GL^*: A Propositional Proof System For Logspace

by

Steven Perron

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

Abstract

GL^*: A Propositional Proof System For Logspace

Steven Perron

Master Of Science

Graduate Department of Computer Science

University of Toronto

2005

In recent years, there has been considerable research exploring connections between propositional proof systems, theories of bounded arithmetic, and complexity classes. We know that NC^1 corresponds to G_0^* and that P corresponds to G_1^*, but no proof system corresponding to a complexity class between NC^1 and P has been defined.

In this work, we construct a proof system GL^*, which corresponds to L. Connections to the theory VL (Zambella's $\Sigma_0^p - rec$) are also considered. GL^* is defined by restricting cuts in the system G_1^*. The first restriction is syntactic: the cut formulas have to be $\Sigma CNF(2)$, which is a new class of formulas. Unfortunately that is not enough; the free variables in cut formulas must be restricted to parameter variables. We prove that GL^* corresponds to VL by translating theorems of VL into tautologies with small GL^* proof.

Acknowledgements

I would like to thank my supervisor, Dr. Stephen Cook, for guiding me, and for the idea this thesis is built on. I would also like to thank NSERC and the University of Toronto for their financial support.

Contents

Chapter 1

Introduction

One of the important areas in computer science is computational complexity. In this area, we are interested in determining the difficulty of computing a function. The main goal is to determine which functions can be computed in a reasonable amount of time and space. Functions are divided into complexity classes, and these classes form a hierarchy. Researchers want to prove that inclusions in this hierarchy are proper.

This problem has proved to be hard. So to help find another way to solve this problem, Parikh [16] proposed bounded arithmetic. Since that time, much research has gone into bounded arithmetic, but the work by Buss has had the largest effect. In his dissertation [4, 5], he introduced the bounded arithmetic hierarchy $S_2^1 \subseteq T_2^1 \subseteq S_2^2 \subseteq T_2^2 \subseteq \ldots$, and showed a number of conservation results. Later, Buss was able to show a connection between the polynomial hierarchy (PH) and the bounded arithmetic hierarchy [1].

The third area we consider is propositional proof complexity. In [12], Krajicek and Pudlak introduced the proof systems $G_1, G_2, \ldots$. They were able to show a connection between the bounded arithmetic hierarchy and these proof systems. This was accomplished by showing that T_2^i proves that G_i is sound and that G_i simulates T_2^i. Later, in [11], the proof system G_i^* was defined by restricting G_i to treelike proofs. G_i^* was shown to correspond to S_2^i in the same way G_i corresponds T_2^i.

This work provides a good framework. We have a well-defined method of connecting complexity classes, theories of bounded arithmetic, and propositional proof systems, but a few problems arise when searching for connections to smaller complexity classes. The main problem is with the strength of the functions in the language of S^i_2. We cannot multiply two binary number in AC^0, but the language has this function. The other problem is the translation used in the simulation. The strength of the language forces the translation to be more complicated than is necessary.

To solve these complications some researchers have explored second order theories. In the second order setting, the main hierarchy consist of the theories $V^0, V^1, \ldots$. There is a natural isomorphism, called the RSUV isomorphism, between S^i_2 and V^i for $i \leq 1$. So the connection between G^*_i and S^i_2 can be translated to a connection between G^*_i and V^i. In [17], Zambella independently proved the results that are in [1], but his proofs were in the second order setting.

The theory V^0 is an important base theory. We know V^0 corresponds to AC^0 [7] and bounded depth Frege [11]. Since many complexity classes have problems that are complete for the class under AC^0 reductions, it is possible to construct theories that correspond to those complexity classes by adding a single axiom. For example, VTC^0, VNC^1, and VL are constructed this way.

In [8], Cook and Morioka defined the second order theory VNC^1. As well, they developed the proof system G^*_0, which is defined slightly different than it is in [11]. They showed that Σ^B_1 theorems of VNC^1 could be translated into a family of tautologies that have small G^*_0 proofs. That is, G^*_0 simulates VNC^1. They also showed that the G^*_0 witnessing problem is complete for NC^1. This shows the three-way connection between VNC^1, G^*_0, and NC^1.

In [17], Zambella defined VL, which he called Σ^B_0-rec. He gave an informal argument that showed that VL corresponds to L. However, the point of the paper was not to examine VL so little else is known about it. In particular, there is no known proof system that corresponds to VL. A possible reason is that the axiom Σ_0^B-rec does not describe a total function in L since the output can be found in L only under an assumption, which is part of the axiom. The translation of this axiom would give a family of tautologies that do not belong to a class of formulas that can be evaluated in L.

In this work, we will define GL^*, a proof system that corresponds to VL and L. We reformulate the Σ_0^B-rec axiom to get a new theory VL', which is equivalent to VL. Then we prove tautologies of VL can be translated into a family of tautologies with small GL^* proofs using a VL' proof of the theorem.

The rest of the document is organized as follows as follows. The next chapter defines the complexity class L and the corresponding function class FL, and a recursive definition of FL is given. Chapter 3 defines the theories VL and $\overline{VL}$, which are the main theories used in this work. Chapter 4 defines the proof system GL^*, the main contribution of the thesis. Chapter 5 shows the connection between the VL and GL^*, which shows that GL^* has the complexity we want it to have.

Chapter 2

Log Space Computation

2.1 The Complexity Classes L and FL

Before we can define theories and propositional proof systems that corresponds to log space computation, we must know what we mean by log space computation. We cannot use a standard single tape Turing Machine TM because, in general, the amount of space the TM would be allowed to use is smaller than the size of the input. So we will use a slight modification of the standard TM. We will use a TM with 2 tapes: a read only input tape and a read-write right-infinite work tape. The work tape alphabet is $\{0, 1\}$, and initially the work tape is $000\ldots$. We use this convention so the contents of the work tape can be easily represented by a binary string. There are two kinds of inputs: number and (binary) string. Number inputs are natural numbers and will be input in unary. We will use lower case letter $a, b, c, \ldots$ to represent number variables. String inputs will represented as binary strings where the rightmost symbol of the string is 1 or a single 0 if the string is all zeros. We can think of a string as a finite subset of the natural numbers. On the input tape, inputs will be separated by a blank symbol $\sqcup$. For example, if the input is $(3, 4, 001010101, 1001)$ the input tape will be $111\sqcup1111\sqcup001010101\sqcup1001$. From now on, we assume all TM follow these conventions. Let M be a TM. Then $M(\vec{x}, \vec{X})$

will denote the output of the machine M as a boolean value.

Formally, a TM is a quadruple $M = (Q, q_{accept}, q_{reject}, \delta)$, where $Q = \{q_0, \ldots, q_m\}$ is a set of states with q_0 as the initial state, $q_{accept}, q_{reject} \in Q$ are the accepting and rejecting states, respectively, and δ is the transition function (with the condition that δ does not write on the input tape). The function δ is undefined for the states q_{accept} and q_{reject} since M halts when it enters those states. $M(\vec{x}, \vec{X})$ is true if M halts in the q_{accept} state on input $(\vec{x}, \vec{X})$. We define $s_M(\vec{x}, \vec{X})$ as the number of squares visited by the work-tape head during the computation with input $(\vec{x}, \vec{X})$. Now we are ready to define L and FL.

Definition 2.1.1. A relation $R(\vec{x}, \vec{X})$ is in L if and only if there exists a TM M such that $M(\vec{x}, \vec{X}) \iff R(\vec{x}, \vec{X})$, and $s_M(\vec{x}, \vec{X}) \leq O(log_2(\sum \vec{x} + \sum |\vec{X}|))$.

To define the corresponding set of functions, we use the general definition given in [7].

Definition 2.1.2. Let C be a class of relations. A number function $f(\vec{x}, \vec{X})$ is in FC if the graph of f, $f(\vec{x}, \vec{X}) = z$, is a relation in C and $f(\vec{x}, \vec{X}) \leq p(\vec{x}, |\vec{X}|)$ for some polynomial p.

A string function $F(\vec{x}, \vec{X})$ is in FC if the bit-graph of F, $F(\vec{x}, \vec{X})(i)$, is a relation in C and $|F(\vec{x}, \vec{X})| \leq p(\vec{x}, |\vec{X}|)$ for some polynomial p.

FL, the class of functions computable in log space, is defined this way.

2.2 The Language $\mathcal{L}_{FL}$

In this section, we define a language $\mathcal{L}_{FL}$. This language is defined so that we have an alternate method of showing a function is in FL. This method will be useful when we start proving connections between our theory of bounded arithmetic and FL.

The language extends $\mathcal{L}_A^2$. There is at least one function symbol in $\mathcal{L}_{FL}$ for every function in FL. The given recursive scheme is based on a scheme in [13]. The p-bounded number recursion is equivalent to the *log*-bounded string recursion given in [13]. The other schemes come from the definition of $\mathcal{L}_{FAC^0}$ in [7]. In the next definition, we define the set of function symbols in $\mathcal{L}_{FL}$ and give their intended meaning.

Definition 2.2.1. 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 $B1$-$B14$, $L1$, and $L2$, (see section 3.1 for these axioms) , and the axioms

$$pd(0) = 0 \tag{2.2.1}$$

$$pd(x+1) = x \tag{2.2.2}$$

$$min(x,y) = z \iff (z = x \wedge x \leq y) \vee (z = y \wedge y \leq x) \tag{2.2.3}$$

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

$$F_{\alpha,t}(\vec{x}, \vec{X})(i) \iff i < t(\vec{x}, \vec{X}) \wedge \alpha(i, \vec{x}, \vec{X}) \tag{2.2.4}$$

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

$$f_{\alpha,t}(\vec{x}, \vec{X}) \leq t(\vec{x}, \vec{X}) \tag{2.2.5}$$

$$z < t(\vec{x}, \vec{X}) \wedge \alpha(z, \vec{x}, \vec{X}) \implies \alpha(f_{\alpha,t}(\vec{x}, \vec{X}), \vec{x}, \vec{X}) \tag{2.2.6}$$

$$z < f_{\alpha,t}(\vec{x}, \vec{X}) \implies \neg\alpha(z, \vec{x}, \vec{X}) \tag{2.2.7}$$

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

$$f_{g,h,t}(0, \vec{x}, \vec{X}) = \min(g(\vec{x}, \vec{X}), t(\vec{x}, \vec{X})) \tag{2.2.8}$$

$$f_{g,h,t}(y+1, \vec{x}, \vec{X}) = \min(h(y, \vec{x}, \vec{X}, f(y, \vec{x}, \vec{X})), t(\vec{x}, \vec{X})) \tag{2.2.9}$$

The last scheme is called p-bounded number recursion. 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 log space. We now show every function in FL has at least one function symbol in $\mathcal{L}_{FL}$ that corresponds to it. The idea behind the proof is that the transition from one configuration of a TM to another configuration can be done in AC^0, and, since the contents of the work tape are not too large, we can simulate the computation of the TM using the number recursion.

