DIMACS Series in Discrete Mathematics
and Theoretical Computer Science

A Propositional Proof System for R_2^i

Chris Pollett

Abstract. In this paper we introduce Gentzen-style quantified propositional proof systems L_i^* for the theories R_2^i. We formalize the systems L_i^* within the bounded arithmetic theory R_2^1 and we show that for $i \geq 1$, R_2^i can prove the validity of a sequent derived by an L_i^*-proof. This statement is formally called i-$RFN(L_i^*)$. We show if $R_2^i \vdash \forall x A(x)$ where $A \in \Sigma_i^b$, then for each integer n there is a translation of the formula A into quantified propositional logic such that R_2^i proves there is an L_i^*-proof of this translated formula. Using the proofs of these two facts we show that L_i^* is in some sense the strongest system for which R_2^i can prove i-RFN and we show for $i \geq j \geq 2$ that the $\forall\Sigma_j^b$-consequences of R_2^i are finitely axiomatized.

1. Introduction

Propositional proof systems and bounded arithmetic are closely connected. Cook [**10**] introduced the equational arithmetic theory PV of polynomial time computable functions and showed PV could prove the soundness of the propositional proof system known as extended Frege or EF. Krajíček and Pudlak [**15**] developed Gentzen-style quantified propositional proof systems G_i and G_i^* for the well-studied bounded arithmetic theories T_2^i and S_2^i respectively. The theories S_2^i are of especial interest to computer scientists since their Σ_i^b-definable functions are precisely those functions computable in polynomial time with access to a Σ_{i-1}^p-oracle [**5**]. The theory S_2^1 is a conservative extension of PV [**5**]. Krajíček and Pudlak [**15**] showed S_2^i could prove certain reflection principles for the quantified propositional proof system G_i^*. They also showed that S_2^i proofs can be simulated by an infinite sequence of polynomial sized G_i^* proofs. They used these results to show the $\forall\Sigma_j^b$-consequences of S_2^i are finitely axiomatizable where $i \geq j \geq 2$. This result is of interest since if the union of the theories S_2^i, the theory S_2, is finitely axiomatized then the polynomial hierarchy collapses. More recently, Clote and Takeuti [**9**] have introduced a weaker notion than Σ_1^b-definability which they call essentially sharply bounded definable (esb-definable) and used this notion to come up with bounded arithmetic theories for various function classes within $\square_1^p$. Gaisi Takeuti [**19**] has shown that TNC^0, a theory whose *esb*-definable functions are exactly the functions in the circuit class NC^1, can prove the soundness of Frege proof systems. Further he showed translations of TNC^0 proofs into propositional logic have polynomial

1991 *Mathematics Subject Classification.* 03F20 03F30.

sized Frege proofs. Given these relationship between proof systems and bounded arithmetic, determining whether or not Frege can polynomially simulate extended Frege could be important in solving whether $NC^1 = P$.

Besides S^i_2 and T^i_2 another important sequence of bounded arithmetic theories are the theories R^i_2 developed in [**1**], [**8**], [**18**], and [**2**]. These theories are of interest to computer scientists since the Σ^b_1-definable functions of R^1_2 correspond to functions in the parallel computation class NC [**1**]. The theory R^1_2 lies between the theories TNC^0 and S^1_2 so a proof system for R^1_2 may help shed some light on how difficult it is to separate these two theories.

We now discuss the organization of this paper. In Section 2, we briefly introduce the necessary concepts from bounded arithmetic and quantified propositional logic for this paper. In Section 3, we discuss some functions and multifunctions that R^i_2 has at its disposal and also show some closure properties for the Σ^b_i-definable multifunctions and Δ^b_i-predicates of R^i_2. Section 4 describes how to formalize quantified propositional logic within R^i_2. We describe in this section a $B(\Sigma^b_i)$-formula $TRU_i(\ulcorner A \urcorner, \tau)$ which returns the truth value of the quantified propositional formula A with respect to a truth assignment τ. Section 5 explains how to translate bounded arithmetic formulas into sequences of quantified propositional formulas. In Section 6, we define what a quantified propositional proof system is and we introduce a Gentzen-style quantified propositional proof systems L^*_i for the theories R^i_2. Proofs in the systems L^*_i are tree-like, $(\log)^2$-height quantified propositional sequent calculus proofs where formulas are restricted to the class $\Sigma^q_i \cup \Pi^q_i$ and the number of $\Sigma^q_i \cup \Pi^q_i$-cuts along any branch in a proof is restricted to be less than $(\log\log)^2$ of the size of the proof. (The exponents of 2 in the above are not especially critical since one can prove the same results we do for any fixed number greater than 1.) We then formalize the systems L^*_i within the bounded arithmetic theory R^1_2. Section 7 contains a witnessing argument used to show R^i_2 can prove the validity of any Σ^q_i-formula derived by an L^*_i-proof. This statement is formally called i-$RFN(L^*_i)$. In Section 8, we show if $R^i_2 \vdash \forall x A(x)$ where $A \in \Sigma^b_i$, then for each integer n there is a translation of the formula A into quantified propositional logic such that R^i_2 proves there is an L^*_i-proof of this translated formula. This proof is similar to the simulation of T^i_2 by G_i in Krajíček and Pudlak [**15**] except that we have to be a little careful because of the various bounds on L^*_i. In particular, when we simulate sharply bounded quantifier rules we need to make sure our AND or OR rules are done in a balanced fashion; otherwise, we violate the height requirement of L^*_i-proofs. Similarly, in simulating Σ^b_i-$LLIND$ with cuts we must make sure our cuts are done in a balanced fashion. Section 9 contains applications of Section 7 and Section 8. In this section we show that L^*_i is in some sense the strongest system for which R^i_2 can prove i-RFN and we show for $i \geq j \geq 2$ that the $\forall\Sigma^b_j$-consequences of R^i_2 are finitely axiomatized.

2. Preliminaries

The theories R^i_2 for $i \geq 0$ are the first-order theories introduced by Gaisi Takeuti in [**18**]. Equivalent theories were introduced by Allen [**1**]. Clote and Takeuti [**8**] have a theory equivalent to R^1_2. The language, L_2, of R^i_2 consists of the symbols 0, S, $+$, $\dot{-}$, $\cdot$, $\lfloor \frac{1}{2}x \rfloor$, $\#$, $|x|$, MSP, $\leq$ and $=$. The intended meaning of $|x|$ is the function $\lceil \log_2(x+1) \rceil$. The intended meaning of $x\#y$ is $2^{|x||y|}$. The intended meaning $MSP(a,i)$ is $\lfloor a/2^i \rfloor$. $BASIC$ is a finite list of open axioms for these symbols. Most of these axioms can be found in Buss [**5**] p.30; however, what we will call $BASIC$ is his list of axioms extended by Takeuti [**18**]'s axioms for MSP and $\dot{-}$.

A quantifier is said to be *bounded* if it is of the form $(\exists x \leq t(\vec{a}))$ or of the form $(\forall x \leq t(\vec{a}))$ for some term t in L_2. A quantifier is sharply bounded if, in addition, t is of the form $|s|$ for some term in L_2. We write $Open$ for the set of open formulas. We now inductively define the sets Σ^b_i and Π^b_i. The set $\Sigma^b_0 = \Pi^b_0$ is the set of formulas all of whose quantifiers are sharply bounded. The set Σ^b_i is the smallest set containing Π^b_{i-1} and closed under conjunction, disjunction, sharply bounded quantification, and bounded existential quantification. The set Π^b_i is the smallest class containing Σ^b_{i-1} and closed under conjunction, disjunction, sharply bounded quantification, and bounded universal quantification.

We define several kinds of induction axioms:

$$(IND_\alpha) \qquad \alpha(0) \wedge (\forall n)(\alpha(n) \supset \alpha(S(n))) \supset (\forall n)\alpha(n)$$

$$(PIND_\alpha) \qquad \alpha(0) \wedge (\forall n)(\alpha(\lfloor \tfrac{1}{2}n \rfloor) \supset \alpha(n)) \supset (\forall n)\alpha(n)$$

$$(LIND_\alpha) \qquad \alpha(0) \wedge (\forall n)(\alpha(n) \supset \alpha(S(n))) \supset (\forall n)\alpha(|n|)$$

$$(PLIND_\alpha) \qquad \alpha(0) \wedge (\forall n)(\alpha(\lfloor \tfrac{1}{2}n \rfloor) \supset \alpha(n)) \supset (\forall n)\alpha(|n|)$$

$$(LLIND_\alpha) \qquad \alpha(0) \wedge (\forall n)(\alpha(n) \supset \alpha(S(n))) \supset (\forall n)\alpha(||n||)$$

For Ψ a set of formula we define the Ψ-IND axioms to be the axioms IND_α for $\alpha \in \Psi$. We define Ψ-$PIND$, Ψ-$LIND$, Ψ-$PLIND$, and Ψ-$LLIND$ similarly.

The theory T^i_2 is axiomatized as $BASIC$+Σ^b_i-IND, the theory S^i_2 is axiomatized as $BASIC$+Σ^b_i-$PIND$ axioms, and finally the theory R^i_2 is axiomatized as $BASIC$+Σ^b_i-$PLIND$ axioms. The theories S^i_2 and T^i_2 as introduced in [**5**] are usually formulated over a language without $\dot{-}$, or MSP; however, it turns out adding these symbols does not increase the power of these theories [**5**]. The original formulation of R^i_2 in [**18**] also had a bit extensionality axiom shown to be unnecessary in [**13**].

Recall that a theory T can Σ^b_i-define a function $f(x)$, if there is a Σ^b_i-formula $A_f(x,y)$ for which $T \vdash (\forall x)(\exists! y)A_f(x,y)$ and $\mathbb{N} \models A_f(x, f(x))$. A predicate is said to be Δ^b_i with respect to a theory T if it is provably equivalent in T to both a Σ^b_i and a Π^b_i-formula. It is a theorem of [**5**] that once we can Σ^b_1-define a function f in a theory we can add the function symbol to the theory without changing the Σ^b_i or Π^b_i -consequences of the theory $i \geq 1$. A similar result holds for adding Δ^b_1-predicate symbols [**5**].

In concluding our preliminary discussion of R^i_2 we would like to mention that the theory R^i_2 can also prove Σ^b_i-LLIND axioms, Δ^b_i-$PIND$ axioms, and Δ^b_i-$LIND$ axioms [**1**], [**18**]. So R^i_2 contains S^{i-1}_2 and one can also show that S^{i-1}_2 contains T^{i-2}_2 [**5**].

Now that we have finished introducing R^i_2 we turn to quantified propositional logic. The class of *quantified propositional formulas* is the least class containing the atoms $p_0, p_1, \ldots$, the constants $\bot$ (false) and $\top$ (true), closed under the connectives $\wedge$, $\vee$, $\supset$, and $\neg$ and for any formula $A(p)$ the class of quantified propositional formulas contains the formulas $\exists x A(x)$ and $\forall x A(x)$ where x substitutes occurrences of p in $A(p)$. The intended meaning of $\exists x A(x)$ is $A(\bot) \vee A(\top)$ and that of $\forall x A(x)$ is $A(\bot) \wedge A(\top)$. We define hierarchies Σ^q_i and Π^q_i of propositional formulas in analogous way to the first-order case. We define Δ^q_{i+1} to mean the class of Boolean combinations of Σ^q_i-formulas.

3. Functions definable in R^i_2

In this section, we describe some of the Δ^b_1-predicates and Σ^b_1-functions available to code sequences in R^1_2. These predicates and functions will enable us to formalize the syntax of quantified propositional formulas in R^1_2 and to give a truth predicate for these formulas. We also discuss what kinds of multifunctions are available in the theories R^i_2.

