ and } \vec{S} \in \vec{C} \right\}$$

This set corresponds to taking the closure of the constants under Σ_0^B-COMP. That is, M_1 is the set of all strings that can be constructed from the constants $C_0, C_1, \ldots$ using Σ_0^B-COMP.

The second step is to close M_1 under the LC axiom. So,

$$M_2 = \{Y : \psi_{LC}(s, E, Y) \text{ where } s \in M^* \text{ and } E \in M_1 \}.$$

The final step is to close M_2 under Σ_0^B-COMP.

$$M_3 = \left\{ Y : \forall i < m(Y(i) \leftrightarrow \phi(i, \vec{n}, \vec{S}),\text{ where } \phi \in \Sigma_0^B \text{ and } m, n \in M^* \text{ and } \vec{S} \in M_2 \right\}$$

Then we let $M = M_3$.

The model $\mathcal{M}$ is obtained by restricting $\mathcal{M}^*$ to the strings in M, and this is a model of VNL. The Σ_0^B-comp axioms also hold since M_3 is obtained by taking the closure under Σ_0^B-COMP. By Lemma 8.2.16 below, $\mathcal{M}$ satisfies the LC axiom. □

Lemma 8.2.16. *Let $\mathcal{M}$ be the model constructed in the theorem above with string universe M. Then, $\forall E \forall a \exists Z \psi_{LC}(a, E, Z)$ is true.*

Proof. Fix E and a. First suppose $E \in M_1$. Since E is defined by a Σ_0^B formula with no free string variables, a witness for the LC axiom exists in M^*, and it would be included in M by the definition of M_2.

Now suppose $E \in M - M_1$. Then

$$E(i, j) \leftrightarrow \phi(i, j, \vec{s}, \vec{S})$$

for some Σ_0^B formula ϕ and $\vec{S} \in M_2$. Without loss of generality, assume that $\vec{s}$ is empty and that $\vec{S}$ is a single variable $S \in M_2$.

This construction proceeds in a similar manner to the corresponding theorem in [35]. In that paper, the author showed how to compose two branching programs. Here we show how to compose two *labeled graphs.*

Definition 8.2.17 (labeled graph)**.** A *labeled graph* is a complete directed graph where each edge is labeled with a Σ_0^B formula. The formula is intended to tell you whether or not the corresponding edge is in the graph.

The first graph we start with is G_0 where the edge (u, v) is labeled with $\phi(u, v, S)$. Using DeMorgan's Law, we can assume that for each label negations appear only on atomic formulas. As with the construction in [35], we now want to change G_0 one label at a time so that every label is an atomic formula or the negation of an atomic formula. We describe how this is done in one case. Other cases are left to the reader.

Suppose the graph G_i has an edge (u, v) labeled with $\forall a \leq b\alpha(a)$. Then G_{i+1} is constructed by adding b new nodes call them $\{u, v\}_1, \ldots, \{u, v\}_b$. Then $(u, \{u, v\}_1)$ is labeled with $\alpha(0)$, $(\{u, v\}_i, \{u, v\}_{i+1})$ is labeled with $\alpha(i)$ for $i < b$, and $(\{u, v\}_b, v)$ is labeled with $\alpha(b)$. The label for (u, v) becomes $\perp$, and all other new edges are labeled with $\perp$. Notice that there is a path between u and v if and only if $\forall a \leq b\alpha(a)$. As well, given the reachability matrix for G_{i+1} we can obtain the reachability matrix for G_i using Σ_0^B-COMP.

When this is done we obtain a labeled graph G where every label is an atomic formula or the negation of one. We are interested in labels of the form $S(t_1, t_2)$ or $\neg S(t_1, t_2)$ for some terms t_1, t_2. These are the labels we must change in order to define E'. Since $S \in M_2$, there exists an $E_1 \in M_1$ such that $\psi_{LC}(a_1, E_1, S)$. By Theorem 8.2.14, there exists an $E_2 \in M_1$ such that, if $\psi_{LC}(a_2, E_2, S')$, then $S(m, n) \leftrightarrow \neg S'(0, t(m, n))$ for an appropriate term t (note that we are ignoring the constant c_0, but it is there implicitly). A new graph G' is obtained from G by making two changes: (1) every node labeled with $S(t_1, t_2)$ is replaced with a copy of the graph of E_1 and (2) every node labeled with $\neg S(t_1, t_2)$ is replaced with a copy of the graph of E_2. This graph can be described by a Σ_0^B formula using only the string variables E_1 and E_2, which are both in M_1. Then E' is the string defined by this formula.