Theorem 2.2.2. *Let $R(\vec{x}, \vec{X})$ be a relation in L. Then there is an open formula α over the language $\mathcal{L}_{FL}$ such that $R(\vec{x}, \vec{X}) \iff \alpha(\vec{x}, \vec{X})$.*

Proof. Let $M = (Q, q_{accept}, q_{reject}, \delta)$ be the TM that accepts R with $s_M(\vec{x}, \vec{X}) \leq c_1 \times log_2(\sum \vec{x} + \sum |\vec{X}|) + c_2$. The first thing to do is to define number functions $ns(s, v_i, v_w)$, $wt(s, v_i, v_w)$, and $it(s, v_i, v_w)$, which give the parts of the tuple output by δ. Let s be the number of the current state, v_i be the symbol on the square currently being read by the head of the input tape, and v_w be the symbol on the square currently being read by the head of the work tape. Then $ns(s, v_i, v_w)$ is meant to be the next state, as defined by δ. The function $wt(s, v_i, v_w)$ and $it(s, v_i, v_w)$ are meant to represent the action taken by the work tape and input tape, respectively. More specifically,

$$wt(s, v_i, v_w) = \begin{cases} 0, & \text{0 was written;} \\ 1, & \text{1 was written;} \\ 2, & \text{the head moved to the left;} \\ 3, & \text{the head moved to the right;} \\ 4, & \text{no action was taken.} \end{cases}$$

The function $it(s, v_i, v_w)$ has a similar meaning, except it can never have the value 0 or 1 since the input tape is read-only. For all these functions, if s is a halting state no actions is to be taken.

All these functions are easy to define. Therefore ns will be the only one defined explicitly. Let $t(\vec{x}, \vec{X})$ be the constant term $|Q|$. Let

$$\beta(q, s, v_i, v_w) =_{syn} \bigwedge_{(s', v_i', v_w') \in Q \times \Sigma_I \times \Sigma_W} (s = s' \wedge v_i = v_i' \wedge v_w = v_w' \implies q = q_n) \vee (s \in H \wedge q = s),$$

where $\delta(s', v_i', v_w') = (q_n, i_n, w_n)$ for some i_n and w_n and Σ_I and Σ_W are the alphabets for the input tape and work tape, respectively. Then the function ns is the same as $f_{\beta,t}$.

The rest of the proof is dependent on how a configuration is represented. Configurations will be represented by numbers. The number is interpreted as a 4-tuple. If the current configuration is $\langle w, p_w, p_i, s\rangle$, the content of the work tape is the binary representation of w, the work-tape head is on the p_wth square of the work tape, the input-tape head is on the p_ith square of the input tape, and the current state is q_s. Let n be the size of the input. Then the number of squares used on the work tape is bounded by $c_1 \times log_2(n) + c_2$. This implies $w \leq 2^{c_1 \times log_2(n) + c_2} = 2^{c_2} \times n^{c_1}$. Also, $p_w \leq log_2(w) \leq 2^{c_2} \times n^{c_1}$, $p_i \leq n$, and $s \leq |Q|$. Putting all of this together, we know a number representing any configuration of M is less then a polynomial in n. Let $t(\vec{x}, \vec{X})$ be a term that bounds this polynomial. We will use t as the bounding term when we use the p-bounded number recursion. Now to define the functions we will use.

Now we can define $init(\vec{x}, \vec{X})$, which returns the initial configuration of M with the given input. This is simple because the work tape is empty with the head in the first position. The initial state is q_0 and the head of the input tape is at position 0. Note there is no mention of $\vec{x}$ or $\vec{X}$; so $init(\vec{x}, vecX)$ is a constant function, which is obviously in $\mathcal{L}_{FL}$.

Next we need to define $next\_config(\langle w, p_w, p_i, s\rangle, \vec{x}, \vec{X})$. For this we let

$$\begin{aligned}
\gamma(\langle w', p'_w, p'_i, s'\rangle, \langle w, p_w, p_i, s\rangle, \vec{x}, \vec{X}) \iff\ & s' = ns(s, vi(p_i, \vec{x}, \vec{X}), bit(p_w, w)) \\
&\wedge it(s, vi(p_i, \vec{x}, \vec{X}), bit(p_w, w)) = 2 \implies p_i = p'_i + 1 \\
&\wedge it(s, vi(p_i, \vec{x}, \vec{X}), bit(p_w, w)) = 3 \implies p_i + 1 = p'_i \\
&\wedge it(s, vi(p_i, \vec{x}, \vec{X}), bit(p_w, w)) = 4 \implies p_i = p'_i \\
&\wedge wt(s, vi(p_i, \vec{x}, \vec{X}), bit(p_w, w)) = 2 \implies p_w = p'_w + 1 \wedge w = w' \\
&\wedge wt(s, vi(p_i, \vec{x}, \vec{X}), bit(p_w, w)) = 3 \implies p_w + 1 = p'_w \wedge w = w' \\
&\wedge wt(s, vi(p_i, \vec{x}, \vec{X}), bit(p_w, w)) = 4 \implies p_w = p'_w \wedge w = w' \\
&\wedge wt(s, vi(p_i, \vec{x}, \vec{X}), bit(p_w, w)) = 0 \implies p_w = p'_w \wedge w = set(p_w, w') \\
&\wedge wt(s, vi(p_i, \vec{x}, \vec{X}), bit(p_w, w)) = 1 \implies p_w = p'_w \wedge set(p_w, w) = w'
\end{aligned}$$

This formula simply looks at the possible outputs of the transition function and makes sure the configuration changes appropriately. Then it is easy to see that $next\_config(\langle w, p_w, p_i, s\rangle, \vec{x}, \vec{X})$ is $f_{\gamma,t}(\langle w, p_w, p_i, s\rangle, \vec{x}, \vec{X})$. With these two functions, we can define $f_{init,next\_config,t}(m, \vec{x}, \vec{X})$, which gives the configuration of machine M with input $(\vec{x}, \vec{X})$ at time m. It is well known that any TM that has logarithmic space bounds and that halts on every input will halt in polynomial time: there is only a polynomial number of possible configuration. Let $t'(x)$ be a term that bounds the runtime of M. Then we get

$$R(\vec{x}, \vec{X}) \iff \langle f_{init,next\_config,t}(t'(\vec{x}, \vec{X}), \vec{x}, \vec{X})\rangle^{-1}_{4,4} = q_{accept},$$

where $\langle a\rangle^{-1}_{m,n}$ is the inverse of the tupling function: it returns the mth element of the n-tuple represented by a. This works because it essentially runs the TM until it halts and then checks if it is in the accept state. □

By the previous theorem and the definition of FL, we get the following corollary.

Corollary 2.2.3. *The function symbols in $\mathcal{L}_{FL}$ represent precisely the functions in FL.*

Proof. Let $F(\vec{x}, \vec{X})$ be a function in FL. Then the bit-graph of F is in L. By theorem 2.2.2, there is an open formula $\alpha(i, \vec{x}, \vec{X})$ in $\mathcal{L}_{FL}$ such that $F(\vec{x}, \vec{X})(i) \iff \alpha(i, \vec{x}, \vec{X})$. We also know there is a polynomial bound on $|F(\vec{x}, \vec{X})|$, so it follows there is a term $t(\vec{x}, \vec{X})$ such that $F \equiv F_{alpha,t}$. A similar argument can be made for number functions. $\square$

Chapter 3

Bounded Arithmetic

In this chapter, we introduce the theories of bounded arithmetic we will use. These are second order theories, as was mentioned in the introduction. Many theories are constructed by adding an axiom to the theory V^0, so we will take a quick look at this theory.

3.1 V^0 and Its Properties

V^0 is defined over the language $\mathcal{L}_A^2$. The language includes the functions $+$ and $\times$, which have their usual meaning, and the constants 0 and 1. The language also includes two functions string. The first is the size of a string, $|X|$, and the second accesses the bits of the sting; $X(i)$ is true if the the ith bit of X is 1. V^0 is axiomatized by the following axioms:

$B1.\ x+1 \neq 0$

$B2.\ x+1 = y+1 \implies x = y$

$B3.\ x+0 = x$

$B4.\ x+(y+1) = (x+y)+1$

$B5.\ x \times 0 = 0$

$B6.\ x \times (y+1) = (x \times y) + x$

$B7.\ x \leq x+y$

$B8.\ (x \leq y \wedge y \leq x) \implies x = y$

$B9.\ 0+1 = 1$

$B10.\ 0 \leq x$

$B11.\ x \leq y \wedge y \leq z \implies x \leq z$

$B12.\ x \leq y \vee y \leq x$

$B13.\ x \leq y \iff x < y+1$

$B14.\ x \neq 0 \implies \exists y \leq x(y+1 = x)$

$L1.\ X(y) \implies y < |X|$

$L2.\ y+1 = |X| \implies X(y)$

$\Sigma_0^B - COMP$: $\exists Z \leq y \forall i < y(Z(i) \iff \phi(i, \vec{x}, \vec{X}))$, where ϕ is Σ_0^B.

The axioms $B1 - B14$, $L1$, and $L2$ are known as the 2-BASIC axioms. These axioms define the functions and relations in $\mathcal{L}_A^2$. This theory has been extensively studied. We will mention a few properties of V^0 that will be used later.

Theorem 3.1.1 ([7] Theorem 2.11). *V^0 proves the following schemes:*

$$\Sigma_0^B - IND : [\phi(0) \wedge \forall z(\phi(z) \implies \phi(z+1))] \implies \forall z \phi(z)$$

and

$$\Sigma_0^B - MIN : [\exists z \phi(z)] \implies \exists z[\phi(z) \wedge \forall y < z \neg \phi(y)],$$

where ϕ is any Σ_0^B formula.

Researchers are often interested in determining whether or not a theory is finitely axiomatizable. For some theories, this is an important open problem, but for V^0, it is known that it is finitely axiomatizable [9].

The final property we are interested in is that V^0 corresponds to AC^0. To make this connection, we are interested in finding what functions are definable in the theory.

Definition 3.1.2. Let Φ be a set of formulas over a language $\mathcal{L}$, and let T be a theory over the same language. Then a string function $F(\vec{x}, \vec{X})$ is Φ-definable in T if there exists a formula ϕ in Φ such that

$$\phi(\vec{x}, \vec{X}, Y) \iff F(\vec{x}, \vec{X}) = Y$$

and

$$T \vdash \forall \vec{x} \forall \vec{X} \exists ! Y \phi(\vec{x}, \vec{X}, Y).$$

A similar definition exists for number functions.

Essentially, this definition says that T proves the function is total and has a unique output. Using this definition we can state the connection between V^0 and AC^0 more formally.

Theorem 3.1.3 ([7] Theorem 2.26). *A function is Σ_1^B definable in V^0 if and only if it is in FAC^0.*

3.2 The Theory VL and $\overline{VL}$

3.2.1 Definition of VL and $\overline{VL}$

In this section, we define two theories that correspond to FL. The first is VL, which is the same as Σ_0^p-rec [18]. The main axiom of VL is the $X - rec$ axiom:

$$\begin{aligned}
&[\forall x \leq a \exists y \leq a X(x, y)] \implies \exists Z, \forall w \leq b \exists ! x \leq a Z(w, x) \\
&\qquad \wedge \forall w < b \forall x \leq a \forall y \leq a [Z(w, x) \wedge Z(w+1, y) \implies X(x, y)].
\end{aligned} \tag{3.2.1}$$

In this formula, Z is a path in the directed graph with a nodes and edge relation X. The formula says that if every node has out-degree at least 1, then there is a path of length b. If $Z(w, x)$ is true, then x is the wth node in the path. We now define VL.