Methods in R^1_2 for coding a sequence of natural numbers $\langle a_1, \dots, a_n\rangle$ as a single natural number have been described by both Allen [**1**] and Clote-Takeuti [**8**]. In particular, R^1_2 can prove the following functions and predicates which are useful in manipulating sequences to be either Σ^b_1-definable or Δ^b_1:

- $\mathit{Seq}(w)$ which is true if and only if w is a natural number coding a sequence.
- $\beta(i, w)$ which returns the ith element of the sequence w.
- $\mathit{len}(w)$ which returns the length of the sequence w.
- $\mathit{Bit}(i, a)$ which returns the ith bit of a number a.
- $\mathit{Subseq}(i, j, w)$ which returns the subsequence of the ith through the jth entry of w.
- $\mathit{Front}(w) := \mathit{Subseq}(1, \mathit{len}(w) - 1, w)$ which returns all but the last element of a sequence
- $\mathit{Tail}(w) := \mathit{Subseq}(2, \mathit{len}(w), w)$ which returns all but the first element of a sequence
- $w * a$ which if w is a sequence and a is an entry returns the concatenation of a to w.
- $w ** v$ which if w and v are sequences returns their concatenation.
- $2^{|y|}$ which equals $1\#y$
- $\mathit{cond}(x, y, z)$ which outputs y if x is 0 and outputs z otherwise. It allows us give definitions by cases.
- $\max(x, y)$ which equals $\mathit{cond}(x \dot{-} y, y, x)$
- $\min(x, y)$ which equals $\mathit{cond}(x \dot{-} y, x, y)$
- $\lceil \log_2 x \rceil := \mathit{cond}(2 \cdot x \dot{-} 2^{|x|}, |x| \dot{-} 1, |x|)$. We will frequently be lazy and write this as $\log x$.
- For A a Δ^b_1-predicate, R^1_2 can Σ^b_1-define the function $(\#x \leq |t|)A(x)$ which returns the number of values less than or equal to $|t|$ where $A(x)$ is true [**1**].
- For A a Δ^b_1-predicate, R^1_2 can Σ^b_1-define the function $(\mu x \leq |t|)A(x)$ which returns the least value $x \leq |t|$ such that $A(x)$ holds [**1**].

We now discuss the multifunctions available in the theories R^i_2. First, we make precise what we mean by a multifunction.

Definition 1. *A* multifunction *is a set $f \subseteq \mathbb{N} \times \mathbb{N}$ such that for all $x \in \mathbb{N}$ there exists $\langle x, y\rangle \in f$. We usually express $\langle x, y\rangle \in f$ as $f(x) = y$. We write $f \circ g$ for the composition of the two multifunctions f, g and define $(f \circ g)(x) = z$ if there is some $y \in \mathbb{N}$ such that $f(x) = y$ and $g(y) = z$. If f is a multifunction and r is a function, we write $f(x) > r(x)$ if there exists $y > r(x)$ such that $f(x) = y$. We define $f(x) < r(x)$ if there exists $y < r(x)$ such $f(x) = y$.*

We say a theory T can Σ^b_i-define a multifunction f if there is a Σ^b_i-formula A_f and T can prove $(\forall x)(\exists y)A_f(x,y)$ and $\mathbb{N} \models f(x) = y \Leftrightarrow A_f(x,y)$. In [**16**], it was shown for $i \geq 1$ that the Σ^b_{i+1}-definable functions of R^{i+1}_2 are precisely the multifunctions in the class $FP^{\Sigma^p_i}(wit, \log^{O(1)})$, the class of multifunctions computable in polynomial time with $O(log^{O(1)})$ many queries to a Σ^p_i-oracle which returns witnesses for the truth of its "Yes" answers. It is useful, however, to further know that the Σ^b_i-definable multifunctions of R^i_2 are closed under certain kinds of recursion. We now define two different kinds of weak recursion and show that the Σ^b_i-definable functions of R^i_2 are closed under these operations. These recursion schemes are slight modifications of schemes from [**8**] and [**1**].

Definition 2.

1. *The Ψ-SLLIND axioms are the set of axioms $SLLIND_{\alpha,m}$ defined as*
$$\alpha(0) \wedge (\forall n)(\alpha(n) \supset \alpha(S(n))) \supset (\forall n)\alpha(||n||^m)$$
where α is a formula in Ψ and $m \in \mathbb{N}$.
2. *A multifunction f is defined by SW_2BPR (sped doubly weak bounded primitive recursion) from multifunctions g, h, k, and r if*
$$\begin{aligned} F(0,\vec{x}) &= g(\vec{x}) \\ F(n+1,\vec{x}) &= \min(h(n,\vec{x},F(n,\vec{x})), r(n,\vec{x})) \\ f(n,\vec{x}) &= F(||t(n,\vec{x})||^m, \vec{x}) \end{aligned}$$
for some L_2-terms r and t, and $m \in \mathbb{N}$.
3. *A multifunction f is defined by IC (iteration of concatenation) from multifunction g and L_2-term k if*
$$\begin{aligned} F(0,\vec{x}) &= 0 \\ F(n+1,\vec{x}) &= F(n,\vec{x}) * g(n+1,\vec{x}) \\ f(n,\vec{x}) &= F(|k(n,\vec{x})|, \vec{x}) \end{aligned}$$

If g, h, t, and r are multifunctions then f obtained by SW_2BPR is the multifunction which results by viewing each step in the above iteration as a composition of multifunctions. Similar statements apply to the formation f by IC.

Theorem 3. $(i \geq 1)$

1. *The theory R^i_2 proves Σ^b_i-SLLIND.*
2. *The Σ^b_i-definable multifunctions of R^i_2 are closed under the operations of composition, SW_2BPR and IC.*
3. *If f and g are Σ^b_i-defined functions in R^i_2 then $f = g$ is Δ^b_i with respect to R^i_2.*
4. *The predicates Δ^b_i with respect to R^i_2 are closed under boolean combinations, sharply bounded quantifications and substitutions by Σ^b_i-defined functions.*

Proof. (Σ^b_i-*SLLIND*) It suffices to show for $A(x)$ a Σ^b_i-formula and m a fixed positive integer, that R^i_2 proves $SLLIND_{A,m}$. This is proven by induction on m. The $m = 1$ case follows since R^i_2 proves $LLIND_A$. Assume $SLLIND_{A,m-1}$. Let $B(j) := A(j \cdot ||x||^{m-1})$. The result then follows using $LLIND_B$ once we can show $A(j \cdot ||x||^{m-1}) \supset A((j+1) \cdot ||x||^{m-1})$. This follows from $LLIND_{C(k)}$ on $C(k) := A(j \cdot ||x||^{m-1} + k)$ since we have by the hypothesis of $SLLIND_{A,m}$ that $(\forall x)(A(x) \supset A(Sx))$.

(Composition) See Clote-Takeuti [**8**] for a proof for a theory equivalent to R^1_2 the same proof works in R^i_2 case.

(SW_2BPR) Given that we have Σ^b_i-$SLLIND$ available we can use the same proof as in Clote-Takeuti [**8**] to show TNC (a theory equivalent to R^1_2) is closed under W_2BPR.

(IC) A proof of this can be found in Allen [**1**].

($f = g$) Let $f(x)$ and $g(x)$ be Σ^b_i-definable functions in R^i_2. Then $R^i_2 \vdash (\forall x)(\exists! y)A_f(x,y)$ and $R^i_2 \vdash (\forall x)(\exists! y)A_g(x,y)$ where A_f is a Σ^b_i-formula for the graph of f and A_g is a Σ^b_i-formula of the graph of g. By Parikh's Theorem there are bounds t_f and t_g on the existential quantifiers. So $f = g$ will be Δ^b_i since

$$R^i_2 \vdash (\forall y)(\forall z)(A_f(x,y) \wedge A_g(x,z) \supset y = z) \Leftrightarrow \\ (\exists y)(\exists z)(A_f(x,y) \wedge A_g(x,z) \wedge y = z).$$

(Boolean combinations) This is obvious from the definition of Δ^b_i with respect to a theory.

(Sharply bounded quantification) Suppose A is a Δ^b_i with respect to R^i_2 and is equivalent to the Σ^b_i-formula A^Σ and the Π^b_i-formula A^Π. Then we have $(Qx \leq |t|)A^\Sigma$ where Q is either $\exists$ or $\forall$ is a Σ^b_i since this class is closed under sharply bounded quantification. Similarly, $(Qx \leq |t|)A^\Pi$ is Π^b_i. So $(Qx \leq |t|)A$ is Δ^b_i with respect to R^i_2 since R^i_2 proves $(Qx \leq |t|)A^\Sigma \Leftrightarrow (Qx \leq |t|)A^\Pi$.

(Substitutions) Suppose $f(x)$ is a Σ^b_i-definable function in R^i_2. Then $R^i_2 \vdash (\forall x)(\exists! y)A_f(x,y)$ where A_f is a Σ^b_i-formula for the graph of f. By Parikh's theorem we can assume there is some bound t on y. Now suppose $B(z)$ is a Δ^b_i with respect to R^i_2 and is equivalent to the Σ^b_i-formula Σ and the Π^b_i-formula B^Π. Then $B(f(x))$ will be Δ^b_i with respect to R^i_2 since R^i_2 proves

$$(\exists y \leq t)(A_f(x,y) \wedge B^\Sigma(y)) \Leftrightarrow (\forall y \leq t)(A_f(x,y) \supset B^\Pi(y)).$$

□

4. Arithmetizing Propositional Formulas

To formalize quantified propositional formulas in R^1_2 we use trees which are formalized as in Buss [**5**]. Trees in Buss' scheme are coded using sequences of open and closed brackets with the labels on each node occurring between the brackets. There he was working with the theory S^1_2; however, it is not hard to check that all of his basic predicates and functions can be appropriately defined in R^1_2. We omit the details. To formalize quantified propositional formulas we fix some Gödel numbering for $\wedge$, $\vee$, $\neg$, $\top$, $\bot$, free variables p_i, bound variables x_i, and quantifier symbols $\exists x_i$. Following the usual convention we write $\ulcorner x_i \urcorner$ to denote the Gödel number for x_i. In our formalization, we view $\forall x_i$ as an abbreviation for $\neg\exists x_i \neg$ and $A \supset B$ as an abbreviation for $\neg A \vee B$.

A *formalized quantified boolean formula* w is a tree with the following additional properties: (1) Each of w's nodes is labelled with the number for a symbol, quantifier symbol, or a variable. (2) If i is a leaf then $\beta(i,w)$, the label of this node, is a variable, $\ulcorner\top\urcorner$, or $\ulcorner\bot\urcorner$. (3) If a node i is labelled with a quantifier symbol or a negation symbol it has only one child. Otherwise, all non-leaves have two children. (4) If a leaf is labelled $\ulcorner x_i \urcorner$ then it has an ancestor labelled $\ulcorner \exists x_i \urcorner$. (5) A given quantifier symbol can appear at most once in a formula.

Each of these properties is Δ^b_1 with respect to R^1_2, so the conjunction $QBF(w)$ which asserts that w is a quantified Boolean formula will also be Δ^b_1 with respect to R^1_2.

Once one has formalized the notion of quantified Boolean formula, it is not to hard to give Δ^b_1-predicates in R^1_2 which check if a formula A is in Δ^q_i, Σ^q_i, or Π^q_i. We will write these predicates as $\Sigma^q_i(\ulcorner A\urcorner)$, $\Pi^q_i(\ulcorner A\urcorner)$, and $\Delta^q_i(\ulcorner A\urcorner)$. These predicates are not to be confused with, for instance, $\Sigma^q_i(A)$ which could be viewed as the class of Σ^q_i-formulas with respect to some oracle set A. A function related to these predicates which R^1_2 can Σ^b_1-define is the function $type_i(\ulcorner x_j\urcorner, \ulcorner A\urcorner)$. If there is a subformula w of $\ulcorner A\urcorner$ beginning with $\ulcorner \exists x_j\urcorner$ then $type_i(\ulcorner x_j\urcorner, \ulcorner A\urcorner)$ is the least number k less than i such that $w \in \Sigma^q_k$; otherwise, $type_i(\ulcorner x_j\urcorner, \ulcorner A\urcorner)$ equals $i+1$.

We will be working with sequent calculus proof systems. We formalize the notion of a cedent as a sequence of formulas and a sequent $\Gamma \to \Delta$ as an ordered pair of cedents.

We now want to define a predicate $TRU_i(\ulcorner A\urcorner, \tau)$ which is true if $A \in \Sigma^q_i \cup \Pi^q_i$ and τ is a satisfying truth assignment for $\ulcorner A\urcorner$. Our definition will be a Boolean combination of Σ^b_i-formulas (we denote this class $B(\Sigma^b_i)$). This is the usual complexity for such definitions [**15**]. However, we go into some detail about this predicate so that we can argue in a Section 7 that we can define the Σ^b_i-formula $Wit^i(\ulcorner A\urcorner, w, \tau)$. The formula $Wit^i(\ulcorner A\urcorner, w, \tau)$ as we will define it checks whether replacing the variables x_j in A that have $type_i(x_j, \ulcorner A\urcorner) = i$ by their value in the truth assignment w makes τ a satisfying assignment for the resulting formula.