We know that $E' \in M_1$, and there exists $Z' \in M_2$ such that $\psi_{LC}(a', E', Z')$. Then, from the construction of E', we can construct a Σ_0^B formula that extracts the list of reachable nodes in E using Z'. This is the witness Z that we want. Since Z' is in M_2, Z is in M_3. □

8.3 Propositional Translations of VKI

Theorem 8.3.1. *Suppose $VKI \vdash \exists Z\phi(\vec{x}, \vec{X}, Z)$. Then there exist polynomial size GNL^* proofs of $||\exists Z\phi(\vec{x}, \vec{X}, Z)||[\vec{n}; \vec{m}]$.*

Proof. By Theorems 8.2.6, 8.2.8, 8.2.13, and 8.2.15, $VKI^* \vdash \exists Z\phi(\vec{x}, \vec{C}, Z)$. Let π be an anchored VKI^* proof of this formula. Set $\vec{n}$ and $\vec{m}$. We show, by induction on the depth of π, how to translate π into a GNL^* proof π' where the size of the proof is polynomial in $\vec{n}$ and $\vec{m}$. Note that we can assume that, if the string constant C_i appears in the proof, it appears in the final sequent. If not, we can simply replace C_i by a different constant that does. This can be done since the axioms of VKI^* are closed under substitution of one constant for another.

The base case is easy since the axioms used in the proofs are all axioms of V^0. The inductive step is divided into cases: one case for each rule. Most of the rules are handled the same way they are in other proofs of this type. See Chapter 7 of [11] for an example. The only two that require comment are the induction rule and the cut rule.

Suppose the last rule of inference is

$$\frac{\Gamma, \phi(i) \rightarrow \phi(i+1), \Delta}{\Gamma, \phi(0) \rightarrow \phi(t), \Delta}$$

By induction, there are GNL^* proofs π_i' of $||\Gamma, \phi(i) \to \phi(i+1), \Delta||$. Let $n = val(t)$. The proofs $\pi_0', \ldots, \pi_n'$ are combined by repeated cutting to get π'. This is similar to the proof of the V^0 translation theorem in [11]. The formula $\phi(i)$ is a Σ_1^B-Krom-formula with no free string variables. These formulas translate into $\Sigma Krom$ formulas where the free variables correspond to the constants in the language of VKI^*, which appear in the final sequent.

The other case to consider is cut. Because π is anchored, the cut formula must be an axiom of V^0 or a descendant of a principal formula used in an induction inference. In the first case, standard methods can be used to make the cut formula Σ_0^q (see [10] for details). In the second case, the cut formula must a Σ_1^B-Krom-formula with no free string variables. This formula translates to a $\Sigma Krom$ formula where the only free variables correspond to the constants, so the corresponding propositional formula can be cut to form π'. $\square$

8.4 Reflection Principles

As with other proof systems, we would like to be able to prove that the proof system is sound in the theory. This proof is essentially the same as the proof that VL proves the reflection principle for GL^*. See Section 7.3.2. The only thing that will change from that proof is that instead of witnessing $\Sigma CNF(2)$ formulas we have to witness $\Sigma KROM$ formulas.

To witness $\Sigma KROM$ formulas, we formalize the algorithm used to witness Σ_1^B-Krom-formula (Theorem 8.2.9). Since the method is the exact same, we will not repeat the construction, only state the result. As well, we will not formally define the theory $\overline{VNL}$. It is a universal theory with a function symbol for each function in FNL. This language is referred to as $\mathcal{L}_{FNL}$. It is similar to $\overline{VL}$ (Section 7.1). For a complete exposition, see Chapter 9 of [11].

Lemma 8.4.1. *Let $\exists\vec{z}A(\vec{x}, \vec{z})$ be a $\Sigma KROM$ formula, where A is the quantifier-free* *portion of the formula. Then there is a $\mathcal{L}_{FNL}$ function F such that $\overline{VNL}$ proves*

$$X \cup F(X, A) \models_0 A \leftrightarrow X \models_1 \exists \vec{z} A(\vec{x}, \vec{z}).$$

Then using the same proof as Theorem 7.3.8, we can prove the following.