Definition 3.2.1. VL is the theory axiomatized by the axioms of V^0 plus the $X - rec$ axiom.

Since V^0 is finitely axiomatizable, and VL has only one more axiom than V^0, we get the following corollary.

Corollary 3.2.2. *VL is finitely axiomatizable.*

Since VL in an extension of V^0, VL proves everything that V^0 proves, including the $\Sigma_0^B - MIN$ and $\Sigma_0^B - IND$ formulas.

One drawback of the $X - rec$ axiom is that we do not know where the path starts. To get around this problem, we show, as in [18], that VL proves the following scheme:

$$[\forall w < b \forall x \leq a \exists y \leq a \phi(w,x,y)] \implies \exists Z, \forall w \leq b \exists! x \leq a Z(w,x) \\ \wedge \forall w < b \forall x \leq a \forall y \leq a[Z(w,x) \wedge Z(w+1,y) \implies \phi(w,x,y)], \tag{3.2.2}$$

where ϕ is a Σ_0^B formula. This formula says there exists a path, as in the $X - rec$ axiom, but, in this formula, the graph can change after every step. To get a path that starts at a particular node s, we make every edge go to s for the first step; then we continue with the original graph. After removing the first node in that path, we have a path in the original graph that starts at s.

Theorem 3.2.3. *VL proves Formula 3.2.2*

Proof. We define

$$\psi(\langle x,w \rangle, \langle y,z \rangle) =_{syn} \phi(w,x,y) \wedge [(w < b \wedge z = w+1) \vee (w = b \wedge z = 0)].$$

This formula gives the edge relation of a graph in which the nodes are pairs. The first element in the pair is a node in the original formula. The second element represents its position in a path of the original formula. So an edge from $\langle 8,4\rangle$ to $\langle 3,5\rangle$ means, if 8 is the 4th node in a path in the original formula, then 3 is a potential 5th node. When we have reached the end of the path, we start over; that is, we go from $\langle 8,b\rangle$ to say $\langle 3,0\rangle$.

Using $X - rec$, we can find a path Z of length $2b+1$ in the graph of ψ. We now find a continuous sub-path of Z that has the form $\langle x_1, 0\rangle \dots \langle x_b, b\rangle$. One of the first $b+1$ nodes in the path must have the form $\langle x_1, 0\rangle$: there are only $b+1$ possible values for the second element. Then the following b elements complete the path. We can define a path in the original formula as $Z'(i,x) \iff x = x_i$. This Z' witnesses Formula 3.2.2, and the proof is complete. □

Our goal is to show that VL corresponds to FL in the same way V^0 corresponds to AC^0. We will use the method that was used in [7] to show that V^0 corresponds to AC^0. The idea is to define a universal theory $\overline{VL}$ that has a function symbol for every function in FL. Once we show this theory is a conservative extension of VL, we use a second order version of the Herbrand Theorem to show a function is Σ_1^B definable in VL if and only if it is in $\mathcal{L}_{FL}$.

$\overline{VL}$ 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 $2-BASIC$ axioms. $B14$ is removed because of the existential quantifier. This is OK since there is a predecessor function pd in $\mathcal{L}_{FL}$, with the two defining axioms 2.2.1 and 2.2.2, which can be used to prove $B14$. Now we define $\overline{VL}$.

Definition 3.2.4. $\overline{VL}$ is the theory over the language $\mathcal{L}_{FL}$ with axioms $B1$-$B13$; 2.2.1; 2.2.2; $L1$; $L2$; axiom 2.2.4 for each string function $F_{\alpha,t}$ in $\mathcal{L}_{FL}$; axioms 2.2.5, 2.2.6, and 2.2.7 for each number function $f_{\alpha,t}$ in $\mathcal{L}_{FL}$; and axioms 2.2.8 and 2.2.9 for each number function $f_{g,h,t}$ in $\mathcal{L}_{FL}$.

Since $\mathcal{L}_{FAC^0}$ is contained in $\mathcal{L}_{FL}$, we can see $\overline{VL}$ is an extension of $\overline{V^0}$, and therefore

$\overline{VL}$ proves the $2-BASIC$ and Σ_0^B-COMP axioms (see [7]). To show that $\overline{VL}$ proves the $X-rec$ axiom, we define a number function $xmin(a, X)$, which returns the index of the first bit in X that is set to 1 or a, whichever is smaller. The function is defined as follows:

$$xmin(a, X) = i \iff (i < a \implies X(i)) \wedge \forall j < i \neg X(j).$$

It is easy to see this function is in $\mathcal{L}_{FL}$ since it has a Σ_0^B definition. Now define $path(a, b, s, X)$ as

$$\begin{aligned} &path(a, 0, s, X) = s \\ &path(a, b+1, s, X) = xmin(a, X^{[path(a,b,s,X)]}), \end{aligned} \tag{3.2.3}$$

where $X^{[i]}(j) \iff X(i,j)$. The function $path$ finds a path that starts at node s in the graph with edge relation X and a nodes; then the next node is found by searching for the first node that is adjacent to the current node. If no such node exists, the next node is a, and we continue as before. The value of $path(a, b, s, X)$ is x when x is the bth node in that path. Then we can witness the $X-rec$ axiom with

$$PATH(a, b, X)(w, x) \iff x \leq a \wedge w \leq b \wedge x = path(a, w, 0, X).$$

The variables a,b, and X are the free variables in the $X-rec$ axiom, and the string output by $PATH$ is the Z the axiom says exists. This shows that $\overline{VL}$ is an extension of VL. So now we want to show this extension is conservative.

Let $OPEN(\mathcal{L})$ be the set of formulas over the language $\mathcal{L}$ that do not have any quantifiers, number or string.

Theorem 3.2.5. *$\overline{VL}$ is a conservative extension of VL.*

Proof. We have already shown $\overline{VL}$ is an extension of VL. We now want to prove the extension is conservative. This is done by showing that every function in $\mathcal{L}_{FL}$ is Σ_1^B definable in VL and its defining axioms can be proved from the Σ_1^B definition. Then it follows that $\overline{VL}$ is a conservative extension of VL.

We will actually prove something stronger. Let T be a theory over the language $\mathcal{L}$ that is a conservative extension of VL. Suppose that $\mathcal{L}$ contains all the functions necessary to define $f \in \mathcal{L}_{FL}$ or $F \in \mathcal{L}_{FL}$, and that T proves $OPEN(\mathcal{L}) - COMP$. Let $T+$ be the theory over the language $\mathcal{L}+$ that extends T with an axiom defining the new function. Then $T+$ is a conservative extension of T, and $T+$ proves $OPEN(\mathcal{L}+) - COMP$. This proof is done by induction on the number of functions needed to define the new function.

For the base case the function is pd, min, or in $\mathcal{L}_A^2$. In this case, the theory T is VL, and it is easy to see that the function can be Σ_0^B defined since it is an AC^0 function, and that VL proves $OPEN(\mathcal{L}+) - COMP$

For the inductive step, let α be an open formula over the language $\mathcal{L}$, and let t be a term over $\mathcal{L}_A^2$. We look at two cases based on the definition of $\mathcal{L}_{FL}$ (see Definition 2.2.1).

In the first case, we add $f =_{syn} f_{\alpha,t}$ to $\mathcal{L}$ and the defining axiom

$$f() = z \iff \forall z' < z \neg\alpha(z') \wedge [(\alpha(z) \wedge z < t) \vee z = t]$$

to T to get the theory $T+$ over the language $\mathcal{L}+$. We show that $T+$ is a conservative extension of T. Since T proves the $OPEN(\mathcal{L}) - MIN$ formulas, which can be proved from the $OPEN(\mathcal{L}) - COMP$ axioms, T proves that there exists a unique output for f, the minimum value satisfying α or t if no such value exists. Therefore $T+$ is a conservative extension of T.

So now we want to show $T+ \vdash OPEN(\mathcal{L}+) - COMP$. Let α be any open formula over the language $\mathcal{L}$. We proceed by induction on the number of occurrences of f in α. For the base case, suppose α does not mention f. Then $T+$ proves comprehension for this formula since T proves $OPEN(\mathcal{L}) - COMP$. For the inductive step, let $f(\vec{s}, \vec{S})$ be a term in α, where $\vec{s}$ are number terms that do not mention f and $\vec{S}$ are string terms that do not mention f. Let $\alpha'(z)$ be α with $f(\vec{s}, \vec{S})$ replaced by z; that is $\alpha =_{syn} \alpha'(f(\vec{s}, \vec{S}))$. Since α is open, it is easy to see

$$T+ \vdash z = f(\vec{s}, \vec{S}) \implies [\alpha \iff \alpha'(z)].$$

So by induction, $T+$ proves comprehension for α'. Since $T+$ proves that $\exists z, z = f(\vec{s}, \vec{S})$, we know $T+$ proves comprehension for α.

For the second case, we add the function $F_{\alpha,t}$ to $\mathcal{L}$ and add the defining axiom

$$F()(i) \iff \alpha(i) \wedge i < t.$$

Since T proves $OPEN(\mathcal{L}) - COMP$, T proves there exists a unique output for F; therefore the extended theory is a conservative extension of T. It is also easy to see the new theory proves $OPEN(\mathcal{L}+) - COMP$; the same argument as in the previous case can be carried out, except the number variable z becomes a string variable Z.

For the final case, we want to add a function $f =_{syn} f_{g,h,t}$, where $g, h \in \mathcal{L}$ and t is a term over $\mathcal{L}_A^2$. This covers the last method of constructing functions in Definition 2.2.1. Let

$$\beta(w, i, j) =_{syn} i \leq t \wedge j \leq t \wedge [h(w, i) = j \vee (j < h(w, i) \wedge j = t)].$$

The formula gives us the next value for f defined by Formula 2.2.9. By the previous case and by Lemma 3.2.7 (see below), we can assume $F =_{syn} F_{\beta,\langle t,t\rangle}$ and $path$ are in $\mathcal{L}$. If they are not, we can add them. Let the defining axiom for f be

$$f(i) = z \iff path(t, i, g(), F()) = z.$$

Since T proves g, F, and $path$ have an output, the same goes for f. Also, by an argument similar to the previous cases, the new theory proves $OPEN(\mathcal{L}+) - COMP$. $\square$

Corollary 3.2.6. *(1) Every function in FL can be Σ_1^B defined in VL, and its defining axioms proved. (2) Every function Σ_1^B-definable in VL is in FL.*

Proof. (1) follows directly from the previous theorem. To prove (2), let $\exists Z < tA(Z)$ be a Σ_1^B formula provable in VL. Then it is provable in $\overline{VL}$ since $\overline{VL}$ is an extension of VL. Since $\overline{VL}$ is universal, we know from the Herbrand Theorem there must be a function F in $\mathcal{L}_{FL}$ such that $\overline{VL}$ proves $A(F())$ (see [7] Theorem 2.19). This means the function Σ_1^B-defined by $\exists Z < tA(Z)$ is F, which is in FL. □

So all that is left to do is prove that *path* is Σ_1^B definable in VL and the theory with *path* proves open comprehension with the new language.