To begin we define a *truth assignment* to be a sequence of ordered pairs of the form $\langle \ulcorner x_i\urcorner, \ulcorner\top\urcorner\rangle$ or $\langle \ulcorner x_i\urcorner, \ulcorner\bot\urcorner\rangle$ where x_i is the variable being assigned and $\ulcorner\bot\urcorner$ or $\ulcorner\top\urcorner$ is the value assigned. We require that a variable not appear twice in a truth assignment. Let $Assign(\tau)$ be the Δ^b_1-predicate which asserts τ is a truth assignment.

To substitute the variables in a formula for their values in a truth assignment we use the function $Sub(\ulcorner A\urcorner, \tau)$. This function returns a sequence which is the same as $\ulcorner A\urcorner$ except that where a variable in $\ulcorner A\urcorner$ appears in τ it is replaced by the value that τ assigns that variable. If $\ulcorner A\urcorner$ is not a formula or τ is not a truth assignment then $Sub(\ulcorner A\urcorner, \tau)$ is 0. It is not hard to Σ^b_1-define Sub in R^1_2 using IC.

In order to evaluate a Δ^q_0-formula all of whose leaves are either $\ulcorner\top\urcorner$ or $\ulcorner\bot\urcorner$ we first define a function $Evaltree_0(\ulcorner A\urcorner)$ which produces a tree shaped like A but where the values of A's leaves have been propagated up to the root. To be more precise let $Evaltree_0(\ulcorner A\urcorner)$ be the function with the following properties: (1) $Evaltree_0(\ulcorner A\urcorner)$ outputs either 0 or a tree y with the same structure as $\ulcorner A\urcorner$ but with different node labels. (2) It outputs 0 if any leaf of $\ulcorner A\urcorner$ is a variable or if $\ulcorner A\urcorner$ is not a QBF-formula. (3) Otherwise the leaves of y are labelled as in $\ulcorner A\urcorner$. (4) If node i is labelled with a propositional connective in $\ulcorner A\urcorner$ then i is labelled with $\ulcorner\top\urcorner$ or $\ulcorner\bot\urcorner$ in y according to the usual semantics of propositional logic from the values of its children. (5) If node i is labelled with an $\ulcorner\exists x_j\urcorner$ in $\ulcorner A\urcorner$ then i is labelled with the same label as its son in y (we will use $Evaltree_0$ when we define $Evaltree_k$ for Σ^q_k-formulas so it is important that $Evaltree_0$ can handle nodes labelled $\ulcorner\exists x_j\urcorner$).

Given that the Boolean Formula Value Problem (BFVP) is in NC^1 [**7**] it should seem plausible that we can Σ^b_1-define $Evaltree_0(\ulcorner A\urcorner)$ in R^1_2. Since R^1_2 has functions in NC at its disposal we could implement such a function using an algorithm based on Brent-Spira [**4, 17**]. As similar constructions have already been done for theories weaker than R^1_2 such as TNC^0 (see [**19**]) we do not give the proof.

Using Theorem 3, we can now give a Δ^b_1-definition of $TRU_0(\ulcorner A \urcorner, \tau)$.

$$TRU_0(\ulcorner A \urcorner, \tau) \quad \Leftrightarrow \quad A \in \Delta^q_0 \wedge \beta(1, Evaltree_0(Sub(\ulcorner A \urcorner, \tau))) = \ulcorner \top \urcorner$$

$TRU_0(\ulcorner A \urcorner, \tau)$ is true iff the truth assignment τ satisfies $\ulcorner A \urcorner \in \Delta^q_0$. Given the definition of $Evaltree_0$ it is not hard to show that TRU_0 respects propositional connectives. For instance,

$$R^1_2 \vdash TRU_0(\ulcorner A \wedge B \urcorner, \tau) \Leftrightarrow TRU_0(\ulcorner A \urcorner, \tau) \wedge TRU_0(\ulcorner B \urcorner, \tau)$$

We now inductively extend the definition of $TRU_0(\ulcorner A \urcorner, \tau)$ to a definition for $TRU_i(\ulcorner A \urcorner, \tau)$ that works for $A \in \Sigma^q_i$. First, R^i_2 can Σ^b_i-define for $i \geq 1$ the following multifunctions:

$$\begin{aligned} WitTree_i(\ulcorner A \urcorner, w) = y \Leftrightarrow A \in \Sigma^q_i \wedge Assign(w) \wedge \\ (\forall k < len(w))(type_i(\beta(1, \beta(k+1, w)), \ulcorner A \urcorner) = i) \wedge \\ Evaltree_{i-1}(Sub(\ulcorner A \urcorner, w)) = y \end{aligned}$$

$$Evaltree^{\exists}_i(\ulcorner A \urcorner) = y \Leftrightarrow (\exists w \leq (\ulcorner A \urcorner)^2)(WitTree_i(\ulcorner A \urcorner, w) = y).$$

Intuitively, $WitTree_i(\ulcorner A \urcorner, w)$ on input A, a closed Σ^q_i-formula, and w a truth assignment for A's outermost existential variables produces a tree shaped like A where the values of A's leaves have been propagated up the tree and where the outermost existential variables of A have been evaluated according to w. It is Σ^b_i-definable since we are assuming $Evaltree_{i-1}$ is Σ^b_i-definable. The multifunction $Evaltree^{\exists}_i(\ulcorner A \urcorner)$ outputs y if there is some choice of truth assignment w for the outermost existential variables of A such that $WitTree_i(\ulcorner A \urcorner, w) = y$. To show for each Σ^q_i-formula $\ulcorner A \urcorner$ there is a y output by $Evaltree^{\exists}$, the theory R^i_2 can run $WitTree_i$ on the assignment which assigns all the outermost existential variable of A zero. It should be noted that the w which appears in the definition of $Evaltree^{\exists}_i$ is not necessarily unique so $Evaltree^{\exists}_i$ is really a multifunction. Recall the Σ^b_1-function $Subtree(j, \ulcorner A \urcorner)$ returns the subtree under the node j of A viewed as tree and is defined in [**5**].

From the above two multifunctions R^{i+1}_2 can Σ^b_{i+1}-define the multifunction $Evaltree_i(\ulcorner A \urcorner)$ which extends the operations of $Evaltree^{\exists}_i$ to Δ^q_{i+1}-formulas. We define $Evaltree_i(\ulcorner A \urcorner)$ so that: (1) $Evaltree_i(\ulcorner A \urcorner)$ is either 0 or it outputs a tree y with the same structure as $\ulcorner A \urcorner$ but with different node labels. (2) It equals 0 if any leaf of $\ulcorner A \urcorner$ is a free variable. (3) For any node j in a Σ^q_i subtree,

$$Subtree(j, y) = Evaltree^{\exists}_i(Subtree(j, \ulcorner A \urcorner)).$$

(4) For any $\Sigma^q_{i-1} \cup \Pi^q_{i-1}$-subtree B of A, the value of its root node j in y is $\top$ iff $TRU_{i-1}(B, \langle\rangle)$ and is $\bot$ otherwise. (5) $Evaltree_i(\ulcorner A \urcorner)$ has properties (4) and (5) of $Evaltree_0(\ulcorner A \urcorner)$.

Finally, we give a $B(\Sigma^b_i)$-definition of the predicate $TRU_i(\ulcorner A \urcorner, \tau)$ as

$$\begin{aligned} & TRU_i(\ulcorner A \urcorner, \tau) \Leftrightarrow \\ & \quad (A \in \Sigma^q_i \wedge (\exists y)(Evaltree^{\exists}_i(Sub(\ulcorner A \urcorner, \tau)) = y \wedge \beta(1, y) = \ulcorner \top \urcorner)) \\ & \quad \vee (A \in \Pi^q_i \wedge \neg(\exists y)(Evaltree^{\exists}_i(Sub(\ulcorner \neg A \urcorner, \tau)) = y \wedge \beta(1, y) = \ulcorner \top \urcorner)) \end{aligned}$$

This predicate returns the truth value of formulas in $\Sigma^q_i \cup \Pi^q_i$.

5. Translations of Σ_i^b-formulas

Consider a bounded formula $A(a_1, \dots, a_k)$. The functions in the language L_2 are polynomial time computable, so there exists a polynomial $p_A(x)$ such that for $|n_1|, \dots, |n_k| \leq m$ one only needs to compute numbers of length less than $p_A(m)$ to determine the truth of $A(n_1, \dots, n_k)$. Any polynomial which is larger than $p_A(x)$ we will call a bounding polynomial [**15, 10**].

Given a bounding polynomial $q(x)$ we shall make a sequence of propositional translations of $A(a_1, \dots, a_k)$, which we will call $[\![A]\!]^m_{q(m)}$, using the variables

$$\begin{array}{ccc} \vec{p}_{a_1} & := & p^0_{a_1}, \dots, p^{q(m)}_{a_1} \\ \vdots & \vdots & \vdots \quad \vdots \quad \vdots \\ \vec{p}_{a_k} & := & p^0_{a_k}, \dots, p^{q(m)}_{a_k}. \end{array}$$

We shall often write $[\![A]\!]^m_{q(m)}$ as $[\![A]\!]$ for simplicity. If A is atomic then it is of the form:

$$t(a_1, \dots, a_k) = s(a_1, \dots, a_k)$$

or

$$t(a_1, \dots, a_k) \leq s(a_1, \dots, a_k).$$

To translate A we construct polynomial size, log-depth formulas computing the first $q(m)$ bits for the terms t and s on the variables $\vec{p}_{a_1}, \dots, \vec{p}_{a_k}$. As s and t are NC computable (since circuits for $+$, $\cdot$, MSP, etc are NC^1 computable) this can be done. We substitute these two formulas into another polynomial-sized, log-depth formula which checks if two $q(m)$ bit numbers are equal (resp. less than or equal) to get a polynomial-sized, log-depth formula for the atomic formula A. Given a vector $\vec{\epsilon} = \epsilon^1, \dots, \epsilon^{q(m)}$, we write $[\![A(a)]\!](\vec{\epsilon}/\vec{p}_a)$ to denote the substitution of the variables ϵ^i for the variables p^i_a in $[\![A(a)]\!]$ where $0 \leq i \leq q(m)$. We extend the propositional translation of atomic formulas of R^i_2 to general nonatomic formulas with quantifiers in the natural way as in [**10, 15, 6**]:

1. $[\![A \circ B]\!]$ is $[\![A]\!] \circ [\![B]\!]$ for $\circ = \wedge, \vee, \supset$.
2. $[\![\neg A]\!]$ is $\neg[\![A]\!]$.
3. $[\![(\exists x \leq |t|)A(x)]\!]$ is $\vee_{\vec{\epsilon} \in S}[\![a \leq |t| \wedge A(a)]\!](\vec{\epsilon}/\vec{p}_a)$ where $S = \{(\epsilon_0, \dots, \epsilon_{q(m)}) \in \{0,1\}^{q(m)+1} \mid (\forall i \geq |q(m)|)(\epsilon_i = 0)\}$.
4. $[\![(\forall x \leq |t|)A(x)]\!]$ is $\wedge_{\vec{\epsilon} \in S}[\![a \leq |t| \wedge A(a)]\!](\vec{\epsilon}/\vec{p}_a)$.
5. $[\![(\exists y \leq t)A(y)]\!]$ is $\exists x^0_y \dots \exists x^{q(m)}_y [\![a \leq t \wedge A(a)]\!](\vec{x}_y/\vec{p}_a)$.
6. $[\![(\forall y \leq t)A(y)]\!]$ is $\forall x^0_y \dots \forall x^{q(m)}_y [\![a \leq t \wedge A(a)]\!](\vec{x}_y/\vec{p}_a)$.

Since all of the functions in L_2 are computable in NC^1, it is not hard to see that the function $I^m \mapsto [\![A]\!]^m_{q(m)}$ is Σ^b_1-definable in R^1_2. Two important properties of the above translation are that there is a polynomial $r(x)$ such that $|[\![A]\!]^m_{q(m)}| \leq r(m)$; and if $A \in \Sigma^b_i$, $i \geq 0$, then $[\![A]\!] \in \Sigma^q_i$. We close this section with the following useful lemma.