Theorem 8.4.2. *$VNL \vdash \Sigma_1^q$-RFN(GNL^*).*

Chapter 9

A Proof System for TV^i

The point of this section is to understand how to restrict the strength of a proof system when it is proving formulas with high quantifier complexity. As we saw in Sections 6.1 and 5.1, we cannot distinguish G_i^* and G_{i+1}^* by looking at the witnessing problem or the reflection principles for complex formulas. In order to better understand what is happening, we define a family of proof systems that has the same strength as TV^i, even for complex formulas.

Comparing the KPT witnessing theorem for TV^i and for G_{i+1}^*, we notice that for TV^i the "student-teacher" game has a constant number of rounds before the student is guaranteed to win; however, for G_{i+1}^* there is potentially a polynomial number of rounds. This gives an indication that, if we can somehow restrict G_{i+1}^* to reduce the number of rounds to a constant, this may give us a proof system that truly corresponds to TV^i.

Definition 9.0.3 (Repeatition of Definition 4.1.1)**.** For $i \geq 0$, the proof system GPV_{i+1}^* is G_{i+1}^* in which cut formulas are restricted to Σ_i^q formulas or formulas of the form $\exists x[x \leftrightarrow A]$, where A is a Σ_i^q formula that does not mention x. The proof system GPV^* will refer to GPV_1^*.

Definition 9.0.4. A $GPV_i^*(c)$ proof is a GPV_i^* proof where on any branch there are at most c $\forall$-right inferences where the principal formula ceases to be Σ_j^q for $j \geq i$.

This restriction comes from a careful examination of the Herbrand Theorem for G_i^* (Theorem 3.2.9). The idea is to find the places in that construction where new groups of quantifiers are introduced, and to limit those instances to a constant number. The reason we restrict GPV_i^* instead of G_i^* is that we would not be able to prove the reflection principle. This will become evident in the proof, but we rely on the fact that if we are cutting a Σ_i^q formula in a GPV_i^* proof, then we know the cut formula $\exists z[z \leftrightarrow A]$ is true. That is not the case with G_i^*.

Another way of viewing the restriction is to limit the number of groups of quantifiers introduced along any branch to a constant. Recall the definition of a group as given in the proof of Theorem 5.1.2. Two universal quantifiers are in the same group if they are in the scope of the same existential quantifiers.

We claim that the family of proof systems $\{GPV_i^*(c) : c \geq 0\}$ corresponds to TV^i in the following sense: (1) TV^i proves Σ_j^q-RFN$(GPV_i^*(c))$ for all $j, c \geq 0$, and (2) for every theorem ϕ of TV^i, there exists a c such that ϕ can be translated into a family of valid QPC formulas that have polynomial-size $GPV_i^*(c)$ proofs.

9.0.1 Propositional Translations

Before we prove the translation theorem, we want to give an alternate axiomatization of TV^i that is useful for the translation. As was mentioned in Section 2.2, TV^i is defined by V^0 plus Σ_i^B-string-ind, repeated here to remind the reader:

$$[\phi(\emptyset) \wedge \forall X[\phi(X) \supset \phi(S(X))]] \supset \phi(Y).$$

For the translation, it will be easier to axiomatize TV^i using V^0 plus the Σ_i^B-bit-recursion:

$$\exists X \forall i < y(X(i) \leftrightarrow \phi(i, X^{<i}))$$

where $\phi(i, Y)$ is a Σ_i^B formula that does not mention X, and $Y = X^{<i}$ means $|Y| \leq i$ and $Y(j) \leftrightarrow X(j)$ for $j < i$. As with Σ_i^B-string-ind, we should think of the axiom as an $\mathcal{L}_A^2$ formula where the chop function ($X^{<i}$) is replaced by its Σ_0^B definition.

Theorem 9.0.5 (Theorem 3.12 [8]). *$TV^i = V^0 + \Sigma_i^B$-bit-recursion.*

Note that the proof of this theorem is based on a similar proof in the single-sorted setting (Theorem 8 [2]).

Lemma 9.0.6. *There are polynomial-size $GPV_{i+1}^*(0)$ proofs of the sequent*

$$\rightarrow \exists z[z \leftrightarrow A],$$

where A is a Σ_i^q formula that does not mention z.