Lemma 3.2.7. *Let $\mathcal{L} = \mathcal{L}_A^2 \cup \{path\}$, and let T be the theory over the language $\mathcal{L}$ with the axioms of VL and the defining axiom*

$$\begin{aligned} path(a,b,s,X) = z \iff & \exists Z, \forall i \le a[Z(0,i) \iff i = s] \\ & \wedge \forall j < b \forall i \le a[Z(j+1,i) \iff \exists k \le a(Z(j,k) \wedge i = xmin(a, Z^{[k]})] \\ & \wedge Z(b,z). \end{aligned} \tag{3.2.4}$$

Then T is a conservative extension of VL and $T \vdash OPEN(\mathcal{L}) - COMP$.

Proof. To show that T is a conservative extension of VL, it suffices to show that $VL \vdash \exists! z, \phi(z, w, \vec{x}, \vec{X})$, where ϕ is the Σ_0^B formula that defines *path*. The first step is to prove

$$\begin{aligned} &\exists Z, \forall i \le a[Z(0,i) \iff i = s] \\ &\quad \wedge \forall j < b \forall i \le a[Z(j+1,i) \iff \exists k \le a(Z(j,k) \wedge i = xmin(a, Z^{[k]}))] \end{aligned} \tag{3.2.5}$$

Let

$$\begin{aligned} \psi(w,i,j,s,a,X) =_{syn} & [w = 0 \implies j = s] \\ & \wedge [0 < w \implies j = xmin(a, X^{[i]})] \end{aligned} \tag{3.2.6}$$

Then existence follows from $\Sigma_0^B - COMP$ with this formula and Formula 3.2.2.

Suppose Z_1 and Z_2 both witness Formula 3.2.5. We want to show that $\forall i \leq b \forall j \leq a, Z_1(i,j) \iff Z_2(i,j)$. This is done by induction on i. The base case is easy. $\forall j \leq a, Z_1(0,j) \iff Z_2(0,j)$ is true; otherwise $s \neq s$, which is not possible. As the inductive hypothesis, suppose $\forall j \leq a, Z_1(i,j) \iff Z_2(i,j)$. If $Z_1(i+1,j)$ is true, then there exists a $j' \leq a$ such that $Z_1(i,j')$ and $j = xmin(a, X^{[j']})$. By the induction hypothesis, $Z_2(i,j')$ is true, which implies $Z_2(i,j)$ since $xmin$ has a unique output. Doing the same argument in other direction implies $\forall j \leq a, Z_1(i+1,j) \iff Z_2(i+1,j)$. Since VL proves $\Sigma_0^B - IND$, the induction can be done in VL.

The next step is to prove that, for any witness Z to 3.2.5, $\forall w \leq b, \exists! z \leq a, Z(w,z)$. This would imply a unique z for ϕ. This is true since our construction of Z that shows existence showed that it also satisfies Formula 3.2.2. This means *path* is Σ_1^B definable in VL and T is a conservative extension of VL. To prove that T proves $OPEN(\mathcal{L}){-}COMP$, we can carry out the same argument at in Theorem 3.2.5. $\square$

Before we move on, we should note that $\overline{VL}$ proves $\Sigma_0^B(\mathcal{L}_{FL}) - COMP$. This follows from the next lemma.

Lemma 3.2.8. *For every $\Sigma_0^B(\mathcal{L}_{FL})$ formula ϕ there is an $OPEN(\mathcal{L}_{FL})$ formula α such that*

$$\overline{VL} \vdash \phi \iff \alpha.$$

Proof. The proof is done by induction on the number of quantifiers in ϕ. If there are none, then we are done. For the inductive step, suppose

$$\phi =_{syn} \exists z < t \psi(z).$$

By induction there exists an open formula β such that $\overline{VL} \vdash \psi(z) \iff \beta(z)$. Let $f \equiv f_{\beta,t}$. Then $T \vdash \phi \iff \beta(f())$. The construction for other connectives is left to the reader. □

3.2.2 VL and VTC^0

In this section, we want to show that VL is an extension of VTC^0 [15], which corresponds to TC^0. To show that VL extends this theory, it suffices to show that VL proves its axioms. Because this theory is axiomatized by the axiom of V^0 plus the $NUMONES$ axiom (see below), all that is left to show is that VL proves that axiom.

Lemma 3.2.9. *VL proves the following formula:*

$$NUMONES : \forall X \exists Y, \phi_1(X,Y) \wedge \phi_2(X,Y),$$

where

$$\phi_1(X,Y) =_{syn} \forall i \leq |X| \exists! j \leq |X| [Y(i,j) \wedge Y(0,0)]$$

and

$$\phi_2(X,Y) =_{syn} \forall i < |X| \forall j \leq |X| [(Y(i,j) \wedge X(i) \implies Y(i+1,j+1)) \wedge (Y(i,j) \wedge \neg X(i) \implies Y(i+1,j))].$$

Proof. Define

$$\phi(\langle i_1,p_1\rangle, \langle i_2,p_2\rangle, X) =_{syn} i_2 = i_1 + 1 \wedge (X(i) \implies p_2 = p_1 + 1) \wedge (\neg X(i) \implies p_2 = p_1).$$

Then VL proves that there exists a path of length $|X|$ that starts at node $\langle 0,0\rangle$ in the graph with edge relation ϕ. It is easy to observe this path satisfies $NUMONES$. □

This lemma implies VL is an extension of VTC^0. Since VTC^0 proves

$$X - PHP : [\forall z \leq a \exists y < a X(y,z)] \implies [\exists y < a \exists z_1 \leq a \exists z_2 < z_1 (X(y,z_1) \wedge X(y,z_2))],$$

we get the following corollary.

Corollary 3.2.10. *$\overline{VL}$ proves the $\Sigma_0^B(\mathcal{L}_{FL}) - PHP$*

Proof. $\overline{VL}$ prove $X - PHP$ scheme and the $\Sigma_0^B(\mathcal{L}_{FL}) - COMP$ scheme. □

This will be useful later on. This also implies VL can define string multiplication and prove its basic properties.

Chapter 4

Propositional Proof Complexity

In propositional proof complexity, we are interested in finding lower bounds on the size of proofs in proof systems. As it turns out, there are connections with computational complexity and bounded arithmetic.

4.1 The Proof System G and Its Fragments

The first proof systems correspond to small complexity classes such as AC^0 and NC^1. Frege systems are known to correspond to NC^1 [7]. There was a need to find a way to extend Frege to find proof systems that correspond to larger complexity classes. One method is to add an extension rule, which gives extended Frege. Another method is to move to the quantified propositional calculus. In [12], Krajicek and Pudlak defined a proof system G for the quantified propositional calculus, and they examined its fragments. Later G^* was defined by restricting G to treelike proofs.

We will work with the definition of G_i^* given in [14, 8], which is slightly different than the definition in [12]. The set $\Sigma_0^q = \Pi_0^q$ will be the set of all quantifier free propositional formulas. Then Σ_i^q is the set of formulas of the form $\exists \vec{z}, A(\vec{z}, \vec{x})$, where $A \in \Pi_{i-1}^q$, and Π_i^q is the set of formulas of the form $\forall \vec{z}, A(\vec{z}, \vec{x})$, where $A \in \Sigma_{i-1}^q$. Note that this definition only allows formulas that are in prenex form; that is the formula has all quantifiers in front then a quantifier free part. Also note that free variables will be denoted by x and bound variables are denoted by z. The notation $\vec{x}$ is used to denote the free variables $x_1, x_2, \ldots, x_n$, for some n. The same for $\vec{z}$ with bound variables.

The proof system G is an extension of Gentzen's sequent calculus PK, which is LK for the propositional calculus (see [6] for a description of PK). The axiom schemes are the same: $A \to A$, for any atomic formula A; $0 \to$; and $\to 1$. The rules of inference include the rules of PK plus four quantifier rules:

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

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

where x does not appear in the bottom sequent of the $\exists$-left and $\forall$-right rules, and B is a Σ_0^q formula that does not mention any z-variable. For any formula in Γ or Δ, the formula in the upper sequent is the direct ancestor of the corresponding formula in the lower sequent. The formula $A(B)$ and $A(x)$ are direct ancestors of the formula $\exists z A(z)$ in the appropriate rule. The *direct ancestor* relation is defined similarly for every other rule. The *ancestor* relation is the transitive closure of the direct ancestor relation.

The proof system G_i^* is the proof system G^* with cuts restricted to Σ_i^q formulas. Suppose π is a proof in one of these proof systems. Then a variable x_i is a *parameter variable* if it appears in the final sequent.

4.2 $\Sigma CNF(2)$ Formulas

The fragments G_0^* and G_1^* were shown to correspond to NC^1 and P, respectively [8, 11]. This means that, in our proof system, cut formulas should be restricted to some subset of the Σ_1^q formulas. As well, the formulas should somehow relate to L.

To this end, we look at $CNF(2)$ formulas. A formula is a $CNF(2)$ formula if it is a CNF formula and no variable appears more than twice in the formula. The problem $SAT(2)$ is given a $CNF(2)$ formula determine whether or not it is satisfiable. In [10], Johannsen showed that $SAT(2)$ is L-complete.

The class of formulas $CNF(2)$ is not the class we are looking for. Notice that SAT is NP-complete, but G_0^* corresponds to NC^1. The reason is a formula can be evaluated in NC^1. We will work with this idea, and we will develop a class of formulas where evaluating a formula is L-complete.

Definition 4.2.1. 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.

There is a problem when we parse these formulas. There is an ambiguity when a formula with principal connective $\vee$ replaces an x-variable. We cannot tell if two variables were replaced or just one. To remove this ambiguity, we assume some parsing information is given. In particular, we assume we know which $\vee$'s separate elements in a clause and which are part of a formula that replaced a free variable.

The problem $EVAL(\Sigma CNF(2))$ is, given a $\Sigma CNF(2)$ formula and an assignment to the free variables, does the formula evaluate to true.

Lemma 4.2.2. *$EVAL(\Sigma CNF(2))$ is L-complete.*

Proof. We begin by showing that the problem is L-hard. Let $SAT(2)^-$ be the same problem as $SAT(2)$ except the formula does not contain any pure literals. A pure literal is a literal that appears positively, but not negatively, or vice versa. This problem is L-complete [10]. Let $F(\vec{x})$ be a $CNF(2)$ formula with no pure literals. This means no literal can appear more than once; otherwise its negation could not appear, and it would be a pure literal. This means that $\exists\vec{z}, F(\vec{z})$ is a $\Sigma CNF(2)$ formula, and $F(\vec{x})$ is satisfiable if and only if $\exists\vec{z}, F(\vec{z})$ evaluates to true, by definition. This means $EVAL(\Sigma CNF(2))$ is L-hard.