Lemma 4. ($i \geq 1$) *Let $A(a_1, \dots, a_k)$ be a Σ^b_i-formula and $q(x)$ a bounding polynomial for A then*

$$R^1_2 \vdash (\forall y)[(\forall \tau)((Assign(\tau) \supset TRU_i([\![A]\!]^{|y|}_{q(|y|)}, \tau)) \equiv$$
$$(\forall x_1, \dots x_k)(|x_1| \leq |y| \wedge \dots \wedge |x_k| \leq |y| \supset A(\vec{x}))]$$

Proof. This statement is proven by a straightforward induction on the complexity of A. The base case, when A is atomic, is the only hard case. Let $\rho(x_1, \dots, x_k)$ be the Σ^b_1-defined truth assignment which for $1 \leq i \leq k$ and for $0 \leq j \leq q(m)$ assigns $p^j_{x_t}$ the value $\top$ if $Bit(j, x_t)$ is true and assigns $p^j_{x_t}$ the value $\bot$ otherwise. By Theorem 3, the formula $B(m)$ defined as

$$TRU_0([\![A]\!]^m_{q(m)}, \rho(x_1, \dots, x_k)) \equiv$$
$$(|x_1| \leq m \wedge \dots |x_k| \leq m \supset A(\vec{x}))$$

is a Δ^b_1-predicate in R^1_2 since the function $I^m \mapsto [\![A]\!]^m_{q(m)}$ and the function ρ are Σ^b_1-definable in R^1_2. Thus, R^1_2 using Δ^b_1-$LIND$ on $B(m)$ can prove the base case. Since $[\![A]\!]^m_{q(m)}$ is Δ^q_0 it is not hard to show from the definition of TRU_i that $TRU_0([\![A]\!]^m_{q(m)}, \rho) \Leftrightarrow TRU_i([\![A]\!]^m_{q(m)}, \rho)$. □

6. Proof Systems

We proceed with the task of defining a propositional proof system for R^i_2. We define quantified propositional proof systems as in [**15**]:

Definition 5. *A polynomial time computable binary relation $P(d, A)$ is a* quantified propositional proof system *(or just* proof system*) iff $\exists d P(d, \ulcorner A \urcorner)$ implies A is a valid $\Sigma^q_i \cup \Pi^q_i$-formula for some $i \geq 1$.*

We will often write $d : P \vdash A$ instead of $P(d, \ulcorner A \urcorner)$ and call d a P-proof of A. The proof systems we use in this paper are not only computable in polynomial time they are also computable in NC.

Definition 6. *Let Q and R be two proofs systems.*

1. *We say Q* p-simulates *R iff there is a polynomial time function r such that for any proof d of A in R, $r(d, \ulcorner A \urcorner)$ is a proof of A in Q.*
2. *We say Q* j-polynomially simulates *R, $Q \geq_j R$ in symbols, iff there is a polynomial function $f(d, \ulcorner A \urcorner)$ such that*

$$(\forall d, A)(Q(d, \ulcorner A \urcorner) \wedge A \in \Sigma^q_j \cup \Pi^q_j \supset R(f(d, \ulcorner A \urcorner), \ulcorner A \urcorner)$$

We will mention the notion of j-polynomially simulates again in the last section of this paper. This notion was first defined in [**15**].

Definition 7. *For* P *a proof system and $i \geq 0$, i-$RFN(P)$ is the formula:*

$$(\forall d, \ulcorner A \urcorner, \tau)(d : P \vdash A \wedge A \in \Sigma^q_i \wedge Assign(\tau) \supset TRU_i(\ulcorner A \urcorner, \tau))$$

Since TRU_i is a $B(\Sigma^b_i)$-formula, i-$RFN(P)$ is a $\forall\Sigma^b_i$-formula.

The quantified propositional proof systems we will be working with in this paper are formulated over the sequent calculus and will allow the following rules of inference:

(a) structural rules, propositional rules, and the cut rule for the system LK as defined in [**14**]. As we make frequent use of the cut rule and the $\wedge$-right rule, we list them as examples:

(Cut rule) $\dfrac{\Gamma \to \Delta, A \qquad A, \Gamma \to \Delta}{\Gamma \to \Delta}$ ($\wedge$:right) $\dfrac{\Gamma \to \Delta, A \qquad \Gamma \to \Delta, B}{\Gamma \to \Delta, A \wedge B}$

(b) propositional quantifier rules defined as follows:

($\forall$:left) $\dfrac{A(\vec{B}), \Gamma \to \Delta}{\forall \vec{x} A(\vec{x}), \Gamma \to \Delta}$ ($\forall$:right) $\dfrac{\Gamma \to \Delta, A(\vec{p})}{\Gamma \to \Delta, \forall \vec{x} A(\vec{x})}$

($\exists$:left) $\dfrac{A(\vec{p}), \Gamma \to \Delta}{\exists \vec{x} A(\vec{x}), \Gamma \to \Delta}$ ($\exists$:right) $\dfrac{\Gamma \to \Delta, A(\vec{B})}{\Gamma \to \Delta, \exists \vec{x} A(\vec{x})}$

The $\vec{B}$ in the above is a vector of quantifier-free (i.e., propositional) formulas. The $\vec{p}$ in the above is a vector of distinct variables each of which must not appear in the lower sequent of the inference in which they are involved.

Definition 8. *The* proof system G_i^* *is the tree-like proof system with initial sequents of the form* $\bot \to$, $\to \top$, *or* $A \to A$ *where* A *can be any* $\Sigma_i^q \cup \Pi_i^q$*-formula. It has the above rules of inference and all formulas in a* G_i^**-proof are restricted to be in* $\Sigma_i^q \cup \Pi_i^q$*. (We have made inessential modifications to the definition of* [**15**]*.) The* proof system L_i^* *is the proof system* G_i^* *with the following additional restrictions: An* L_i^**-proof* P *must have height less than* $(\log(|P|))^2$ *and* P *can have at most* $(\log\log(|P|))^2$ *cuts on* $\Sigma_i^q \cup \Pi_i^q$*-formulas along any branch.*

Recall $A \supset B$ is formalized in R_2^1 as $\neg A \vee B$. Thus, the result of an $\supset$-rule can be achieved using $\vee$-rules and $\neg$-rules with at most a constant increase in the height of the proof. All of our results about the proof system L_i^* go through for the proof system which consists of L_i^* without these rules; however, these rules are convenient to have for several of our proofs.

Remark 9 The exponent of 2 in the bounds on the height and number of $\Sigma_i^q \cup \Pi_i^q$-cuts in an L_i^*-proof is not too critical, essentially we need a function which grows superlinearly. We will have more to say on this after each of our main results.

We can formalize L_i^*-proofs in R_2^1 as a Δ_1^b-predicate $L_i^*(d, \ulcorner \Gamma \to \Delta \urcorner)$. To do this we only need a predicate which checks for a sharply bounded number of nodes that each is a initial sequent or follows from its children by one of a finite list of inferences. We also need to check the bounds on the height of the proof and on the number of $\Sigma_i^q \cup \Pi_i^q$-cuts. These are all Δ_1^b properties.

We can similarly formalize G_i^*-proofs in R_2^1 as a Δ_1^b-predicate $G_i^*(d, \ulcorner \Gamma \to \Delta \urcorner)$.

Definition 10. *Let* T *be a theory over the language* L_2*. Let* $\forall\Sigma_i^b(T)$ *denote* $\forall\Sigma_i^b$*-consequences of* T*. If* P *is a proof system,* P simulates $\forall\Sigma_i^b(T)$ *iff for any* $\forall \vec{x} A(\vec{x}) \in \forall\Sigma_i^b(T)$ *there is a bounding polynomial* $p(x)$ *of* A *such that:*

$$R_2^1 \vdash \forall y (P \vdash [\![A]\!]_{p(|y|)}^{|y|}).$$

This is a slightly weaker notion of simulation than was defined in [**15**] where S_2^1 was used in place of R_2^1. A *Frege proof system* is a complete propositional proof system that uses a finite axiom schemata, and has modus ponens as its only rule of inference. (We are requiring $\supset$ to be in language. The usual definition of Frege proof system is more general in that it does not have this condition and it allows so-called

Frege rules (see [**11, 14**]) as opposed to modus ponens; however, our definition will suit our present purpose.) A *Frege proof* is a sequence of propositions $A_1, \dots, A_n$ where each A_i is either a substitution instance of an axiom or follows from earlier propositions by modus ponens. An *extended Frege proof system* is a modification of Frege proof systems that allows extension rules of the form $p \equiv B$ in the proof sequence provided p does not occur earlier in the sequence or in the final line of the proof. A *substitution Frege proof system* is a modification of Frege proof systems that allows substitution inferences. All Frege systems p-simulate each other. Any extended Frege or substitution Frege system can p-simulate any other extended Frege or substitution system (see [**14**]). It is also known that Frege systems can simulate $\forall\Sigma^b_0(BASIC)$ and extended Frege systems simulate $\forall\Sigma^b_0(S^1_2)$ [**6**]. In fact, Frege can simulate $\forall\Sigma^b_0(TNC^0)$, a theory for NC^1 containing $BASIC$ developed in Takeuti [**19**]. The Frege system that was used to show this had only the rule modus ponens and was over the basis $\wedge$, $\vee$, $\neg$, and $\supset$. The next lemma will allow us to use the fact that Frege systems simulate $\forall\Sigma^b_0(BASIC)$ in our simulation of R^i_2 by L^*_i. The basic idea of this lemma is based on techniques in Krajíček [**14**] and Bonet [**3**].

Lemma 11. *Let F be a Frege System as defined above. The proof systems L^*_1 p-simulates F. Further, this is provable in R^1_2.*

Proof. We can formalize Frege systems in R^1_2 in much the same way as we formalized L^*_i. Let $A_1, \dots, A_n$ be a proof in F of A_n. Let B_j be the balanced conjunction $\bigwedge\!\!\!\bigwedge^j_{i=1} A_i$. We first derive $\rightarrow B_1$ and $B_j \rightarrow B_{j+1}$ for $1 \leq j \leq n-1$. Now $B_1 = A_1$ is a substitution instance of a Frege axiom and we can give an L^*_1-proof of this axiom since a substitution instance of an axiom can be simulated with an L^*_1-proof of height which is independent of the size of the substituted formula. This is because in L^*_1-proofs we allow initial sequents $A \rightarrow A$ where A can be any $\Sigma^q_1 \cup \Pi^q_1$-formula. To derive $B_j \rightarrow B_{j+1}$ there are two cases to consider. In the first case, A_{j+1} is an axiom. We first derive $\rightarrow A_{j+1}$ and weaken on B_j to get $B_j \rightarrow A_{j+1}$. We then derive $B_j \rightarrow A_i$ for $i \leq j$ with a proof of height less than $2 \cdot \log n$. We then use $(\wedge : right)$ rules in a balanced fashion to derive $B_j \rightarrow B_{j+1}$. In the second case, A_{j+1} follows by modus ponens from some earlier A_k and $A_l := A_k \supset A_{j+1}$. In this case we derive $B_j \rightarrow A_k$ with a proof d of height less than $2 \cdot \log n$. We also do the following proof:

$$\dfrac{\begin{array}{c}[d]\\ B_j\rightarrow A_k\end{array} \qquad \dfrac{\dfrac{\dfrac{A_k\rightarrow A_k}{A_k\rightarrow A_k, A_{j+1}}}{A_k\rightarrow A_{j+1}, A_k} \qquad \dfrac{\dfrac{A_{j+1}\rightarrow A_{j+1}}{A_k, A_{j+1}\rightarrow A_{j+1}}}{A_{j+1}, A_k\rightarrow A_{j+1},}}{A_k \supset A_{j+1}, A_k\rightarrow A_{j+1}}}{B_j, A_k\supset A_{j+1}\rightarrow A_{j+1}}$$

Recall $A_k \supset A_{j+1}$ is just the formula A_l. By $2 \cdot |n|$ uses of $(\wedge : right)$, a weakening, and a contraction we can derive $B_j \rightarrow A_{j+1}$. From this, doing the same thing as we did in the axiom case we can derive $B_j \rightarrow B_{j+1}$. Using $\rightarrow B_1$ and $B_j \rightarrow B_{j+1}$ for $1 \leq j \leq n-1$ and cutting in a balanced way we can derive $\rightarrow B_n$. Cutting this and a proof of $B_n \rightarrow A_n$, we get $\rightarrow A_n$. Call this new proof P. Since there are only finitely many axiom schemes in our Frege system and each one can be derived by some fixed height L^*_1-proof, there is a fixed number m which bounds the height of any axiom derivation in L^*_1. So the height of deriving $B_j \rightarrow B_{j+1}$ if A_{j+1} is an axiom is bounded by $3 \cdot \log n + m$. The height of deriving $B_j \to B_{j+1}$ if A_{j+1} follows from modus ponens is bounded by $5 \cdot |n|$. Finally, the end of the proof has fewer then $2 \cdot \log n$ many cuts. The total height of P is bounded by $7 \cdot \log n + m$. To guarantee this is less than $||P||^2$, we can add the following padding to the bottom of our proof

$$\dfrac{\dfrac{\to \top}{\to \top^{n^{7+m}}} \qquad \dfrac{\dfrac{[P]}{\to A_n}}{\top^{n^{7+m}} \to A_n}}{\to A_n}$$

The symbol $\top^{i+1}$ means $\top \wedge \top^i$. The idea of the above proof fragment is to use $\top^{n^{7+m}}$ to pad the proof so that it is indeed (log^2) height. As there are no cuts on $\Sigma_1^q \cup \Pi_1^q$-formula used in our construction this will be an L_1^*-proof.