Proof. Begin by proving

$$A \rightarrow \top \leftrightarrow A$$

and

$$\rightarrow \bot \leftrightarrow A, A.$$

Proofs of these sequents can be shown to exist by structural induction on A. By applying $\exists$-right to these sequents, we get proofs of

$$A \rightarrow \exists z[z \leftrightarrow A]$$

and

$$\rightarrow \exists z[z \leftrightarrow A], A.$$

Then we get the sequent we want by cutting A. □

Theorem 9.0.7 (TV^i-$GPV_{i+1}^*(c)$ Translation Theorem). *If TV^i proves $\phi(\vec{x}, \vec{X})$, then there exists a c such that there are polynomial-size $GPV_{i+1}^*(c)$ proofs of $||\phi(\vec{x}, \vec{X})||[\vec{m}; \vec{n}]$.*

Proof. Take a TV^i proof π of ϕ. Let c be the number of $\forall$-right inferences in π where the principal formula ceases to be Σ_j^B, for $j > i$. The proof is similar to the proofs of other translation theorems. The main difference to how we handle the Σ_0^B-bit-recursion axiom and the need to keep track of the number of special inferences in the propositional proof. To take care of these special inferences, we need to state the induction hypothesis carefully.

For each sequent S in π, let c_S be the number of $\forall$-right rules in the subproof ending with S where the principal formula ceases to be Σ_j^B, for $j > i$. We prove by induction on the depth of S that there is are polynomial-size $GPV_i^*(c_S)$ proofs of the translation of S. There is a separate case for each rule of inference. Most rules can be simulated using the corresponding rule in the propositional proof system, so we only mention the few rules that need extra work.

Axioms: It is known how to prove the translation of the 2BASIC axioms. It is a simple exercise to prove the translation of Σ_0^B-bit-recursion axiom using Lemma 9.0.6.

Cut: Since the proof is anchored, the cut formula in the TV^i proof is an axiom. If it is one of the 2BASIC axioms, then the rule is simulated by cutting the translation of the cut formula.

Otherwise, the cut formula is an instance of the Σ_i^B-bit-recursion axiom. Note that Σ_0^B-COMP is a special instance of the recursion scheme where ϕ does not mention $X^{<i}$. The inference is of the form

$$\frac{\exists X \le t \forall i < t \psi(i, X), \Gamma \to \Delta \qquad \to \exists X \le t \forall i < t \psi(i, X)}{\Gamma \to \Delta}$$

where

$$\psi(i, X) \equiv X(i) \leftrightarrow \phi(i, X^{<i}).$$

Let $A_{m,n} \equiv ||\psi(i, X)||[m; n]$ and $B_n \equiv \bigwedge_{m=0}^{val(t)} A_{m,n}$. Note that B_n is $||\forall i < t\psi(i, X)||[; n]$.

Informally, the formula $A_{m,n}$ is saying the size of X is n and the mth bit has the correct value (given the bits 0 to $m - 1$). The formula B_n ($n \le val(t)$) says that the size of X is n and all of the bits have the correct value.

Now consider the formula $B_{val(t)+1}$. Because of the chop function the formulas $A_{m,val(t)+1}$ do not reference the leading 1 bit that is assumed to exist in X. In fact, the formula $B_{val(t)+1}$ says that the variables $p_0^X, \ldots, p_{val(t)-1}^X$ satisfy the recursion. Then, if i is the largest value such that p_i^X is true, B_{i+1} is true since $i+1$ would be the size of the string representing the first $val(t)$ bits. This means we are able to show the following:

$$B_{val(t)+1} \supset \bigvee_{n=0}^{val(t)} B_n. \tag{9.0.1}$$

Now to continue the proof of the cut case. By induction with the upper-left sequent in the inference given above, we get a polynomial-size $GPV_{i+1}(c_S)$ proof of

$$\exists x_0 \ldots \exists x_{val(t)} \bigvee_{n=0}^{val(t)} B_n, ||\Gamma|| \rightarrow ||\Delta||.$$

By Lemma 3.1.1, this can be changed into a proof of

$$\bigvee_{n=0}^{val(t)} B_n, ||\Gamma|| \rightarrow ||\Delta||$$

without increasing the size. Cutting with (9.0.1), we get a proof of

$$B_{val(t)+1}, ||\Gamma|| \rightarrow ||\Delta||.$$

We can now apply the construction of Lemma 3.1.1 to get a proof of

$$A_{0,val(t)+1}, \ldots, A_{val(t)-1,val(t)+1}, ||\Gamma|| \rightarrow ||\Delta||.$$