So now we prove $EVAL(\Sigma CNF(2))$ is in L. Let $\exists\vec{z}, F(\vec{x}, \vec{z})$ be a $\Sigma CNF(2)$ formula that has form 2 from the definition. Given an assignment to the free variables, we are able to simplify F in the obvious way to get F'. Then we claim that F' is a $CNF(2)$ formula. To prove the claim, note that a literal will appear at most once in F' by property (2); this means a variable will appear at most twice. Also, F' is a CNF formula by property (1). We can now check the satisfiability of F'. For the next case, suppose $\vec{x}$ is substituted by $\vec{B}$, where the Bs are Σ_0^q formulas. Then we simply evaluate the Bs, which can be done in NC^1, and do as in the first case. The final case is if F does not have any quantifiers. Then the formula can be evaluated in NC^1. $\square$

4.3 The Proof System GL^*

Since L is between NC^1 and P, we want to find a proof system between G_0^* and G_1^*. This proof system will be GL^*, and is defined as follows.

Definition 4.3.1. GL^* is the propositional proof system G^* with cuts restricted to $\Sigma CNF(2)$ formulas in which no cut formula that is not Σ_0^q contains a non-parameter free variable.

Recall a parameter variable, with respect to a proof, are those that appear in the final sequent of the proof. Note that GL^* extends G_0^* since every cut formula in G_0^* is Σ_0^q, which is a valid cut formula in GL^*, and G_1^* extends GL^* since every cut in GL^* is a valid cut in G_1^*. This means our proof system corresponds to a complexity class between

NC^1 and P. Later we show that it corresponds to L.

4.3.1 Justifying the Restrictions

Before we continue, we should examine the definition of GL^* to try to understand the reason for the restrictions. Intuitively, restricting cuts to $\Sigma CNF(2)$ formulas is understandable since evaluating those formulas is complete for L, but why should we restrict the free variables in the cut formulas? As it turns out, without this restriction, the proof system corresponds to P, which we now prove.

At a high level, if we allow non-parameter variables in the cut formulas, we are able to prove there exists an output to a given circuit by repeated nesting of variables in cut formulas. This is a problem since determining the output of a circuit is P-complete.

Let H^* be the proof system G^* with cuts restricted to $\Sigma CNF(2)$ formulas and no restriction on the free variables. We will show that H^* p-simulates G_1^*. This means that, unless $L = P$, H^* does not correspond to L. This is because the G_1^* and, by this result, the H^* witnessing problems are P-complete.

Definition 4.3.2. An extension cedent Λ is a sequence of formulas

$$\Lambda =_{syn} y_1 \iff B_1, y_2 \iff B_2, \ldots, y_n \iff B_n \tag{4.3.1}$$

where B_i is a Σ_0^q formula that does not mention any of the variables $y_i, \ldots, y_n$. We call the variables $y_1, \ldots, y_n$ extension variables.

Based on a lemma in [11], Cook proved the following lemma in [6].

Lemma 4.3.3. *If π is a G_1^* proof of $\exists \vec{z} A(\vec{z}, \vec{x})$, then there exists a PK proof π' of*

$$\Lambda \rightarrow A(\vec{y}, \vec{x})$$

where Λ is as in 4.3.1 and $|\pi'| \leq p(|\pi|)$, for some polynomial p.

The proof guaranteed by this lemma is also an H^* proof since every PK proof is also an H^* proof. Extending this proof with a number of applications of $\exists$-right, we get an H^* proof of

$$\Lambda \rightarrow \exists \vec{z}, A(\vec{z}, \vec{x}). \tag{4.3.2}$$

So now we need to find a way to remove the extension cedent Λ. This is done one formula at a time. Suppose $y \iff B$ is the last formula in Λ. We will show that there are small H^* proofs of a sequent of the form

$$C_1(r_1), \ldots, C_m(r_m), C_{m+1}(y) \implies [y \iff B], \tag{4.3.3}$$

where $\exists z_i, C_i(z_i)$ is a $\Sigma CNF(2)$ formula and C_i is equivalent to $r_i \iff D_i$ where D_i does not mention $r_i, \ldots, r_m$ or y. The variable r_i is meant to refer to a subformula of B. Note that the left side of the sequent is equivalent to an extension cedent. To make things simpler, we will refer to C_i by the equivalent formula $r_i \iff D_i$.

Lemma 4.3.4. *H^* has proofs of sequents of the form 4.3.3 where the size of the proof is bounded by the size of B.*

Proof. The proof is done by structural induction on B. For the base case, suppose B is atomic. Then $B =_{syn} y'$ for some variable y'. There are small H^* proofs of

$$y \iff y' \rightarrow y \iff y'$$

Since $\exists z, (\neg z \vee y') \wedge (z \vee \neg y')$ is $\Sigma CNF(2)$, we have completed the base case.

For the inductive step, suppose $B =_{syn} C \wedge D$. Then there are proofs of

$$\Lambda_1 \rightarrow y_1 \iff C$$

and

$$\Lambda_2 \to y_2 \iff D.$$

Since the proofs are treelike, we can assume the extension variables of Λ_1 and Λ_2 are disjoint. We can easily combine these proofs to get a proof of

$$\Lambda_1, \Lambda_2, y \iff y_1 \wedge y_2 \to y \iff B.$$

Since the proof does not require any quantifiers, it is easy to see the proof is a valid H^* proof. Looking at

$$\exists y, y \iff y_1 \wedge y_2,$$

we see this formula is equivalent to

$$\exists y, (\neg y \vee y_1) \wedge (\neg y \vee \neg y_1 \vee y_2) \wedge (y \vee \neg y_1 \vee \neg y_2)$$

which is a $\Sigma CNF(2)$ formula.

When $B =_{syn} C \vee D$ or $B =_{syn} \neg C$, the proof is similar. Looking at the construction, it is easy to see the size of the H^* proof is bounded by a polynomial. $\square$

From this lemma and the sequent 4.3.2, we are able to prove the following:

$$\Lambda_1, \ldots, \Lambda_n \to \exists \vec{z}, A(\vec{z}, \vec{x}), \tag{4.3.4}$$

where each Λ_i is the extension cedent from the lemma for each formula in Λ.

The final step is to cut the Λ_is. This is easy. Start with the last formula of Λ_n. Let that formula be $y \iff D$. We apply $\exists$-left with y as the eigenvariable. The variable restriction is met by the definition of an extension cedent. Then we can cut that formula since we can prove $\exists z, z \iff D$ from $D \iff D$.

This proves the following theorem.

Theorem 4.3.5. *H^* p-simulates G_1^**

Chapter 5

Connecting L, VL, and GL^*

We have already shown the connection between L and VL (see Corollary 3.2.6). In this section, we demonstrate the connection between VL and GL^*. Then, as a corollary, we will show the connection between L and GL^*.

To connect VL and GL^*, two things must be shown. To show that our proof system is powerful enough, we show that Σ_1^B theorems of VL can be translated into a family of tautologies that have polynomial size GL^* proofs; that is, GL^* simulates VL. This is similar to the translation done in [8]. As a corollary, we can then show that the GL^* witnessing problem is L-hard. We must also show that GL^* is not too strong. This involves showing that VL proves GL^* is sound. Then as a corollary, the GL^* witnessing problem is in L. The soundness proof will be left to a later time.

5.1 Propositional Translations

We begin by proving the translation theorem. The translation that we use is described in [7, 8]. It is a modification of the Paris-Wilkie translation (see [7]). 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 $||\phi(\vec{x}, \vec{X})||[\vec{n}]$, where the size of the formulas is bounded by a polynomial in $\vec{n}$ and the value given to $\vec{x}$.

The first step is to substitute constant values for all the number variables $\vec{x}$. For now we assume ϕ has no free number variables. The formula $||\phi(\vec{X})||[\vec{n}]$ is meant to be a formula that says $\phi(\vec{X})$ is true whenever $|X_i| = n_i$ and the number variables are equal to the constants that replaced them. Then if $\phi(\vec{X})$ is true for all $\vec{X}$, then every $||\phi(\vec{X})||[\vec{n}]$ is a tautology. Note that any term t that appears in ϕ can be evaluated immediately. This is because there are no number variables and the size of each string variable is known. So we will let $val(t(\vec{n}))$ be value of the term. The variables $\vec{n}$ 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 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 0 and 1, for terms s and t. In the first case, we define $||\phi(\vec{X})||[\vec{n}]$ as the formula 1, if $val(s) = val(t)$, and 0, otherwise. A similar construction is done for $s < t$. If ϕ is one of the trivial formulas, then $||\phi(\vec{X})||[\vec{n}]$ is the same trivial formula. So now, if $\phi(\vec{X}) =_{syn} X_i(t)$, then let $j = val(t)$. Then the translation is defined as follows:

$$||\phi(\vec{X})||[\vec{n}] =_{syn} \begin{cases} p_j^{X_i} & \text{if } j < n_i - 1 \\ 1 & \text{if } j = n_1 - 1 \\ 0 & \text{if } j > n_1 - 1 \end{cases}$$

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

$$||\phi(\vec{X})||[\vec{n}] =_{syn} ||\alpha(\vec{X})||[\vec{n}] \wedge ||\beta(\vec{X})||[\vec{n}].$$

When the connective is $\vee$, or $\neg$, the definition is similar. Let $j = val(t)$. If the outer most connective is a quantifier, then the translation is defined as

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

This completes the definition.

The standard method of proving translation theorems is to translate a proof, where every cut is an instance of an axiom, one formula at a time. We will use this method, but there are a few complications. We need to be sure cut formulas in the VL proof translate into cut formulas in the appropriate form. This means that axioms must translate to $\Sigma CNF(2)$ formulas and that free string variables in cut formulas must be parameter variables. These conditions are considered in next two sections.

5.1.1 The Theory VL'

We want to reformulate the axioms so they translate into $\Sigma CNF(2)$ formulas. The main issue is the $X - rec$ axiom. At a high level, we will show how to reduce the path problem defined by that axiom to a $SAT(2)$ problem (see section 4.2). Since $SAT(2)$ is L-complete, we know this is possible.

Now to look at the axioms. The Σ_0^B axioms translate to Σ_0^q formulas, which are $\Sigma CNF(2)$. That leaves $\Sigma_0^B - COMP$ and $\Sigma_0^B - rec$. We are not going to worry about $\Sigma_0^B - COMP$; we will handle this axiom the same way Cook and Morioka did in [8]. That is, if the proof system is asked to cut the translation of an instance the $\Sigma_0^B - COMP$ axiom, $\exists X \forall i < t, X(i) \iff \phi(i)$, then that cut becomes $\bigwedge_{i=0}^{t}[||\phi(i)|| \iff ||\phi(i)||]$, which is $\Sigma CNF(2)$. So now we consider $\Sigma_0^B - rec$.