To formalize this simulation in R_2^1 is not hard and we only give the barest outline. First, R_2^1 can Σ_1^b-define functions using IC which take substitution instances of axioms of F and outputs L_1^*-proofs of them. Since there are only two cases we need to consider, we can use these functions to Σ_1^b-define a function using IC which given j and $A_1, \ldots, A_n$ outputs a L_1^*-proof $B_j \to B_{j+1}$ for $j > 1$ and $\to B_1$ for $j = 1$. From this function in turn we can Σ_1^b-define a function using IC which takes $A_1, \ldots, A_n$ and outputs the desired L_1^*-proof. □

7. Witnessing for TRU_i

We would like to prove $i\text{-}RFN(L_i^*)$ in R_2^i. Since $i\text{-}RFN(L_i^*)$ is a $\forall\Sigma_i^b$-formula trying to prove this by direct induction on, for instance, the size of subproofs of a fixed L_i^*-proof d would require induction on a Π_{i+1}^b-formula. To avoid this we use a version of the now familiar witnessing argument of Buss [**5**]. A witness in this argument will be a truth assignment for the outermost boolean existential quantifiers in a Σ_i^q-formula. We define our witness predicate as follows:

$$Wit^i(\ulcorner A \urcorner, w, \tau) \quad := \quad \beta(1, WitTree_i(Sub(\ulcorner A \urcorner, \tau), w)) = \ulcorner \top \urcorner$$

The idea behind $Wit^i(\ulcorner A \urcorner, w, \tau)$ is that it checks whether replacing the variables x_j that have $type_i(x_j, \ulcorner A \urcorner) = i$ by their values in the truth assignment w makes τ a satisfying assignment for the resulting formula. Because of the definition of $WitTree_i$, Wit^i is false if A is not a Σ_i^q-formula. We now argue Wit^i is a Δ_i^b-predicate in R_2^i. From the definition of $WitTree_i$ the theory R_2^i can prove if $\ulcorner A \urcorner$ is a Σ_i^q-formula and w and τ are truth assignments then $\beta(1, WitTree_i(Sub(\ulcorner A \urcorner, \tau), w)$ outputs either $\ulcorner \top \urcorner$ or $\ulcorner \bot \urcorner$. By property (4) of $Evaltree_i$, the theory R_2^i can prove for a given input only one of these two possibilities holds. Hence, $\beta(1, WitTree_i(Sub(\ulcorner A \urcorner, \tau), w)$ is a Σ_i^b-definable function in R_2^i. So by Lemma 3 Wit^i is a Δ_i^b-predicate in R_2^i.

For $\Gamma = (A_0, .., A_j)$ a cedent of Σ_i^q-formulas we define the following two predicates:

$$Wit_\wedge^i(\ulcorner \Gamma \urcorner, w, \tau) \Leftrightarrow (\forall k < len(\Gamma)) Wit^i(\ulcorner A_{S(k)} \urcorner, \beta(S(k), w), \tau)$$

and

$$Wit_\vee^i(\ulcorner \Gamma \urcorner, w, \tau) \Leftrightarrow (\exists k < len(\Gamma)) Wit^i(\ulcorner A_{S(k)} \urcorner, \beta(S(k), w), \tau).$$

Since Wit^i is Δ_i^b with respect to R_2^i by Lemma 3, both these predicates are also Δ_i^b with respect to R_2^i. Before we make precise how we will use the witness predicates to prove $i\text{-}RFN(L_i^*)$, let us establish what R_2^i can say about witnessing.

Proposition 12. *Let A be a Σ^q_i-formula and τ a truth assignment for the free variables in A. Then:*

$$R^1_2 \vdash Wit^i(\ulcorner A \urcorner, w, \tau) \supset TRU_i(\ulcorner A \urcorner, \tau)$$

PROOF. From the definition of Wit^i, the theory R^1_2 proves $Wit^i(\ulcorner A \urcorner, w, \tau)$ implies $(\exists y)(Evaltree^{\exists}_i(\ulcorner A \urcorner, \tau)) = y \wedge \beta(1, y) = \ulcorner \top \urcorner)$. This then implies $TRU_i(\ulcorner A \urcorner, \tau)$. □

The next result will allow us to prove i-$RFN(L^*_i)$ in R^i_2. Before we state it, we define the functions $l(\ulcorner \Gamma \to \Delta \urcorner)$ and $r(\ulcorner \Gamma \to \Delta \urcorner)$ which we will need because R^i_2 has only a limited ability to manipulate L^*_i-proofs. For instance, given an L^*_i-proof of a sequent of Σ^q_i-formulas, it seems complicated to show R^i_2 can transform this proof into one with only Σ^q_i-formulas.

The function $l(\ulcorner \Gamma \to \Delta \urcorner)$ produces a cedent Γ^* which contains the Σ^q_i-formulas of Γ in the order they appear in Γ followed by the negations of the Π^q_i-formulas which are not Σ^q_i-formulas of Δ in the order they appear in Δ. The function $r(\ulcorner \Gamma \to \Delta \urcorner)$ produces a cedent Δ^* which contains the negations of the Π^q_i-formulas which are not Σ^q_i-formulas of Γ in the order they appear in Γ followed by the Σ^q_i-formulas of Δ in the order they appear in Δ. We will usually write Γ^* for $l(\ulcorner \Gamma \to \Delta \urcorner)$ and Δ^* for $r(\ulcorner \Gamma \to \Delta \urcorner)$. In a slight abuse of the usual convention, we will often refer to Γ^* as the antecedent of the sequent $\Gamma \to \Delta$ and refer to Δ^* as the succedent of the sequent $\Gamma \to \Delta$.

Theorem 13. $(i \geq 1)$ *Let τ be a free variable. There is a Σ^b_i-definable in R^i_2 multivalued function $ProofWit_i$ such that:*

$$\begin{aligned} R^i_2 \vdash \forall d(d : L^*_i \vdash \Gamma \to \Delta \supset \\ (Wit^i_{\wedge}(\ulcorner \Gamma^* \urcorner, w, \tau) \supset Wit^i_{\vee}(\ulcorner \Delta^* \urcorner, ProofWit_i(w, d, \tau))). \end{aligned}$$

PROOF. $ProofWit_i$ is Σ^b_i-defined in R^i_2 from three Σ^b_i-definable multifunctions $InitProof_i(d, w, \tau)$, $CompAnt_i(d', w, \tau)$, and $CompSucc_i(d', w, \tau)$. The idea of how $ProofWit_i$ works is that after an initialization, we iteratively apply first the multifunction $CompAnt_i$ which proceeds up a proof trying to find witnesses for the antecedents and then apply the multifunction $CompSucc_i$ which proceeds down this proof trying to find witnesses for succedents. Cuts on Σ^q_i-formulas are what force us to take more than one application of these two multifunctions. At each stage of the computation we require that R^i_2 can prove that if the original witness w for the antecedent of the endsequent of a proof is correct then witnesses computed at each step by $ProofWit_i$ satisfy the desired $Wit^i_{\wedge}$ or $Wit^i_{\vee}$ formulas. We will show the bounds on the height of an L^*_i-proof and on the number of Σ^q_i-cuts on a branch enables R^i_2 to show $ProofWit_i$ eventually computes a witness for the succedent of the endsequent and since R^i_2 proves $ProofWit_i$ computes witnesses correctly at each stage this establishes the theorem.

The argument d' which appears in $CompAnt_i$ and $CompSucc_i$ is supposed to code a *labelled proof.* Any L^*_i-proof is a labelled proof; however, a labelled proof P when viewed as a tree is allowed to have nodes of the form

$$\langle \ulcorner \Gamma \to \Delta \urcorner, w, w', \tau \rangle.$$

where $\ulcorner\Gamma \to \Delta\urcorner$ is a sequent, w is a purported witness for Γ^*, w' is a purported witness to Δ^*, and τ is a truth assignment for the free variables of $\Gamma \to \Delta$. We now give more precise definitions of the three multifunctions which make up $ProofWit_i$.

The function $InitProof_i(d, w, \tau)$ is not hard to define. This function converts an L^*_i-proof d of $\Gamma \to \Delta$ into an L^*_i labelled proof d' where the root in d has been replaced with

$$\langle \ulcorner\Gamma \to \Delta\urcorner, w, 0, \tau\rangle \tag{1}$$

and any cuts in d' are on Σ^q_i-formulas and the only rules in d' that eliminate free variables are ($\exists$:left) rules or ($\forall$:right) rules.

We now define the multifunction $CompAnt_i(d')$. This function begins at the root of the labelled proof d' of the sequent $\Gamma \to \Delta$ where at least the root is of the form (1). Using w in the root node for Γ^*, the multifunction $CompAnt_i(d')$ tries to construct witnesses for all the antecedents in the proof. It outputs a new labelled proof d'' such that: If a node in d' is labelled with only $\Gamma' \to \Delta'$ and $CompAnt_i$ can compute a witness w' for the $(\Gamma')^*$ and also a truth assignment τ' for $\Gamma' \to \Delta'$, then this sequent is replaced with

$$\langle \ulcorner\Gamma' \to \Delta'\urcorner, w', 0, \tau'\rangle.$$

If a node is already labelled with a sequence or if $CompAnt_i$ cannot compute a witness and a truth assignment it leaves the node unchanged. The multifunction $CompAnt_i(d')$ is defined from the subfunctions $AntStep_i(n, d')$ using SW_2BPR. The multifunction $AntStep_i(n, d')$ tries to compute witnesses for the antecedents and truth assignments for the sequents of nodes in d' viewed as a tree of height $n + 1$ from the witnesses and truth assignments already computed in d' for nodes of height n. It then outputs an updated labelled d''. If n is greater than the height of the proof, then $AntStep_i(n, d')$ just returns d'. Given that $AntStep_i$ can be Σ^b_i-defined we define $CompAnt_i(d')$ to be

$$\begin{aligned} F(0, d') &= d' \\ F(n+1, d') &= \min(AntStep_i(n, F(n, d')), 5 \cdot (d' \# d')) \\ CompAnt_i(d') &= F(||d'||^2, d') \end{aligned}$$

The $||d'||^2$ in the above bounds the height of the labelled proof ensuring that we have made an attempt to find a witness for antecedents of each height. The $5 \cdot (d' \# d')$ bounds the size of the proof obtained by replacing all the unlabelled nodes with labels. In defining $AntStep_i(n, d')$ we will show that if $(\Gamma')^*$ and $(\Gamma'')^*$ are two antecedents in d' and $AntStep_i(n, d')$ used witness w' and truth assignment τ' for $(\Gamma')^*$ to compute a witness w'' and truth assignment τ'' for $(\Gamma'')^*$ then

$$R^i_2 \vdash Wit^i_\wedge((\Gamma')^*, w', \tau') \supset Wit^i_\wedge((\Gamma'')^*, w'', \tau'').$$

Thus, using Σ^b_i-$SLLIND$ on the height of a labelled proof d' of $\Gamma \to \Delta$ one can show if w is a witness for Γ^* and τ is a truth assignment for $\Gamma \to \Delta$ and if antecedent $(\Gamma')^*$ in the proof is assigned a witness w' and truth assignment τ' by $CompAnt_i(d')$ then

$$R^i_2 \vdash Wit^i_\wedge(\Gamma^*, w, \tau) \supset Wit^i_\wedge((\Gamma')^*, w', \tau').$$

We now describe how $AntStep_i(n, d')$ is defined. It is defined in R^i_2 using IC from the multifunction $AntCopy_i(j, n, d')$ which we will show is Σ^b_i-defined in R^i_2. The multifunction $AntCopy_i(j, n, d')$ looks at the jth element of the labelled proof d' viewed as a sequence. If this element is not a sequent of height $n+1$ for which the parent of this node has a witness and truth assignment already defined then it just outputs this element. Otherwise, if this element is a sequent of height $n+1$ for which the parent of this node has a witness and truth assignment already defined, then what $AntCopy_i(j,n,d')$ does breaks into cases depending on the kind of inference involved between this element and its parent. As most of these cases are rather straightforward, we only show the cut case and the $(\exists : left)$ case and omit the remaining cases.