Now observe that $A_{0,val(t)+1}, \ldots, A_{val(t),val(t)+1}$ is an extension cedent defining $x_0, \ldots, x_{val(t)}$, and this can be removed as follows. For each x_i starting with $i = val(t)$ and working our way down, apply $\exists$-left with x_i as the eigenvariable, and then cut its defining formula. Note that this construction does not add any new special inferences.

string $\forall$-right, string $\exists$-left: The construction is the same as the $V^1 - G_1^*$-translation theorem. The only point that should be made is that there is a new special $\forall$-right inference in the GPV_i^* proof if and only if the current rule in the TV^i proof is a special $\forall$-right inference. $\square$

9.0.2 The $GPV_i^*(c)$ reflection principles

Recall the proof of the Σ_{j+1}^q reflection principles for G_i^* in V^i. That proof was done by first using the Herbrand Theorem for G_i^* to get an interactive computation that witnesses the final formula. Then, using the maximization principle, we prove that there is a maximum number of rounds that the teacher can respond.

The reflection principles for $GPV_{i+1}^*(c)$ are proved in the same way. The difference is that for G_i^* there must be a polynomial number of rounds in the interactive computation; however, for a $GPV_{i+1}^*(c)$ proof of a Σ_j^q formula, there are at most c rounds with potentially a polynomial number of queries per round.

This is all informal, so at this point we will make the idea of the number of rounds more concrete. In the KPT witnessing theorem for G_i^*, there is an ordering $\prec$ on the extension variables E and the eigenvariables Q. Even though $\prec$ is a total ordering, it does not have to be, so we change it to a partial ordering meeting the necessary conditions. This ordering can be used to impose a partial ordering on the blocks of eigenvariables. See Theorem 5.1.2 for the definition of a block. We will say that a block of quantifiers is in round 1 if it is a minimal block, and, for $r > 1$, a block of quantifiers is in round r if all smaller blocks are in round $r - 1$. We say that $\prec$ has r rounds if every block is in round r. Note that we are using the convention that if a block is in round r then it is also in every round $r' > r$.

We are now able to extend the Herbrand Theorem to say something about the number of rounds.

Theorem 9.0.8. *VPV proves the following. Let π be a $GPV_i^*(c)$ proof of a Σ_j^q formula A, and assume that π is in free-variable normal form. Then there exists a G_i^* proof of a sequent $\Lambda \longrightarrow A^*$ and a partial ordering $\prec$ of the variables $Q_\pi \cup E$, where E is a set of variables that do not appear in π, with the following properties:*

- *Λ is an i-extension cedent defining the variables in E;*
- *for $e \in E$, if e depends on a variable $p \in Q_\pi \cup E$, then $p \prec e$;*
- *A^* is an i-instance relative to $\prec$ of an $\vee$-expansion of A; and*
- *$\prec$ has c rounds.*

Proof. The proof is almost the same as the proof of Theorem 3.2.16 for G_{i+1}^*. The only difference is that we add an extra point in the induction hypothesis: $\prec$ has c_S rounds where c_S is the maximum number of $\forall$-right instances introducing a new block along any branch on the proof of the sequent S.

Except for cut, all of the cases in the construction stay the same. It is important to notice that, in all of the binary inferences except for cut, the blocks from the two proof can remain incomparable. This means the blocks remain in the same round.

The only rule that needs to be discussed is cut. If the cut formula is a Σ_i^q formula, then the construction is the same as before. We can cut the formula in the new proof. The variables from the left and right subproofs can remain incomparable, so the blocks remain the the same round they we in before.

If the cut formula is not Σ_i^q, then it must be of the form $\exists z[z \leftrightarrow B]$, where B is Σ_i^q and does not mention z. The inference has the form

$$\frac{\exists z[z \leftrightarrow B], \Gamma \longrightarrow \Delta, \Omega \qquad \Gamma \longrightarrow \Delta, \exists z[z \leftrightarrow B], \Omega}{\Gamma \longrightarrow \Delta, \Omega}$$

By induction with the upper left sequent, we have a proof π of a sequent

$$\Lambda, q \leftrightarrow B, \Gamma' \to \Delta', \Omega',$$

with an ordering $\prec$. This is the exact sequent we want except that the eigenvariable q must be replaced with a new extension variable e. The ordering is changed by putting e in the same place as q. As well, since we have the exact same blocks as before, the blocks remain in the same round as before. $\square$

Using this version of the witnessing theorem, we can prove the reflection principles for $GPV_i^*(c)$ in TV^i.