Let Σ_0^B-edge-rec be the axiom scheme

$$\exists Z \leq \langle b, a, a\rangle[\rho_1 \wedge \rho_2 \wedge \rho_3 \wedge \rho_4 \wedge \rho_5 \wedge \rho_6 \wedge \rho_7],$$

where

$$\rho_1 =_{syn} \forall j < a, \neg Z(0,0,j) \vee \phi(0,j) \vee \exists l < j\phi(0,l))$$

$$\rho_2 =_{syn} \forall j \leq a \forall k < j, \neg Z(0,0,j) \vee \neg\phi(0,k) \vee \exists l < k\phi(0,l))$$

$$\rho_3 =_{syn} \forall i \leq a \forall j \leq a, i = 0 \vee \neg Z(0,i,j)$$

$$\rho_4 =_{syn} \forall w < b \forall i \leq a \forall j \leq a, \neg Z(w+1,i,j) \vee \exists h \leq aZ(w,h,i) \vee \neg\phi(i,j) \vee \exists l < j\phi(i,l)$$

$$\rho_5 =_{syn} \forall w < b \forall i \leq a \forall j < a, \neg Z(w+1,i,j) \vee \phi(i,j) \vee \exists l < j\phi(i,l)$$

$$\rho_6 =_{syn} \forall w < b \forall i \leq a \forall j \leq a \forall k < j, \neg Z(w+1,i,j) \vee \neg\phi(i,k) \vee \exists l < k\phi(i,l)$$

$$\rho_7 =_{syn} \exists i \leq a \exists j \leq a, Z(b,i,j),$$

where $\phi(i,j)$ is a Σ_0^B formula that does not mention Z, but may have other free variables. Informally this axiom says there exists a string Z that gives a pseudo-path of length b in the graph with a nodes and edge relation $\phi(i,j)$. This path starts at node 0. If (i,j) is an edge in this path, then j is the smallest number with an edge from i to j, or $j = a$ when there are no outgoing edges. Note that the edge may not exist in the original graph when $j = a$. This is why we call is a pseudo-path. If (i,j) is the wth edge in the path, then $Z(w,i,j)$ is true, and $Z(w,i',j')$ is false for every other pair. It is not immediately obvious the axiom says this, so we will look at it closer.

Let Z be a string that witnesses the axiom. We want to make sure Z is the path described above. Looking at ρ_3, we see the path starts at 0. Suppose $Z(0,0,j)$ is true. We must show that j is the first node adjacent to 0. This follows from ρ_1, which guarantees $X(i,j)$ is true when $j < a$, and ρ_2, which guarantees $X(i,k)$ is false when $k < j$. A similar argument can be made with ρ_5 and ρ_6 to show that every node is the smallest node adjacent to its predecessor. To make sure the path is long enough, we have ρ_7, which says there is a bth edge, and ρ_4, which says if there is a $(w+1)$th edge there is a wth. As you may have noticed, there are parts of this formula that semantically are not needed. For example, the $\exists l < j\phi(0,l)$ in ρ_1 is not needed. It will be used later when we show the axiom translates into a $\Sigma CNF(2)$ formula.

Notation. For simplicity, ψ_ϕ is the string-quantifier-free part of the axiom instantiated with ϕ. Note this includes the bound on Z. So the axiom can be written as $\exists Z\psi_\phi$.

So now we define the theory VL'.

Definition 5.1.1. VL' is the theory axiomatized by the 2-BASIC, $\Sigma_0^B - COMP$, and Σ_0^B-edge-rec axioms.

We want to prove VL' is equivalent to VL. Then after we prove the translation theorem with VL', the translation theorem for VL will follow.

Lemma 5.1.2. *VL is equivalent to VL'.*

Proof. To prove the two theories equivalent, we must show that VL proves the Σ_0^B-edge-rec axiom and that VL' proves the X-rec axiom.

To show that VL prove the $\Sigma_0^B - edge - rec$ axiom, recall the *path* function (see 3.2.3). The string Z, defined as

$$Z(w,i,j) \iff path(a,w,0,X) = i \wedge path(a,w+1,0,X) = j,$$

witnesses the $\Sigma_0^B - edge - rec$ axiom for ϕ when $X(i,j) \iff \phi(i,j)$. Since VL proves there exists an output to path, it also proves Z exists.

To show that VL' proves the X-rec axiom, suppose $\forall i \leq a \exists j \leq a, X(i,j)$. By the Σ_0^B-edge-rec axiom, there is a pseudo-path of length b in the graph X. We need to show that this is a real path. Suppose (i,j) is an edge in the path. If $j < a$, then (i,j) is in the graph by ρ_1 and ρ_5. Otherwise, $j = a$, and $\forall k < j \neg X(i,k)$. Then by the assumption $\forall i \leq a \exists j \leq a, X(i,j)$, this implies $X(i,j)$. This means every edge in the path exists, and there exists a path of length b. $\square$

The next step is to be sure the translation of this axiom is a $\Sigma CNF(2)$ formula.

Lemma 5.1.3. *The formula* $||\exists Z \psi_\phi(a,b,Z)||$ *is a* $\Sigma CNF(2)$ *formula.*

Proof. First we assume $\phi(i,j) =_{syn} X(i,j)$ for some variable X. It is easy to see that $||\psi_{X(i,j)}(a,b,Z)||[t, a*a]$, where t is the bound on Z given in the $\Sigma_0^B - edge - rec$ axiom, is a CNF formula. Note that we assigned $|Z| = t$ and $|X| = a*a$. We now need to make sure the clauses have the correct form. This is done by examining each occurrence of a bound literal. To verify this, the proof will require a careful inspection of the definition of the axiom.

The only bound variables are those that come from Z. These are $z_{w,i,j}$. The only free variables are those corresponding to X. These variables will be $x_{i,j}$.

We will first look at the positive occurrences of $z_{w,i,j}$. On inspection, we can observe that, when $w < b$, every occurrence of $z_{w,i,j}$ must be in clauses that are part of the translation of ρ_4. So we want to show that every clause that is part of the translation of ρ_4 has a conflicting free variables. This is true since $\neg\phi(i,j_1)$ will conflict with one of the variables from $\exists l < j_2, \phi(i,l)$ when $j_1 < j_2$. When $w = b$, the variable $z_{b,i,j}$ appears once in ρ_7.

Now we turn to the negative occurrences. When $w = 0$, the variable $z_{0,i,j}$ will appear negatively in the clauses corresponding to ρ_1, ρ_2, and ρ_3. If $i > 0$, it will appear only in the clauses corresponding to ρ_3 and will appear only once. If $i = 0$, the variable $z_{0,0,j}$ will not appear in the translation of ρ_3 because the $i = 0$ part will satisfy the clause. It is easy to observe that every occurrence of the variable in the translation of ρ_1 and ρ_2 will have a conflicting free variable. Simply examine the construction $\phi(0,j) \vee \exists l < j\phi(0,l)$ at the end of ρ_1 and $\neg\phi(0,k) \vee \exists l < k\phi(0,l)$ at the end of ρ_2. A similar argument can be made with ρ_4, ρ_5, and ρ_6 when $w > 0$.

This implies that the translation is a $\Sigma CNF(2)$ formula when $\phi(i,j) =_{syn} X(i,j)$. When ϕ is a more general formula, the translation is the formula in the first case with the free variables substituted with the translation of ϕ, which will be Σ_0^q. Since $\Sigma CNF(2)$ formulas are closed under this type of substitution, the formula is $\Sigma CNF(2)$ in all cases.

□

5.1.2 Cut Variable Normal Form

In this section, we want to find a normal form for VL' proofs. The normal form will be used to show the connection between VL' and GL^*.

The normal form we want is *cut variable normal form* and is defined as follows.

Definition 5.1.4. A formula $\phi(Y)$ is bit-dependent on Y if there is an atomic sub-formula of ϕ of the form $Y(t)$, for some term t.

Definition 5.1.5. A proof is in free variable normal form if every non-parameter free variable y or Y that appears in the proof is used as an eigenvariable of an inference exactly once, and every formula that contains y or Y appears before the inference.

Definition 5.1.6. A cut in a proof is anchored if the cut formula is an axiom.

Definition 5.1.7. A VL' proof π is in *cut variable normal form if π is (1) in free variable normal form, (2) every cut with a non-Σ_0^B cut formula is anchored, and (3) no cut formula that is an instance of the $\Sigma_0^B - edge - rec$ axiom is bit-dependent on a non-parameter free string variable.*

It is known how to find a proof with the first two properties [6, 3], but, to my knowledge, no property similar to the third has ever been considered.

The main theorem of this section is

Figure 5.1: Undesired form for a cut.

$$
\dfrac{\begin{array}{c} P \\ \vdots \\ \exists Z\psi_{\phi(Y)}(t_a, t_b, Z), \Gamma(Y) \rightarrow \Delta(Y) \end{array} \qquad \begin{array}{c} \vdots \\ \Gamma(Y) \rightarrow \Delta(Y), \exists Z\psi_{\phi(Y)}(t_a, t_b, Z) \end{array}}{\Gamma(Y) \rightarrow \Delta(Y)}
$$

$$
\vdots
$$

$$
\dfrac{\gamma(Y), \Gamma' \rightarrow \Delta'}{\exists Y\gamma(Y), \Gamma' \rightarrow \Delta'}
$$

$$
\vdots
$$

Theorem 5.1.8. *Suppose $VL' \vdash \exists Z \leq t\phi(Z)$ from some Σ_0^B formula ϕ. Then there exists a VL'-proof π of $\exists Z \leq t\phi(Z)$ such that π is in cut variable normal form.*

We start with an anchored proof π that is in free variable normal form, and we want to put it in cut variable normal form. We replace every cut formula $\exists Z\psi_{\phi(Y)}(t_a, t_b, Z)$, where Y is a non-parameter variable, with $\exists Z\psi_{\phi'}(Z)$, where ϕ' is bit-independent of Y.

Since π is in free variable normal form, we know the variable Y must be used as the eigenvariable in an $\exists$-string-left rule. This means the proof looks like the proof in Figure 5.1, where $\gamma(Y)$ is a descendant of at least one formula in $\Gamma(Y) \cup \Delta(Y)$. Also, the $\exists Y\gamma(Y)$ must an ancestor of a cut formula since it is not a sub-formula of the final formula; to get the formula on the right of a sequent you would have to introduce a $\neg$ in front. Because $\exists Y\gamma(Y)$ is not Σ_0^B, it must be an instance of $\Sigma_0^B - COMP$ or $\Sigma_0^B - edge - rec$; otherwise the cut would not be anchored. The proof divides into two cases: one for each axiom.

We begin by assuming $\exists Y\gamma(Y)$ in an instance of $\Sigma_0^B - COMP$. That is

$$
\gamma(Y) =_{syn} |Y| \leq t \wedge \forall i < t[Y(i) \iff \phi_2(i)].
$$

In this case, the proof tells us that Y is equivalent to a Σ_0^B formula ϕ_2 with t as the upper bound on the size of Y. So the formula we are looking for will be ϕ with Y replaced by its Σ_0^B definition. This can be done as follows.

Definition 5.1.9. Let $\phi_1(Y)$ and ϕ_2 be $\mathcal{L}_A^2$-formulas and t be some term such that the free variables of t and ϕ_2, except i, are not bound in ϕ_1. Then $\phi_1^{Y,\phi_2,t}$ is defined to be ϕ_1 with every atomic formula of the form $Y(s)$ replaced by $s < t \wedge \phi_2(s)$, where the free variables of s are not bound in ϕ_2.

Note that $\phi_1^{Y,\phi_2,t}$ is no longer bit-dependent on Y. With this definition we can prove the following lemma and corollary.