Case (Cut) Suppose this inference is of the form

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

Because of the initialization done by $InitProof_i$ we can assume this cut is on a Σ^q_i-formula. The two subcases to consider are the case where the node j is $\Gamma' \to \Delta', A$ and the case where the node j is $A, \Gamma' \to \Delta'$. The basic idea is in the first case we can use the witness for the lower antecedent as the witness for upper antecedent and modify the truth assignment slightly. In the second case, we might need to also use a witness for A from the succedent of the left upper sequent as well as a witness for the lower antecedent to construct a witness for the upper right antecedent. To be more precise in the first case, by hypothesis, we have a witness w' for the cedent $(\Gamma')^*$ and a truth assignment τ' for the free variables in the lower sequent. By the initialization done by $InitProof_i$ we can assume cut inferences do not eliminate any variables. So τ' will be a truth assignment for $\Gamma' \to \Delta', A$. Thus, we can use w' as a witness for the antecedent of $\Gamma' \to \Delta', A$ and τ' as a truth assignment for this sequent. Obviously,

$$R^i_2 \vdash Wit^i_\wedge(\ulcorner(\Gamma')^*\urcorner, w', \tau') \supset Wit^i_\wedge(\ulcorner(\Gamma')^*\urcorner, w', \tau').$$

So we define $AntCopy_i(j,n,d')$ to be the sequence $\langle \ulcorner\Gamma' \to \Delta', A\urcorner, w', 0, \tau'\rangle$.

In the second case, we are trying to find a witness for $A, (\Gamma)^*$ and a truth assignment for $A, \Gamma' \to \Delta'$. The multifunction $AntCopy_i$ first checks the node containing $\Gamma' \to \Delta', A$. If this node does not contain a witness for the cedent $(\Delta')^*, A$ and A is $\Sigma^q_i \backslash \Delta^q_i$ then $AntCopy_i(j,n,d,w,\tau)$ just outputs $A, \Gamma' \to \Delta'$. Otherwise, $AntCopy_i(j,n,d,w,\tau)$ computes a witness v' for A and uses this to compute a witness for $A, (\Gamma)^*$. In the case where A is Δ^q_i and the node containing the sequent $\Gamma' \to \Delta', A$ may or may not have a witness for $(\Delta')^*, A$ the witness v' is just the empty assignment $\langle\rangle$. In the case where A is $\Sigma^q_i \backslash \Delta^q_i$ and the node containing the sequent $\Gamma' \to \Delta', A$ does contain a witness v for $(\Delta')^*, A$ let v' be $\beta(len(v), v)$. We then use v' and the witness w' for $(\Gamma')^*$ to construct $w'' := \langle v'\rangle ** w'$ which will be a witness for $A, (\Gamma)^*$. As in the first subcase, we can use τ' as a truth assignment for $A, \Gamma' \to \Delta'$. It is not hard to see that

$$R^i_2 \vdash Wit^i_\wedge(\ulcorner(\Gamma')^*\urcorner, w', \tau') \wedge Wit^i(\ulcorner A\urcorner, v', \tau') \supset Wit^i_\wedge(\ulcorner A, (\Gamma')^*\urcorner, w'', \tau').$$

So we define $AntCopy_i(j,n,d')$ to be the sequence $\langle \ulcorner A, \Gamma' \to \Delta'\urcorner, w'', 0, \tau'\rangle$.

Case $(\exists : left)$ Suppose j is the upper sequent of the inference

$$\frac{A(\vec{p}), \Gamma'{\longrightarrow}\Delta'}{\exists\vec{x}A(\vec{x}), \Gamma'{\longrightarrow}\Delta'}$$

The two subcases we need to consider are the case where $\exists\vec{x}A$ is $\Sigma_i^q \backslash \Pi_i^q$-formula and the case where $\exists\vec{x}A$ is Σ_{i-1}^q-formula. In the first case, by hypothesis we have a witness w' for the cedent $\exists\vec{x}A(\vec{x}), (\Gamma')^*$ and a truth assignment τ' for the free variables in the lower sequent. Let $w_{\vec{y}}$ be the truth assignment obtained from $\beta(1, w')$ by deleting the assignments for the variables $\vec{x}$. So $w_{\vec{y}}$ is a witness for the variables y_j of A such that $type_i(\ulcorner y_j \urcorner, \ulcorner A \urcorner) = i$. A witness w'' for $A, (\Gamma')^*$ will be just $\langle w_{\vec{y}} \rangle ** Tail(w')$. To make a truth assignment for $A, \Gamma' \rightarrow \Delta'$ let $w_{\vec{x}}$ be the witness one gets by deleting the member of $w_{\vec{y}}$ from $\beta(1, w')$. Now replace the variables x_i's in $w_{\vec{x}}$ by their corresponding p_i's to make a truth assignment $\tau_{\vec{p}}$. Then we define a new truth assignment for $A, \Gamma' \rightarrow \Delta'$ as $\tau'' := \tau_{\vec{p}} ** \tau'$. It is not hard to see that

$$R_2^i \vdash Wit_{\wedge}^i(\ulcorner \exists\vec{x}A, (\Gamma')^* \urcorner, w', \tau') \supset Wit_{\wedge}^i(\ulcorner A, (\Gamma')^* \urcorner, w'', \tau'').$$

So we define $AntCopy_i(j, n, d')$ to be the sequence $\langle \ulcorner A, \Gamma' \rightarrow \Delta' \urcorner, w'', 0, \tau'' \rangle$.

In the second case, since $\exists\vec{x}A$ is a Σ_{i-1}^q-formula, we can use w' as our witness for $A, (\Gamma')^*$. To modify our truth assignment we make a witness query to a Σ_{i-1}^p-oracle to find a truth assignment $\tau_{\vec{p}}$ such that

$$TRU_{i-1}(A(\vec{p}), \tau_{\vec{p}} ** \tau').$$

If no such $\tau_{\vec{p}}$ exists then we define $\tau'' := \tau'$. Otherwise, define $\tau'' := \tau_{\vec{p}} ** \tau'$. It is not hard to see that

$$R_2^i \vdash Wit_{\wedge}^i(\ulcorner \exists\vec{x}A, (\Gamma')^* \urcorner, w', \tau') \supset Wit_{\wedge}^i(\ulcorner A, (\Gamma')^* \urcorner, w', \tau'').$$

So we define $AntCopy_i(j, n, d')$ to be the sequence $\langle \ulcorner A, \Gamma' \rightarrow \Delta' \urcorner, w', 0, \tau'' \rangle$.

This completes the cases we will show in the definition of $AntCopy_i(j, n, d')$. As the above operations in each case are Σ_i^b-definable in R_2^i the multifunction $AntCopy_i(j, n, d')$ is Σ_i^b-definable in R_2^i. As stated before $AntStep_i(n, d')$ is defined from $AntCopy_i$ using IC, so it too will be Σ_i^b-defined in R_2^i. Finally, $CompAnt_i(d')$ will be Σ_i^b-defined as it is defined from $AntStep_i$ using SW_2BPR.

We now define the function $CompSucc_i(d')$. This multifunction starts at the leaves of a labelled proof d' and works towards the root. If a node is labelled with a witness for the antecedent but only 0 as a witness for the succedent then $CompSucc_i(d')$ tries to compute a witness for this succedent. The multifunction $CompSucc_i(d')$ outputs a new labelled proof that incorporates all the witnesses for succedents it was able to find. Like $CompAnt_i$, the multifunction $CompSucc_i(d')$ is defined from a subfunction $SuccStep_i(n, d')$ using SW_2BPR. The multifunction $SuccStep_i(n, d')$ tries to compute witnesses for succedents for sequents which appear in nodes of height $Height(d') - n$. Here $Height(d')$ is the Σ_1^b-definable function which returns the height of the proof d viewed as a tree [**5**]. If n is greater than $Height(d')$ than $SuccStep_i(n, d')$ just outputs d'. Given that $SuccStep_i(n, d')$ can be Σ_i^b-defined we define $CompSucc_i(d')$ to be

$$\begin{aligned} F(0, d') &= d' \\ F(n+1, d') &= \min(SuccStep_i(n, F(n, d')), 5 \cdot (d' \# d')) \\ CompSucc_i(d') &= F(||d'||^2, d') \end{aligned}$$

When we define $SuccStep_i(n, d')$ we will show if $\Gamma' \rightarrow \Delta'$ was inferred from $\Gamma'' \rightarrow \Delta''$ and $(\Delta'')^*$ was given witness w'' by $SuccStep_i(n, d')$ and $(\Delta')^*$ was given witness

w' by $SuccStep_i(n+1, d')$ then

$$R^i_2 \vdash Wit^i_\vee((\Delta'')^*, w'', \tau'') \supset Wit^i_\vee((\Delta')^*, w', \tau')$$

where τ'' and τ' are the truth assignments which were assigned to the upper and lower sequents respectively. We will define $SuccStep_i$ so that if it encounters an initial sequent with a witness w' for its antecedent then it outputs w' as a witness for the succedent.

Thus, using Σ^b_i-$SLLIND$ on the height of a L^*_i-proof d of $\Gamma \to \Delta$ one can show the following: If w is a witness for Γ^* and τ is a truth assignment for $\Gamma \to \Delta$ and there is a sequent $\Gamma' \to \Delta'$ assigned a witness w' for $(\Gamma')^*$ and truth assignment τ' by $CompAnt_i(InitProof(d, w, \tau))$ and $CompSucc_i(d')$ computes a witness w'' for $(\Delta')^*$ then we have

$$R^i_2 \vdash Wit^i_\wedge(\Gamma^*, w, \tau) \supset Wit^i_\vee((\Delta')^*, w'', \tau').$$

We now describe how $SuccStep_i(n, d')$ is defined. Basically, it is defined using IC from a Σ^b_i-defined multifunction $SuccCopy_i(j, n, d')$. The multifunction $SuccCopy_i(j, n, d')$ looks at the jth element of the labelled proof d' viewed as a sequence. If this element is not a node of height $Height(d') - n$ labelled with a sequence of the form

$$\langle \ulcorner \Gamma' \to \Delta' \urcorner, w', 0, \tau' \rangle$$

then it just outputs this element. Otherwise, what $SuccCopy_i(j, n, d')$ does breaks into cases depending on what kind of inference the element j and its children define. In the case where jth element contains an initial sequent $A \to A$, a witness w' for A^* (the formula A may be Π^q_i), a witness 0 for the succedent, and a truth assignment τ' for the sequent, we define $SuccCopy_i(j, n, d')$ to be

$$\langle \ulcorner A \to A \urcorner, w', w', \tau' \rangle.$$

For initial sequent of the form $\to \top$ and $\bot \to$ we also just copy the witness for the antecedent as the witness for the succedent, even though these witnesses are unnecessary since these formulas are Δ^q_0-sentences.

Case (Cut) Suppose the element j and its children define the inference

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

By hypothesis, we have a witness w' for the cedent $(\Gamma')^*$ and a truth assignment τ' for the free variables in

$$\Gamma' \to \Delta'.$$

We also have by hypothesis witnesses v' for $(\Delta')^*, A$ and maybe also a v'' for $(\Delta')^*$. Let w'' be $Front(v')$ if $Wit^i_\vee(\ulcorner(\Delta')^*\urcorner, Front(v'), \tau')$ and otherwise let w'' be v'' if $Wit^i_\vee(\ulcorner(\Delta')^*\urcorner, v'', \tau')$. In these cases we define $SuccCopy_i(j, n, d')$ to be $\langle \ulcorner \Gamma \to \Delta \urcorner, w', w'', \tau' \rangle$. It is not hard to see in these cases that R^i_2 proves w'' is the desired witness. If both of these two cases do not hold than we just output the node as is. Since $Wit^i_\vee$ is a Δ^b_i-predicate we can Σ^b_i-define $SuccCopy_i$ to perform the above tasks.

Case ($\exists : right$) Suppose the element j and its children define the inference

$$\frac{\Gamma' \rightarrow \Delta', A(\vec{B})}{\Gamma' \rightarrow \Delta', \exists \vec{x} A(\vec{x}).}$$

By hypothesis, we have a witness w' for the cedent $(\Gamma')^*$ and a truth assignment τ' for the free variables in

$$\Gamma' \rightarrow \Delta', \exists \vec{x} A(\vec{x}).$$

We also have by hypothesis a witness v' for $(\Delta')^*, A(\vec{B})$. As we can assume that this inference does not eliminate free variables, τ' will be a truth assignment for the variables in $\vec{B}$. Since $\vec{B}$ are Δ_0^q-formulas R_2^i can Σ_i^b-define a function which computes

$$w'' \quad := \quad v' ** \langle \langle \langle x_1, TRU_0'(\ulcorner B_1 \urcorner, \tau') \rangle, \ldots, \langle x_k, TRU_0'(\ulcorner B_k \urcorner, \tau') \rangle \rangle \rangle$$

where $B_1, \ldots, B_k$ are the formulas in $\vec{B}$ and the function TRU_0' output $\ulcorner \top \urcorner$ if TRU_0 holds and $\ulcorner \bot \urcorner$ otherwise.