Theorem 9.0.9. *TV^i proves Σ_j^q-RFN($GPV_i^*(c)$) for all $j \geq 0$ and $c \geq 0$.*

Proof. Take a $GPV_i^*(c)$ proof π of a formula A. By the theorem above, there is a G_i^* proof of a sequent $\Lambda \to \Theta'$ and an ordering $\prec$ meeting the conditions of the theorem. Let A^* be the $\vee$-expansion of A.

We define a formula $\psi(i, \prec, \Lambda, \Theta)$ that says there are values for the eigenvariables and extension variables that satisfy Λ and falsify the formulas associated with the groups of quantifiers in the first i rounds. See Theorem 5.1.2 to see how formulas are associated with blocks.

We can prove in TV^i that $\psi(0, \prec, \Lambda, \Theta)$ is true. By Σ_{i+1}^q-RFN(G_i^*), $\psi(c, \prec, \Lambda, \Theta)$ is false. Note that since c is a constant, we can prove by "brute force induction" in TV^i that there is a maximum value for $m < c$ that satisfies ψ regardless of the quantifier complexity of A^*.

Then we reason that one of the guesses in round $m + 1$ must be a witness for A^*. Then, by Lemma 5.1.3, A is true too. $\square$

Appendix A

Min Cut/Max Flow Theorem

In the introduction, we mentioned that a motive for looking at bounded arithmetic was the ability to connect theorems with complexity classes. This area of research is also called bounded reverse mathematics. The idea is to try to find the minimum set of axioms needed to prove a theorem. In this section, we will informally go through the reverse mathematics of the Min Cut/Max Flow Theorem. The idea is to familiarize the reader with this type of work.

Consider a directed graph G with two special nodes s and t. Each edge is labeled with a positive number that we refer to as the capacity. In our case, we will assume the numbers are all positive integers. This graph could be modeling a number of things. For now think of it as a network of computers. The edges represent cables connecting the computers and the number is the maximum number of bits of information the cable can transfer in a second.

A flow from s to t is a labeling of the edges with numbers such that each edge is label with a number smaller than its capacity. The inflow for a vertex is the sum of the values on the edges entering this vertex. The outflow is the same except with outgoing edges. The net flow for a vertex is its inflow minus its outflow. A valid flow from s to t is a flow where every vertex other than s and t has net flow equal to 0. The value of the flow is the net flow of s. A flow could be used to model the transfer of information from computer s to computer t. Every other computer passes information on as it receives it. Finding a maximum valid flow corresponds to finding the maximum amount of information that can be passed from s to t in a second. These types of problems come up in many different situations and are important.

An (s,t)-cut is a set of verticies that contains s but not t. The weight of a cut is the sum of the capacities of the edges that go from C to $V-C$ where V is the set of vertexes in the graph. In terms of flows, you can think of a cut as a bottle neck for the flow. So a minimum cut gives an upper-bound on the maximum flow.

The interesting part of the Min Cut/Max Flow Theorem is that the value of a maximum flow equals the weight of a minimum cut. This was first proved in [14]. We can show that this theorem can be formalized in VNL. Note that this does not involve proving that there exists a maximum flow. In fact, we do not believe that VNL proves that there exists a maximum flow since the problem of finding a maximum flow is P-complete. All we are proving is that, if there exists a maximum flow, there is a minimum cut with the same value.

We start by showing that the value of a cut cannot be smaller than the value of a flow. In fact, we show something stronger. Let C be a cut in a graph $G=(E,V)$ and F be a valid flow in that graph. Then the net flow for the cut is the sum of the flow on edges leaving C minus the sum of the flow on the edges entering C. Then we can show the following:

Lemma A.0.10. *VNL proves that, for any cut, the net flow of the cut equals the value of the flow.*

Proof. Let F be the flow and C be the cut. Define W as

$$W^{[i]} = \{s\} \cup \{u \in C : u < i\}$$

for $i \leq n$, where n is the number of vertexes in the graph. Then note that $W^{[i]} = C$ and that this can be done using Σ_0^B-COMP. We prove by induction on i that the net flow of the cut $W^{[i]}$ equals the value of the flow. This induction hypothesis involves sums, which can be done in TC^0, so the induction can be carried out in VNL.