Lemma 5.1.10. *V^0 proves*

$$[|Y| \leq t \wedge \forall i < t(Y(i) \iff \phi_2(i))] \implies [\phi_1(Y) \iff \phi_1^{Y,\phi_2,t}].$$

Proof. The proof is done by structural induction on ϕ_1. Most cases are easy. We will only look at the quantifier case. Say we have a formula $\phi_1(x, Y)$, and we want to prove the lemma for $\exists x \phi_1(x, Y)$. By the variable restriction in the definition above, we can assume x is not free in ϕ_2 and t. That means

$$\exists x[\phi_1^{Y,\phi_2,t}] =_{syn} [\exists x \phi_1]^{Y,\phi_2,t},$$

and the lemma follows. □

Corollary 5.1.11. *V^0 proves*

$$[|Y| \leq t \wedge \forall i < t(Y(i) \iff \phi_2(i))] \implies [\psi_{\phi_1(Y)}(Z) \iff \psi_{\phi_1^{Y,\phi_2,t}}(Z)],$$

with Z distinct from Y.

Proof. This is simply a special case of the previous lemma since $\psi_{\phi_1^{Y,\phi_2,t}}(Z) =_{syn} [\psi_{\phi_1(Y)}(Z)]^{Y,\phi_2,t}$. □

So now we are equipped to modify π to remove the undesirable cut. For simplicity, we assume the proof has the form

$$\dfrac{\overset{P}{\vdots}\quad \exists Z\psi_{\phi(Y)}(Z), \gamma(Y), \Gamma \to \Delta \qquad \overset{\vdots}{\gamma(Y), \Gamma \to \Delta, \exists Z\psi_{\phi(Y)}(Z)}}{\gamma(Y), \Gamma \to \Delta}$$

We can assume this since we can weaken to get the $\gamma(Y)$, and then contract it later. We also assume all the variable restrictions in the definition of $\phi_1^{Y,\phi_2,t}$ are met. If they are not, a simple renaming of variables will fix the problem.

We now want to find a new proof of $\gamma(Y), \Gamma \to \Delta$ in which one less Σ_0^B-edge-rec cut is bit-dependent on Y. The first step is to change the proof P into a proof P' of the sequent

$$\psi_{\phi(Y)}(Y^*), \gamma(Y), \Gamma \to \Delta$$

where Y^* is a new string variable that does not appear anywhere in π. This can be done by omitting every $\exists$-left rule that introduced the $\exists Z$ part, and by renaming the eigenvariable to Y^*. We can do this since the proof is in free variable normal form.

Let ϕ' be $\phi^{Y,\phi_2,t}$. By corollary 5.1.11, there is an anchored V^0-proof Q of

$$\gamma(Y), \psi_{\phi'}(Y^*) \to \psi_{\phi(Y)}(Y^*)$$

that is in free variable normal form. Since $\Sigma_0^B - edge - rec$ is not in V^0, but all of the other axioms of VL' are in V^0, we know Q is also a VL'-proof in cut variable normal form. The section of the proof now becomes

$$\dfrac{\dfrac{\overset{P'}{\vdots}\quad \psi_{\phi(Y)}(Y^*), \gamma(Y), \Gamma \to \Delta \qquad \overset{Q}{\vdots}\quad \gamma(Y), \psi_{\phi'}(Y^*) \to \psi_{\phi(Y)}(Y^*)}{\psi_{\phi'}(Y^*), \gamma(Y), \Gamma \to \Delta}}{\exists Z\psi_{\phi'}(Z), \gamma(Y), \Gamma \to \Delta}$$

Then we can cut $\exists Z\psi_{\phi'}(Z)$ since it is an axiom. That gives us a new proof that is in still in free variable normal form, all cuts with a non-Σ_0^B cut formula are anchored, and one less $\Sigma_0^B - edge - rec$ cut formula is bit-dependent on Y.

Moving on to the second case, we now assume $\exists Y\gamma(Y)$ is an instance of Σ_0^B-edge-rec; that is, $\gamma(Y) =_{syn} \psi_{\phi_2}(s_a, s_b, Y)$ for some Σ_0^B formula ϕ_2 and terms s_a and s_b. So now we want to transform the proof as we did in the first case. However the construction is more complicated. We will have to define a new graph in which a path can be transformed into a path in the graph of ϕ. The high level idea is to define a graph where a path in this graph represents an execution in a branching program that finds the path. For the rest of this section we will work with the following definition of a branching program.

Definition 5.1.12. A branching program is a nonempty set of nodes labeled with triples (α, i, j), where α is a Σ_0^B formula over some set of variables and $0 \leq i, j \leq t$ for some term t that depends only on the inputs to the program. Semantically, if a node u is labeled with (α, i, j), then, when the branching program is at node u, it will go to node i, if α is true, or node j, otherwise. The initial node is always 0.

This definition is different from the standard definition of a branching program: there are no output nodes and a Σ_0^B formula is evaluated at each node, but, for our purpose, this is alright.

Given a branching program BP where the formulas are over the variables $(\vec{x}, \vec{X})$, we want to define a formula $\phi_{BP}(u, v, \vec{x}, \vec{X})$ that is true if BP goes from node u to node v on the given input. This can be difficult since the programs are not necessarily finite, but the programs we use can be easily turned into an appropriate formula. We will refer to this formula as BP as well.

We are going to define a sequence of branching programs $BP_0, \ldots, BP_n$. Then, using BP_n, we will construct a final branching program that is bit-independent of Y. This final branching program can be used to find Y^*, a path in the graph of $\phi(i, j, Y)$, assuming Y satisfies $\psi_{\phi_2}(s_a, s_b, Y)$.

The first step is to introduce the initial branching program BP_0. The nodes of BP_0 are triples $\langle w, i, j\rangle$. If we reach a particular node $\langle w, i, j\rangle$, then we know the wth node in Y^* is i and $\forall k < j \neg\phi(i, k)$. So as far as we know, j could be the next node. Recall that t_a is the maximum value of a node and t_b is the length of the path. This means the number of nodes in BP_0 is bound by $\langle t_b, t_a, t_a\rangle$. So now to define the labels. If $j < t_a$, then $\langle w, i, j\rangle$ is labeled with $(\phi(i, j), \langle w + 1, j, 0\rangle, \langle w, i, j + 1\rangle)$. If $j = t_a$, then $\langle w, i, j\rangle$ is labeled with $(T, \langle w + 1, j, 0\rangle, 0)$. It is easy to see that the invariants hold and that Y^* can be obtained from a path in BP_0.

Moving on to the second step, we now want to simplify the branching program so that every node whose label is bit-dependent on Y is labeled with an atomic formula. We start with BP_0. Then, given BP_i, we define BP_{i+1} by removing one connective in a node of BP_i that is not in the right form. Let node n in BP_i be labeled with (α, u_1, u_2). The construction is divided into five cases: one for each possible outer connective.

If $\alpha =_{syn} \neg\beta$ then BP_{i+1} is the same as BP_i except node n is now labeled with (β, u_2, u_1).

If $\alpha =_{syn} \beta_1 \wedge \beta_2$, we first rename the nodes. The nodes of BP_{i+1} are interpreted as pairs $\langle u, v\rangle$. So the node $\langle u, 0\rangle$ corresponds to node u in BP_i. The label of $\langle n, 0\rangle$ becomes $(\beta_1, \langle n, 1\rangle, \langle u_2, 0\rangle)$ and the label for $\langle n, 1\rangle$ is $(\beta_2, \langle u_1, 0\rangle, \langle u_2, 0\rangle)$. Notice that $\langle n, 1\rangle$ is used as an intermediate node while evaluating α.

When $\alpha =_{syn} \beta_1 \vee \beta_2$, BP_{i+1} is defined as in the previous case, with a few minor modifications. This case is left to the reader.

Now suppose $\alpha =_{syn} \exists z \leq t\beta(z)$. The nodes become pairs as in the previous case, but this time the labels are different. The node $\langle n, i\rangle$ is labeled with $(\beta(i), \langle u_1, 0\rangle, \langle n, i+1\rangle)$, when $i < t$. Otherwise, the node is labeled with $(\beta(i), \langle u_1, 0\rangle, \langle u_2, 0\rangle)$. In this case, the branching program is looking for an i that satisfies $\beta(i)$. When $\alpha =_{syn} \forall z \leq t\beta(z)$, BP_{i+1} is constructed in a similar way. The only difference is the branching program is looking for an i that falsifies $\beta(i)$.

Let BP_n be the final branching program.

We now move on to the third and final step. We want to show how to find a branching program BP' that is bit-independent of Y, such that given a path in BP', we are able to find a path in BP_n, assuming Y satisfies $\psi_{\phi_2}(s_a, s_b, Y)$. Then we can get Y^* from the path in BP_n

To construct BP', we will show how to change the label on each node u of $BP(Y)$. The nodes in BP' are quadruples $\langle u, w, i, j\rangle$. The three extra values are used to compute Y the same way BP_0 computed Y^*. Recall Y is a path in the graph of ϕ_2. Let u be a node in BP_n that is labeled with (α, u_1, u_2). The label of $\langle u, w, i, j\rangle$ is divided into two cases.

First suppose α is bit-independent of Y. Then the label is $(\alpha, \langle u_1, 0, 0, 0\rangle, \langle u_2, 0, 0, 0\rangle)$. There is no need to compute Y.

For the second case, suppose α is bit-dependent on Y. Then $\alpha =_{syn} Y(w', i', j')$. In this case, $\langle u, w', i, j\rangle$ is labeled with

$$(w' \leq s_b \wedge i' = i \wedge j' = j, \langle u_1, 0, 0, 0\rangle, \langle u_2, 0, 0, 0\rangle),$$

and when $w < w'$, $\langle u, w, i, j\rangle$ is labeled with

$$(\phi_2(i, j) \vee j = s_a, \langle u, w+1, j, 0\rangle, \langle u, w, i, j+1, 0\rangle).$$

Note that BP' computes the path Y then decides if the edge in question is part of Y. This makes it fairly easy to see that we can extract a path in BP_n from a path in BP'.

Using these transformations, we are able to prove the following lemma.

Lemma 5.1.13. *Suppose that Y satisfies $\psi_{\phi_2}(s_a, s_b, Y)$. Then, for every formula $\phi(Y)$ and ϕ_2, V^0 proves that*

$$\psi_{BF'}(Z') \wedge \forall i \leq t[Z(i) \iff \phi_3(i, Z')] \implies \psi_{\phi(Y)}(t_a, t_b, Z), \tag{5.1.1}$$

for some Σ_0^B formulas ϕ_3 and BP'.