It is easy to see

$$R_2^i \vdash Wit_\vee^i(\ulcorner (\Delta')^*, A(\vec{B}) \urcorner, v', \tau') \supset Wit_\vee^i(\ulcorner (\Delta')^*, \exists \vec{x} A \urcorner, w'', \tau')$$

So we define $SuccCopy_i(j, n, d')$ to be $\langle \ulcorner \Gamma' \rightarrow \Delta', \exists \vec{x} A \urcorner, w', w'', \tau' \rangle$.

This completes the definition of $SuccCopy_i(j, n, d')$

We are almost ready to define the multifunction $ProofWit_i$. First, we define the function

$$CutStep_i(d') \quad := \quad CompSucc_i(CompAnt_i(d'))$$

From this we define $ProofWit$ using SW_2BPR as follows:

$$\begin{aligned}
F(0, d, w, \tau) &= InitProof(d, w, \tau) \\
F(n+1, d, w, \tau) &= \min(CutStep_i(F(n, d, w, \tau), w, \tau), \ell(d)) \\
f(d, w, \tau) &= F(||d||^3, w, \tau) \\
ProofWit_i(d, w, \tau) &= \beta(3, (\beta(1, f(d, w, \tau))))
\end{aligned}$$

Here $\ell(d)$ is the L_2-term which is the composition of the bound used to define $CompAnt_i$ with that of $CompSucc_i$. The function f outputs a labelled proof d' where the final node of d' is of the form

$$\langle \ulcorner \Gamma \rightarrow \Delta \urcorner, w, w', \tau \rangle$$

where $\ulcorner \Gamma \rightarrow \Delta \urcorner$ is the endsequent of d, w is the input witness for Γ^*, w' is the calculated witness for Δ^*, and τ is the input truth assignment for $\ulcorner \Gamma \rightarrow \Delta \urcorner$. The first application of β in the definition of $ProofWit_i$ extracts this final node from the output of f and the second application outputs the desired w' from this node. One thing to check is that after $||d||^3$ applications of $CutStep_i$ we will really have a witness for Δ^* if w was a witness for Γ^*. For each $\Sigma_i^q \cup \Pi_i^q$-cut c in d there is a list of $\Sigma_i^q \cup \Pi_i^q$-cuts which need to be evaluated before c can be evaluated. Each application of $CutStep_i$ evaluates a layer of $\Sigma_i^q \cup \Pi_i^q$-cuts which share a binary inference other than $\Sigma_i^q \cup \Pi_i^q$-cut as their least common ancestor. In the worst case, for each $\Sigma_i^q \cup \Pi_i^q$-cut above c (except the topmost layer of such cuts) there is a pair of $\Sigma_i^q \cup \Pi_i^q$-cuts which share it as a least common ancestor. Since there are at most $(\log \log |d|)^2$ such cuts along any branch in d this would yield a maximum of

$$\Sigma_{j=1}^{(\log \log |d|)^2} 2^i = 2^{(\log \log |d|)^2 + 1} - 2 < ||d||^3$$

steps which might need to be taken before c could be evaluated. To show R_2^i can prove $ProofWit_i(d, w, \tau)$ computes what it is supposed to, one uses Σ_i^b-$SLLIND$ on this maximum number. For the induction step one uses the fact that we have already shown that R_2^i proves $CompSucc_i$ and $CompAnt_i$ compute what they are supposed to. □

Remark 14 Let $L^*_{i,m}$ denote the proof system which consists of $(\log)^m$-height G_i^* proofs where the number of $\Sigma_i^q \cup \Pi_i^q$-cuts along any branch is bounded by $(\log\log)^m$. Then by just changing the bounds appropriately on each of the applications of SW_2BPR in the above proof allows us to show for $m \geq 1$ that

$$R_2^i \vdash i\text{-}RFN(L^*_{i,m}).$$

If Γ is empty and Δ is a single formula A, then Theorem 13 gives us:

$$\begin{aligned} R_2^i \vdash \forall d(d : L_i^* \vdash A \supset \\ Wit^i_{\wedge}(\langle\rangle, \langle\rangle, \tau)) \supset Wit^i_{\vee}(\ulcorner A \urcorner, ProofWit(\langle\rangle, d, \tau), \tau)). \end{aligned}$$

This implies

$$R_2^i \vdash \forall d(d : L_i^* \vdash A \supset Wit^i(\ulcorner A \urcorner, ProofWit(\langle\rangle, d, \tau), \tau))).$$

By Proposition 12, we have

$$R_2^i \vdash \forall d(d : L_i^* \vdash A \supset TRU_i(A, \tau)).$$

This is just i-$RFN(L_i^*)$, so we have the following theorem:

Theorem 15. $(i \geq 1)$ *The theory R_2^i proves i-$RFN(L_i^*)$.*

8. Simulating the $\forall\Sigma_i^b$-consequences of R_2^i

We now show L_i^* can simulate the $\forall\Sigma_i^b$-consequences of R_2^i.

Theorem 16. *For $i \geq 1$, L_i^* simulates $\forall\Sigma_i^b(R_2^i)$.*

Proof. Let A be a Σ_i^b-formula provable in R_2^i. Using some R_2^i-proof d of A, we want to show

$$R_2^1 \vdash \forall y(L_i^* \vdash [\![A]\!]^{|y|}_{p(|y|)}) \tag{2}$$

where p is some polynomial that bounds the size of all quantified propositional translations of formulas in d. By cut-elimination for R_2^i we can choose a proof d of A such that all the formulas in d are Σ_i^b-formulas. It suffices to show that we can translate d into a sequence of polynomial-size G_i^*-proofs of height bounded by $h_d \cdot \log(|y|) + h_d$ and with fewer than $c_d \cdot \log\log(|y|) + c_d$ cuts on $\Sigma_i^q \cup \Pi_i^q$-formulas along any branch. This is because we can always choose our bounding polynomial p such that $h_d \cdot \log(|y|) + h_d$ is dominated by $(\log(p(|y|)))^2$ and such that $c_d \cdot \log\log(|y|) + c_d$ is dominated by $(\log\log(p(|y|)))^2$.

To show polynomial-size G_i^*-proofs of height bounded by $h_d \cdot \log(|y|) + h_d$ and number of $\Sigma_i^q \cup \Pi_i^q$-cuts along a branch bounded by $c_d \cdot \log\log(|y|) + c_d$, we translate the formulas in d into quantified propositional formulas and "fill in" the gaps in the resulting pre-proof. The proof is by induction on the number of inferences in d and breaks into cases depending on the last inference in d. As in [**15**] we abbreviate $[\![\cdots]\!]$ for $[\![\cdots]\!]^{|y|}_{p(|y|)}$.

(Base Case) Here d is of the form $A \to A$, is an equality axiom, or is a $BASIC$ axiom. The sequent $[\![A]\!] \to [\![A]\!]$ is an initial sequent of an L_i^*-proof so this case is trivial. Frege proof systems are well known to have polynomial size-proofs of translations of equality axioms and $BASIC$ axioms [**10**], [**6**], [**14**]. Since L_1^* can p-simulate Frege, the systems L_i^* where $i \geq 1$ have polynomial size proofs of these axioms. Further, the L_1^* p-simulation of Frege proofs in Lemma 11 gives L_1^*-proofs, and hence, G_1^*-proofs P with height less than $\log(|P|)$ and with all cuts being on Δ_0^q-formulas.

(Non-Quantifier or Induction Inferences) These are handled by the corresponding G_i^*-rule. Call the subproof of d excluding this last inference inference e. By the induction hypothesis there is a constant h_e such that $h_e \cdot \log(|y|)$ bounds the height of the simulations of e and a constant c_e such that $c_e \cdot \log\log(|y|) + c_e$ bounds the number of $\Sigma_i^q \cup \Pi_i^q$-cuts along any branch of e. For these kinds of inferences, $(h_e + 1) \cdot \log(|y|) + h_e$ can be used to bound the heights of the translations of d. The number of $\Sigma_i^q \cup \Pi_i^q$-cuts will remain unchanged for all of these inferences except a cut rule whose maximum number of $\Sigma_i^q \cup \Pi_i^q$-cuts along any branch can be bounded by

$$c_e \cdot \log\log(|y|) + c_e + 1 \leq (c_e + 1) \cdot \log\log(|y|) + (c_e + 1).$$

($\forall$:left)

$$\frac{B(t), \Gamma \to \Delta}{t \leq s, (\forall x \leq s)B(x), \Gamma \to \Delta}$$

Call the proof of the upper sequent e. There are two cases depending on whether or not s is sharply bounded. In both cases we can apply the induction hypothesis to prove:

$$[\![B(t)]\!], [\![\Gamma]\!] \to [\![\Delta]\!]. \tag{3}$$

Assume we have G_i^*-proofs of this translation which are of size polynomial in $|y|$, of height bounded by $h_e \cdot \log(|y|) + h_e$ and such the number of $\Sigma_i^q \cup \Pi_i^q$-cuts along any branch is bounded by $c_e \cdot \log\log(|y|) + c_e$.
(Not Sharply Bounded) We want to derive

$$[\![t \leq s]\!], [\![(\forall x \leq s)B(x)]\!] \to [\![B(t)]\!]. \tag{4}$$

So we can apply cut with the above to get the desired sequent:

$$[\![t \leq s]\!], [\![(\forall x \leq s)B(x)]\!], [\![\Gamma]\!] \to [\![\Delta]\!]. \tag{5}$$

We first derive with G_i^*-proofs of height 3 the sequents

$$[\![B(t)]\!], [\![t \leq s]\!] \to [\![B(t)]\!]$$

and

$$[\![t \leq s]\!] \to [\![B(t)]\!], [\![t \leq s]\!].$$

By one application of an ($\supset$: $left$) rule we then deduce:

$$[\![t \leq s]\!] \supset [\![B(t)]\!], [\![t \leq s]\!] \to [\![B(t)]\!].$$

Now if we perform a couple $(exchange : left)$ rules and perform a $(\forall : left)$ rule quantifying over the Δ^q_0-formulas $\vec{A}_t$ which make up the translation of t we can derive:

$$\llbracket t \leq s \rrbracket, \forall \vec{x} \llbracket t \leq s \supset B(t) \rrbracket (\vec{x}/\vec{A}_t) \rightarrow \llbracket B(t) \rrbracket \tag{6}$$

Given our translation of bounded arithmetic formulas into quantified propositional ones, the formulas (6) and (4) are actually the same. Hence, we are done as we can cut (4) against (3) to derive (5). The height of these new proofs will be bounded by $(h_e + 7) \cdot \log(|y|) + (h_e + 7)$ and the number of $\Sigma^q_i \cup \Pi^q_i$-cuts will be bounded by $(c_e + 1) \cdot \log\log(|y|) + (c_e + 1)$.
(Sharply Bounded) Again, we can apply the induction hypothesis to show R^1_2 proves there are G^*_i proofs of

$$\llbracket B(t) \rrbracket, \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket \tag{7}$$

which are of size polynomial in $|y|$, of height bounded by $h_e \cdot \log(|y|)$ and such the number of $\Sigma^q_i \cup \Pi^q_i$-cuts along any branch is bounded by $c_e \cdot \log\log(|y|)$. By a weakening we can prove:

$$\llbracket t \leq s \rrbracket, \llbracket B(t) \rrbracket, \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket. \tag{8}$$

If $B(a)$ is atomic then we have already said there are G^*_1-proofs of the translation of the equality axiom

$$\llbracket t = a \rrbracket, \llbracket B(a) \rrbracket \rightarrow \llbracket B(t) \rrbracket \tag{9}$$

with height bounded by $h_B \cdot \log(|y|)$ and all the cuts of Δ^q_0-formulas. By induction on the complexity of B the theory R^1_2 can show there are G^*_i-proofs of these sequents for $B \in \Sigma^q_i \cup \Pi^q_i$ and where the height of these G^*_i-proofs is bounded by $h_B \cdot \log(|y|) + h_B$ and where the number of $\Sigma^q_i \cup \Pi^q_i$-cuts along any branch in these proofs is bounded by $c_B \cdot \log\log(|y|) + c_B$ for some fixed constants h_B and c_B. Weakening on $\llbracket \Gamma \rrbracket$ and $\llbracket \Delta \rrbracket$ to (9), performing some exchanges and then cutting the result against (8) we derive:

$$\llbracket t \leq s \rrbracket, \llbracket t = a \rrbracket, \llbracket B(a) \rrbracket, \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket. \tag{10}$$

Applying some exchanges and using $(\wedge : left)$ rules in a balanced fashion to (10) we arrive at:

$$\llbracket t \leq s \rrbracket, \llbracket t = a \rrbracket, \bigwedge_{\vec{\epsilon} \in S} \llbracket B(a) \rrbracket (\vec{\epsilon}/\vec{p}_a), \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket. \tag{11}$$

where S is the set used for translations of sharply bounded quantifiers in the definition of $\llbracket \cdot \rrbracket$. The height of the proofs of this step can be bounded by $h_\wedge \cdot \log(|y|) + h_\wedge$ for some fixed constant $h_\wedge$ since we applied the $(\wedge : left)$ rules in a balanced fashion and the number of exchanges we need to perform is less than the fixed constant $3e$. We can derive (11) for any choice of truth assignment $\vec{\rho}$ to the vector $\vec{p}_a$. Choosing $\vec{\rho}$ from S, we derive using $(\vee : left)$ rules in a balanced fashion

$$\llbracket t \leq s \rrbracket, \bigvee_{\vec{\rho} \in S} \llbracket t = a \rrbracket (\vec{\rho}/\vec{p}_a), \bigwedge_{\vec{\epsilon} \in S} \llbracket B(a) \rrbracket (\vec{\epsilon}/\vec{p}_a), \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket. \tag{12}$$

Again, we can bound the proofs of this step by $h_\vee \cdot \log(|y|) + h_\vee$ since we applied the $(\vee : left)$ rules in a balanced fashion.