For the base case, note that $W^{[0]} = \{s\}$. So the net flow of this cut is the net flow of s, which is the value of the flow.

Assume the induction hypothesis holds for i. If $W^{[i]} = W^{[i+1]}$, then the induction hypothesis holds for $i+1$. If $W^{[i]} \neq W^{[i+1]}$, then $W^{[i+1]} = \{i\} \cup W^{[i]}$. In this case, the theorem holds because the net flow of i is 0. We leave the details to the reader. □

We now look at the other direction. This can be formalized as a $\forall\Sigma_1^B$ formula as follow:

$$\begin{aligned} &\forall G \forall s \forall t \forall F \exists F' \exists C, \text{"if } F \text{ is a valid flow, then} \\ &\qquad (1)\ C \text{ is a cut that equals the flow or} \\ &\qquad (2)\ F' \text{ is a larger flow."} \end{aligned} \tag{A.0.1}$$

Theorem A.0.11. *VNL proves* (A.0.1).

Proof. The idea is to formalize the Ford-Fulkerson proof of this theorem [14]. It is also given in most introductory graph theory texts. Given G and F, we can define a new graph G' with the same vertexes. There is an edge from u to v in G' if the edge from u to v in G is not at full capacity or if the edge from v to u in G has non-zero flow. A path from s to t in this new graph represents a way to increase the flow. This will give us F'. If there is no path from s to t in G', then the set of nodes reachable from s in G' defines a cut C in G. Every edge leaving C must be at full capacity; otherwise, we could reach a node outside C. Similarly every edge entering C must have 0 flow. So the value of the flow equals the net flow on this cut (Lemma A.0.10). With what was said earlier, the net flow of this cut equals the value of the cut.

The graph G' can be defined by Σ_0^B-COMP. The set of nodes reachable from s in G' can be determined in VNL using the LC axiom. □

This is mildly interesting, but the key to bounded reverse mathematics is proving that VNL is the weakest theory that can prove this theorem. We do not prove this exactly, but we give an indication that this is the case.

Theorem A.0.12. *If VL proves* (A.0.1)*, then $L = NL$.*

Proof. To do this, we will show how to solve the directed graph reachability problem in L assuming VL proves the theorem. If VL proves (A.0.1), then there is a logspace function that, given a graph and a valid flow, returns either a larger flow, or cut with the same value. This follows from the VL witnessing theorem.

Given a graph G with nodes s and t, we want to determine if there is a path from s to t. Let F be the empty flow Using the function above, we can in logspace find either a larger flow (s and t are connected), or a cut with value 0 (s and t are not connected). □

This concludes this proof. Hopefully this gives of idea of the significance of this type of result.

Bibliography

[1] Mark Braverman. Witnessing SAT(2) and NAE-SAT(2) in L. 2003.

[2] Buss. Axiomatizations and conservation results for fragments of bounded arithmetic. In *CMWLC: Logic and Computation: Proceedings of a Workshop held at Carnegie Mellon University*. Contemporary Mathematics Volume 106, American Mathematical Society, 1990.

[3] Sam Buss and Jan Krajicek. An application of boolean complexity to separation problems in bounded arithmetic. *Proceedings of the London Mathematical Society*, 69:1–27, 1994.

[4] Samuel Buss. *Bounded arithmetic.* PhD thesis, Princeton University, Princeton, New Jersey, 1985.

[5] Samuel R. Buss. On Herbrand's theorem. *Lecture Notes in Computer Science*, 960:195–209, 1995.

[6] Samuel R. Buss. Relating the bounded arithmetic and polynomial time hierarchies. *Annals of Pure and Applied Logic*, 75(1-2):67–77, 1995.

[7] S. A. Cook. Feasibly constructive proofs and the propositional calculus, 1975.

[8] Stephen Cook. *Theories for Complexity Classes and their Propositional Translations*, pages 175–227. Quaderni di Matematica. 2003.

[9] Stephen Cook and Antonina Kolokolova. A second-order theory for NL. In *LICS '04: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 398–407, Washington, DC, USA, 2004. IEEE Computer Society.

[10] Stephen Cook and Tsuyoshi Morioka. Quantified propositional calculus and a second-order theory for NC^1. *Archive for Math. Logic*, 44(6):711–749, August 2005.