Proof. The construction of BP' is given above. The formula ϕ_3 transforms a path in BP' into a path of $\phi(Y)$, when Y satisfies $\psi_{\phi_2}(s_a, s_b, Y)$. Looking at the transformations of the path above, we can see they can be done in AC^0. This means there is a Σ_0^B formula that describes this transformation: this formula is ϕ_3. Also, it is easy to see V^0 proves the properties of this function. $\square$

We now need to change the proof as we did in the previous case. Recall the form of the proof (see Figure 5.1). As before, we begin by changing P to a proof of

$$\psi_{\phi(Y)}(Y^*), \gamma(Y), \Gamma \to \Delta.$$

This time we let Q be a proof of Formula 5.1.1. Then we change the proof to the following proof.

$$\dfrac{\dfrac{\begin{matrix} P' \\ \vdots \\ \psi_{\phi(Y)}(t_a, t_b, Y^*), \gamma(Y), \Gamma \to \Delta \end{matrix} \qquad \begin{matrix} Q \\ \vdots \\ \gamma(Y), \psi_{BF'}(Z'), \varepsilon \to \psi_{\phi(Y)}(t_a, t_b, Y^*) \end{matrix}}{\gamma(Y), \psi_{BF'}(Z'), \varepsilon, \Gamma \to \Delta}}{\gamma(Y), \psi_{BF'}(Z'), \exists Y^* \varepsilon, \Gamma \to \Delta}$$

where ε is the formula $|Y^*| \leq t \wedge \forall i \leq t[Y^*(i) \iff \phi_3(i, Z')]$. Note that $\exists Y^*\varepsilon$ and $\exists Z'\psi_{BF'}(Z')$ are instances of axioms; so we are allowed to cut them. The proof continues as follows:

$$\dfrac{\dfrac{\dfrac{\gamma(Y), \psi_{BF'}(Z'), \exists Y^*\varepsilon, \Gamma \to \Delta \qquad \to \exists Y^*\varepsilon}{\gamma(Y), \psi_{BF'}(Z'), \Gamma \to \Delta}}{\gamma(Y), \exists Z'\psi_{BF'}(Z'), \Gamma \to \Delta} \qquad \to \exists Z'\psi_{BF'}(Z')}{\gamma(Y), \Gamma \to \Delta}$$

Observe that the Σ_1^B cut formulas are anchored and that the proof is still in free variable normal form. As well, Y is involved in one less cut that violates the cut variable normal form conditions.

We are now prepared to prove theorem 5.1.8.

Proof of Theorem 5.1.8. It would be nice to be able to simply say we can repeatedly apply the manipulations above and eventually the proof will be in cut variable normal form, but this is not obvious. In both manipulations, if $\gamma(Y)$ is bit-dependent on a string variable other than Y, then the new Σ_0^B-edge-rec cut formula is bit-dependent on that variable. This includes non-parameter string variables. So we need to state our induction hypothesis more carefully.

Let $Y_1, \ldots, Y_n$ be all the non-parameter free string variables that appear in π ordered such that when $i < j$, the variable Y_i is used as a eigenvariable before Y_j. This means Y_i does not appear in $\gamma(Y_j)$ in the manipulations above. So now suppose no Σ_0^B-edge-rec cut formula is bit-dependent on the variables $Y_1, \ldots, Y_k$, for some $k < n$. Then we can manipulate π such that the same holds for the variables $Y_1, \ldots, Y_{k+1}$. To accomplish this, we simply manipulate every Σ_0^B-edge-rec cut formula that is bit-dependent on Y_{k+1} as described above. Since $Y_1, \ldots, Y_k$ cannot appear in the $\gamma(Y_k)$ formulas, those variables will not violate the condition. So by induction, we can get a proof that is in cut variables normal form. $\square$

5.1.3 Translating Theorems of VL

We are now prepared to prove the translation theorem. As has already been mentioned, the proof is done by induction on the length of the proof. For the base case, we need to prove the translation of the axioms of VL'. Since every axiom besides Σ_0^B-edge-rec is an axiom of VNC^1, we know those axioms have polynomial size proofs in G_0^* and, therefore, in GL^* as well. We still need to show how to prove the Σ_0^B-edge-rec axiom in GL^*. Recall that we write the axiom as $\exists Z \psi_\phi(a, b, Z)$. Note that the axiom does have a bound on Z, which has been omitted since the specific bound is not important.

Lemma 5.1.14. *The formula $||\exists Z \psi_\phi(a, b, Z)||$ has a GL^* proof of size $p(a, b)$ for some polynomial p.*

Proof. The proof is done by a brute force induction. We prove, in GL^*, that if there is a pseudo-path of length b, then there exists a pseudo-path of length $b+1$.

The idea is to find Σ_0^q formulas that can be used to determine the next edge. We need to show that GL^* prove exactly one edge is chosen. Let $A_{i,j} =_{syn} ||\phi(i,j)||$. Since ϕ is a Σ_0^B formula, $A_{i,j}$ is a Σ_0^q formula. To prove there is an edge that starts the path, consider the formula

$$B_{0,0,j} =_{syn} A_{0,j} \wedge \bigwedge_{k=0}^{j-1} \neg A_{0,k},$$

when $j < a$, and

$$B_{0,0,a} =_{syn} \bigwedge_{k=0}^{a-1} \neg A_{0,k}.$$

It is easy to see $B_{0,0,j}$ is true for at least one $j \leq a$. This is also provable in GL^*. This shows that GL^* has a polynomial size proof of

$$||\exists Z \psi_\phi(a, 1, Z)||.$$

For the inductive step, there is a path of length b and the path is given by the variables $z_{w,i,j}$. Then the witnesses are defined as follows:

$$B_{b+1,i,j} =_{syn} \bigvee_{k=0}^{a} z_{b,k,i} \wedge A_{i,j} \wedge \bigwedge_{k=0}^{j-1} \neg A_{i,k},$$

when $j < a$, and

$$B_{b+1,i,a} =_{syn} \bigvee_{k=0}^{a} z_{b,k,i} \wedge \bigwedge_{k=0}^{a-1} \neg A_{i,k}.$$

Using the fact that exactly one $z_{b,i,j}$ is true, we can prove in GL^* that exactly one $B_{b+1,i,j}$ is true. This shows that GL^* has a polynomial size proof of

$$||\exists Z \psi_\phi(a, b, Z)|| \to ||\exists Z \psi_\phi(a, b+1, Z)||.$$

So now we are able to prove $||\exists Z \psi_\phi(a, b, Z)||$ for any b by successive cutting. Recall that $||\exists Z\psi_\phi(a, b, Z)||$ is a $\Sigma CNF(2)$ formula, and note that the free variables in $||\exists Z\psi_\phi(a, b, Z)||$ do not change as b changes. This means we are allowed to do the cut. □

We now prove the main theorem of this section.

Theorem 5.1.15. *Suppose VL proves $\exists Z < t\phi(\vec{x}, \vec{X}, Z)$. Then there are polynomial size GL^* proofs of $||\exists Z < t\phi(\vec{x}, \vec{X}, Z)||[\vec{n}]$.*

Proof. Since VL' is equivalent to VL, the formula is a theorem of VL' as well. Then, by Lemma 5.1.8, there exists a VL' proof π of $\exists Z < t\phi(\vec{x}, \vec{X}, Z)$ that is in cut variable normal form.

We proceed by induction on the depth of π. The base case follows from lemma 5.1.14 and the comments that precede it. The inductive step is divided into cases: one for each rule. With the exception of cut, every rule can be handled the same way it is handled in the $V^1 - G_1^*$ Translation Theorem [6], and will not be repeated here.

When looking at cut, there are three cases. If the cut formula is Σ_0^B, then we simply cut the corresponding Σ_0^q formulas in the GL^* proof. If the cut formula is not Σ_0^B, then it must be anchored since the proof is in cut variable normal form. This means the cut formula is an instance of Σ_0^B-edge-rec or an instance of $\Sigma_0^B - COMP$. First suppose it is an instance of Σ_0^B-edge-rec. Then we are able to cut the corresponding formulas in the GL^* proof since the axiom translates to a $\Sigma CNF(2)$ formula and, since it is not bit-dependent on a non-parameter string variable, the free variable in the translation are parameter variables.

When the cut formula is an instance of $\Sigma_0^B - COMP$, we apply the same transformation as the authors of [8]. That is, we remove the quantifier by replacing the variables with Σ_0^q formulas that witness the quantifier. This change would not effect other cut formulas since their free variables are parameter variables or they are Σ_0^q. The current cut formula becomes a Σ_0^q formula, which can be cut. □

Chapter 6

Conclusions

To summarize, the theory VL corresponds to FL in the same way other theories correspond to complexity classes. However, it might be interesting explore other definability results. For example, find which class of functions are Σ_2^B definable in VL.

We have also defined a new proof system GL^*, which is, to our knowledge, the first proof system that corresponds to a complexity class between NC^1 and P. We have shown that GL^* simulates VL, but we did not show VL proves the soundness of GL^*. This would prove GL^* does not correspond to a larger class. We know how to do this, but the final details are still being worked out.

In later work, it would be nice to explore GL, which is the proof system GL^* except that proofs do not have to be treelike. Also, we wonder if it is possible to construct a proof system for NL using the same method. That is, base the proof system on $2-CNF$ formulas instead of $CNF(2)$ formulas.

Bibliography

[1] S. R. Buss. Relating the bounded arithmetic and polynomial time hierarchies. *Annals of Pure and Applied Logic*, 75:67–77, 1995.

[2] Samuel R. Buss. Propositional consistency proofs. *Annals of Pure and Applied Logic*, 52(1-2):3–29, 1991.

[3] Samuel R. Buss. Introduction to proof theory. In Samuel R. Buss, editor, *Handbook of Proof Theory*, pages 1–78. Elsevier Science Publishers, Amsterdam, 1998.

[4] S.R. Buss. *Bounded Arithmetic.* PhD thesis, Princeton University, 1986.

[5] S.R. 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.

[6] S. Cook. Csc 2429s: Proof complexity and bounded arithmetic. Course notes. Available at *http://www.cs.toronto.edu/˜sacook/csc2429h.*

[7] S. Cook. Theories for complexity classes and their propositional translations. Available at *http://www.cs.toronto.edu/˜sacook/*, March 2004.

[8] S. Cook and T. Morioka. Quantified propositional calculus and a second-order theory for NC^1. *Accepted for Archive for Math. Logic.*

[9] Stephen Cook and Antonina Kolokolova. A second-order system for polynomial-time reasoning based on Graedel's theorem. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(24), 2001.

[10] J. Johannsen. Satisfiability problem complete for deterministic logarithmic space. In *STACS 2004, 21st Annual Symposium on Theoretical Aspects of Computer Science, Proceedings*, pages 317–325. Springer, 2004.

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

[12] J. Krajicek and P. Pudlak. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschrift f. Mathematikal Logik u. Grundlagen d. Mathematik*, 36:29–46, 1990.

[13] John C. Lind. Computing in logarithmic space. Mac Technical Memorandom 52, September 1974.

[14] Tsuyoshi Morioka. *Logical Approaches to the Complexity of Search Problems: Proof Complexity, Quantified Propositional Calculus, and Bounded Arithmetic.* PhD thesis, University Of Toronto, 2005.

[15] Phuong Nguyen. *VTC^0: A second-order theory for TC^0*. PhD thesis, University of Toronto, 2004.

[16] R. Parikh. Existence and feasibility in arithmetic. *Journal of Symbolic Logic*, 36:494–508, 1971.

[17] D. Zambella. Notes on polynomially bounded arithmetic. *The Journal of Symbolic Logic*, 61(3):942–966, 1996.

[18] D. Zambella. End extensions of models of linearly bounded arithmetic. *Annals of Pure and Applied Logic*, 88:263–277, 1997.