It is not hard to see that the sequents

$$\llbracket t \leq s \rrbracket \rightarrow \bigvee_{\vec{\rho} \in S} \llbracket t = a \rrbracket (\vec{\rho}/\vec{p}_a) \tag{13}$$

have polynomial size G_1^*-proofs of height bounded by $h_S \cdot \log(|y|)$ and with all cuts on Δ_0^q-formulas. By cutting (12) and (13), we get the desired

$$\llbracket t \leq s \rrbracket, \bigwedge\!\!\!\bigwedge_{\vec{\epsilon} \in S} \llbracket B(a) \rrbracket(\vec{\epsilon}/\vec{p}_a), \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket. \tag{14}$$

The size of the proofs (14) are polynomial in $|y|$. The height of the proof of (14) is bounded by $h' \cdot \log(|y|) + h'$ for $h' := (h_e + 1 + h_B + h_S + h_\vee + h_\wedge)$ and the number of $\Sigma_i^q \cup \Pi_i^q$-cuts on a branch is bounded by

$$(c_e + 1) \cdot \log\log(|y|) + (c_e + 1).$$

The other quantifier cases are also not hard and we omit them.

(Σ_i^b-PLIND rule)

$$\frac{B(\lfloor \frac{1}{2}a \rfloor), \Gamma \longrightarrow \Delta, B(a)}{B(0), \Gamma \longrightarrow \Delta, B(|t|)}$$

Let e be the proof of the upper sequent. Applying the induction hypothesis we have G_i^*-proofs:

$$\llbracket B(\lfloor \frac{1}{2}a \rfloor) \rrbracket, \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket, \llbracket B(a) \rrbracket$$

with height bounded by $h_e \cdot \log(|y|) + h_e$ and number of cuts along any branch bounded by $c_e \cdot \log\log(|y|) + c_e$. By making an appropriate substitution of the variables representing the bits of a for formulas computing the bits of the term $|t|$ we can derive formulas of the following type:

$$\llbracket B(MSP(|t|, k+1)) \rrbracket, \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket, \llbracket B(MSP(|t|, k)) \rrbracket \tag{15}$$

Now cutting formulas of this type against each other in a balanced fashion we can derive the desired formula $\llbracket B(0) \rrbracket, \llbracket \Gamma \rrbracket \longrightarrow \llbracket \Delta \rrbracket, \llbracket B(|t|) \rrbracket$. The size of these new proofs will be polynomial in $|y|$. Since we did the cuts in a balanced fashion this step will require fewer than additional $c_t \cdot \log\log(|y|) + c_t$ cuts along any branch where c_t is a fixed constant. Hence, the number of cuts along any branch of the proofs of (15) will be bounded by $c' \cdot \log\log(|y|) + c'$ where $c' := c_e + c_t$.. Finally, the heights of these new proofs is bounded by $h' \cdot \log(|y|) + h'$ where h' is $h_e + c_t$.

This completes the cases we will show and the proof. □

Remark 17 Notice the above proof goes through for any proof system of the form of L_i^* but with the restriction that the height of proofs be bounded by some function which grows superlinear in log of the size of a proof and with the restriction that the number of $\Sigma_i^q \cup \Pi_i^q$-cuts along any branch is by bounded by a function which grows superlinear in the log of the log of the size of the proof. The point in using L_i^* was that it gives us a fixed proof system which we can verify is formalizable in R_2^1.

9. Applications

In this section we give two applications of having a quantified propositional proof system for R^i_2. We first show that L^*_i is the strongest proof system for which R^i_2 can prove i-reflection. We then show the $\forall\Sigma^b_j$-consequences of R^i_2 are finitely axiomatized. The proofs of these statements follow the proofs of similar statements for T^i_2 and S^i_2 in [**14**].

Theorem 18. $(0 \leq j \leq i)$ *Let P be a proof system which can be represented by a Δ^b_1-predicate in R^1_2 and suppose $R^i_2 \vdash j\text{-}RFN(P)$. Then L^*_i j-polynomial simulates P provably in R^1_2. That is, $R^1_2 \vdash L^*_i \geq_j P$.*

Proof. From the hypothesis and Theorem 16, we have

$$R^i_2 \vdash\!\longrightarrow \big(L^*_i \vdash \big([\![P(d,A)]\!]^{|y|}_{q(|y|)} \wedge [\![A \in \Sigma^q_j]\!]^{|y|}_{q(|y|)} \wedge [\![Assign(\tau)]\!]^{|y|}_{q(|y|)} \supset [\![TRU_j(A,\tau)]\!]^{|y|}_{q(|y|)}\big)\big) \tag{16}$$

If R^1_2 proves d is a P-proof of $A \in \Sigma^q_j$ and τ is a truth assignment then by Theorem 16, R^1_2 proves

$$L^*_i \vdash\!\longrightarrow [\![P(d,A)]\!]^{|y|}_{q(|y|)}, \quad L^*_i \vdash\!\longrightarrow [\![A \in \Sigma^q_j]\!]^{|y|}_{q(|y|)}, \quad \text{and } L^*_i \vdash [\![Assign(\tau)]\!]^{|y|}_{q(|y|)}. \tag{17}$$

Thus, R^1_2 proves

$$L^*_i \vdash\!\longrightarrow [\![TRU_j(A,\tau)]\!]^{|y|}_{q'(|y|)} \tag{18}$$

where q' is a potentially bigger bounding polynomial than q that we use to absorb the slightly added height and number of cuts needed to derive this sequent from (16) and (17). Let w_A be the formula computing the translation of the root of z in $[\![(\exists z)(Evaltree^{\exists}_j(Sub(A),\tau) = z \wedge \beta(1,z) = \ulcorner\top\urcorner)]\!]^{|y|}_{q'(|y|)}$. Then R^1_2 proves

$$[\![(\exists z)(Evaltree^{\exists}_j(Sub(A),\tau) = z \wedge \beta(1,z) = \ulcorner\top\urcorner)]\!]^{|y|}_{q'(|y|)} \to w_A \equiv A \tag{19}$$

This can be proved by induction on the number of rounds of parallel computation of subformulas of A needed in the computation of $Evaltree^{\exists}_j(Sub(A),\tau)$.

Given the definition of the statement TRU_j and (19) it is not hard to see that R^1_2 proves

$$L^*_i \vdash [\![TRU_j(A,\tau)]\!]^{|y|}_{q''(|y|)} \to A$$

This implies R^1_2 proves $L^*_i \vdash A$. Hence, $R^1_2 \vdash L^*_i \geq_j P$. As the Σ^b_1-definable functions of R^1_2 are in $NC \subseteq P$ this simulation is polynomial time. □

Theorem 19. $(2 \leq j \leq i)$ *The $\forall\Sigma^b_j$-consequences of R^i_2 are finitely axiomatized.*

Proof. For $i \geq 2$ the theory R^i_2 contains S^1_2. We will axiomatize the $\forall\Sigma^b_j$-consequences of R^i_2 as $T := S^1_2 + j\text{-}RFN(L^*_i)$ The theory S^1_2 is finitely axiomatized with $\forall\Sigma^b_2$-formulas [**12**]. As we have already stated after Definition 7, the formula $j\text{-}RFN(L^*_i)$ is a $\forall\Sigma^b_j$-formula. So it suffices to show if A is a Σ^b_j-formula provable in R^i_2 then it is provable in T. By Theorem 16, R^1_2, and hence, S^1_2 proves

$(\forall y)(L_i^* \vdash [\![A]\!]^{|y|}_{q(|y|)})$ for some bounding polynomial q. So by Theorem 15, the theory T proves

$$(\forall y, \tau)(Assign(\tau) \supset TRU_j([\![A]\!]^{|y|}_{q(|y|)}, \tau))$$

Thus, by Lemma 4 we have $T \vdash A$. □

Corollary 20. *$(0 \leq j \leq i, 1 \leq m)$ R_2^1 proves $L_i^* \geq_j L_{i,m}^*$.*

Proof. Follows from Remark 14 and the above theorem. □

References

[1] B. Allen. Arithmetizing uniform NC. *Annals of Pure and Applied Logic*, 53:1–50, 1991.

[2] S. Bloch. On parallel hierarchies and R_k^i. Submitted Annals of Pure and Applied Logic., 1996.

[3] M. Bonet. *The Lengths of Propositional Proofs and the Deduction Rule*. PhD thesis, U.C. Berkeley, 1991.

[4] R P. Brent. The parallel evalutaion of general arithmetic expressions. *Journal of the Association of Computing Machinery*, 21(2):201–206, 1974.

[5] S.R. Buss. *Bounded Arithmetic*. Bibliopolis, Napoli, 1986.

[6] S.R. Buss. *Weak formal systems and connections to computational complexity*. 1988. Blue lecture notes for a course at UC Berkeley.

[7] S.R. Buss. Algorithms for boolean formula evaluation and tree contraction. In P. Clote and J. Krajíček, editors, *Arithmetic, Proof Theory and Computational Complexity*, pages 96–115. Oxford Science Publications, 1993.

[8] P. Clote and G. Takeuti. Bounded arithmetic for NC, Alogtime, L and NL. *Annals of Pure and Applied Logic*, 56:73–177, 1992.

[9] P. Clote and G. Takeuti. First order bounded arithmetic and small boolean circuit complexity classes. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 154–218. Birkhauser, 1995.

[10] S. A. Cook. Feasibly constructive proofs and the propositional calculus. In *Proceedings of the 7-th ACM Symposium on the Theory of Computation*, pages 83–97, 1975.

[11] S. A. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36–50, 1977.

[12] P. Hájek and P. Pudlák. *Metamathematics of First-Order Arithmetics*. Springer-Verlag, 1993.

[13] J. Johannsen. A note on sharply bounded arithmetic. *Archive for Mathematical Logic*, 33:159–165, 1994.

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

[15] J. Krajíček and P. Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 36:29–46, 1990.

[16] C. Pollett. *Arithmetic Theories with Prenex Normal Form Induction*. 1996. Document in preparation.

[17] P.M. Spira. On time hardware complexity of tradeoffs for boolean functions. In *Proceedings of the 4th Annual Hawaii Symposium Systems Science*, pages 525–527. Western Periodicals, 1971.

[18] G. Takeuti. *RSUV* isomorphisms. In P. Clote and J. Krajíček, editors, *Arithmetic, Proof Theory and Computational Complexity*, pages 364–386. Oxford Science Publications, 1993.

[19] G. Takeuti. Frege proof systems and TNC^0. In D. Leivant, editor, *Logic and Computational Complexity*, LNCS 960, pages 221–252. Springer-Verlag, 1995.

Department of Mathematics, University of California at San Diego, La Jolla, CA 92903

E-mail address: `cpollett@euclid.ucsd.edu`