[11] Stephen Cook and Phuong Nguyen. Foundations of proof complexity: Bounded arithmetic and propositional translations. Available from http://www.cs.toronto.edu/˜sacook/csc2429h/book, 2006.

[12] Stephen Cook and Neil Thapen. The strength of replacement in weak arithmetic. *ACM Trans. Comput. Logic*, 7(4):749–764, 2006.

[13] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36–50, 1979.

[14] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399 – 404, 1956.

[15] Erich Grädel. Capturing complexity classes by fragments of second-order logic. *Theor. Comput. Sci.*, 101(1):35–57, 1992.

[16] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935–938, 1988.

[17] J. Johannsen. Satisfiability problem complete for deterministic logarithmic space, 2004.

[18] Antonina Kolokolova. *Systems of Bounded Arithmetic from Descriptive Complexity.* PhD thesis, University Of Toronto, 2005.

[19] Jan Kraj&#237;&#269;ek. Lower bounds to the size of constant-depth propositional proofs. *J. Symb. Logic*, 59(1):73–86, 1994.

[20] Jan Krajícek. *No Counter-Example Interpretation and Interactive Computation*, pages 287–293. Logic from Computer Science. Springer-Verlag, 1992.

[21] Jan Krajícek. Fragments of bounded arithmetic and bounded query classes. *Transactions of the American Mathematical Society*, 338:587–598, 1993.

[22] Jan Krajícek. *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Cambridge University Press, 1995.

[23] Jan Krajícek and Pavel Pudlák. Quantified propostitional calculi and fragments of bounded arithmetic. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 36:29–46, 1990.

[24] Jan Krajícek, Pavel Pudlák, and Gaisi Takeuti. Bounded arithmetic and the polynomial hierarchy. *Ann. Pure Appl. Logic*, 52(1-2):143–153, 1991.

[25] Jan Krajícek, Alan Skelley, and Neil Thapen. NP search problems in low fragments of bounded arithmetic. *The Journal of Symbolic Logic*, 72(2):649–672, 2007.

[26] Jan Krajícek and Gaisi Takeuti. On induction-free provability. *Annals of Mathematics and Artificial Intelligence*, 6:107–126, 1992.

[27] John C. Lind. Computing in logarithmic space. Mac Technical Memorandom 52, September 1974.

[28] Alexis Maciel and Toniann Pitassi. Conditional lower bound for a system of constant-depth proofs with modular connectives. In *LICS*, pages 189–200. IEEE Computer Society, 2006.

[29] Tsuyoshi Morioka. *Logical Approaches to the Complexity of Search Problems: Proof Complexity, Quantified Propositional Calculus, and Bounded Arithmetic.* PhD thesis, University Of Toronto, 2005.

[30] Phuong Nguyen. Separating dag-like and tree-like proof systems. In *LICS '07: Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science*, pages 235–244. IEEE Computer Society, 2007.

[31] Phuong Nguyen and Stephen A. Cook. Theories for TC^0 and other small complexity classes. *Logical Methods in Computer Science*, 2(1), 2006.

[32] Rohit Parikh. Existence and feasibility in arithmetic. *Journal of Symbolic Logic*, 36:494–508, 1971.

[33] J. Paris and A. Wilkie. Counting problems in bounded arithmetic. *Methods in Mathematical Logic*, Lecture Notes in Mathematics 1130:317–340, 1985.

[34] Steven Perron. GL^*: A propositional proof system for logspace. Master's thesis, University Of Toronto, 2005.

[35] Steven Perron. Quantified propositional logspace reasoning. Available from http://arxiv.org/abs/0801.4105., 2008.

[36] Chris Pollett. Structure and definability in general bounded arithmetic theories. *Annals of Pure and Applied Logic*, 100(1-3):189–245, 1999.

[37] Pavel Pudlák. Fragments of bounded arithmetic and the lengths of proofs. 2007.

[38] Michael Sipser. *Introduction to the Theory of Computation, Second Edition.* Course Technology, February 2005.

[39] Alan Skelley and Neil Thapen. The provably total search problems of bounded arithmetic. Available from http://www.math.cas.cz/˜thapen/, 2008.

[40] Robert Szelepcsenyi. The method of forcing for nondeterministic automata. *Bulletin of the European Association for Theoretical Computer Science*, 33:96–100, 1987.

[41] Domenico Zambella. Notes on polynomially bounded arithmetic. *The Journal of Symbolic Logic*, 61(3):942–966, 1996.