Proof Complexity and Feasible Arithmetics

DIMACS
Series in Discrete Mathematics
and Theoretical Computer Science

Volume 39

Proof Complexity and Feasible Arithmetics

DIMACS Workshop
April 21–24, 1996

Paul W. Beame
Samuel R. Buss
Editors

NSF Science and Technology Center
in Discrete Mathematics and Theoretical Computer Science
A consortium of Rutgers University, Princeton University,
AT&T Labs, Bell Labs, and Bellcore

American Mathematical Society

This DIMACS volume contains papers from the DIMACS Workshop on Feasible Arithmetics and Length of Proofs, which was part of the DIMACS Special Year on Logic and Algorithms. The Workshop was held on April 21–24, 1996.

1991 *Mathematics Subject Classification.* Primary 03F20, 03F30, 03F50, 68Q15, 68R05, 03D15, 03–06, 68–06.

Library of Congress Cataloging-in-Publication Data

Proof complexity and feasible arithmetics : DIMACS workshop, April 21–24, 1996 / Paul W. Beame, Samuel R. Buss, editors.

p. cm. -- (DIMACS series in discrete mathematics and theoretical computer science, ISSN 1052-1798 ; v. 39)

Papers from the proceedings of a workshop held in Rutgers, N. J.

Includes bibliographical references.

ISBN 0-8218-0577-0 (alk. paper)

1. Proof theory---Congresses. 2. Constructive mathematics—Congresses. 3. Computational complexity--Congresses. I. Beame, Paul W., 1959– II. Buss, Samuel R. III. Series.

QA9.54.P75 1997

511.3—dc21 97-29122

CIP

Contents

Foreword

The DIMACS Workshop on "Feasible Arithmetics and Length of Proofs" held in April 1996 was part of DIMACS Special Year on Logic and Algorithms. We would like to express our appreciation to Paul Beame and Sam Buss for their efforts to organize and plan this successful workshop as well as editing this volume of papers.

The workshop was part of the broader Special Year on Logic and Algorithms program which focused on computer aided verification, finite models, and proof complexity. The special year encouraged collaborations among very different research communities and this volume records one of many workshops in which this was achieved. We also extend our thanks to Eric Allender, Robert Kurshan, and Moshe Vardi for their work over many months as special year organizers.

DIMACS gratefully acknowledges the generous support that makes these programs possible. The National Science Foundation, through its Science and Technology Center program, the New Jersey Commission on Science and Technology, DIMACS partners at Rutgers, Princeton, AT&T Labs, Bell Labs, and Bellcore generously supported the special year.

Fred S. Roberts
Director

Bernard Chazelle
Co-Director for Princeton

Stephen R. Mahaney
Associate Director for Research

Preface

The complexity of proofs in propositional logic and the properties of feasible theories of arithmetic are closely related and give important and strong connections between logical properties and the properties of computational complexity classes.

Defining the proof complexity of logical formulas requires the specification of a particular *proof system* that determines the expressive power of and the relationships between the objects that may be used in their proofs. In addition, a proof system includes an efficient mechanism for checking the validity of proofs. Within such a proof system, the proof complexity of a formula is typically measured as the amount of space required to write down a proof. The study of proof complexity seeks to understand the complexity of proofs of formulas in specific proof systems, classify the relative complexity of proofs of formulas within different proof systems, and to develop better proof systems. For example, a major open question in proof complexity, equivalent to the NP versus co-NP problem, is whether or not there is a proof system in which all propositional tautologies have polynomial-size proofs.

Feasible theories of arithmetic are first-order theories of arithmetic designed to have proof-theoretic strength closely corresponding to low-level computational classes from the polynomial-time or NC hierarchies. The most natural feasible theories of arithmetic have the property that the functions which are provably total in the theory are precisely the functions which are computable in some given natural complexity class. Furthermore, it is frequently the case that first-order proofs in feasible theories of arithmetic can be translated directly into propositional proofs. The primary goals of the study of feasible theories of arithmetic are to understand the computational implications of various proof-theoretic constructions and use these to characterize complexity classes, and to establish connections between the logical properties of feasible theories and open problems in computational complexity. It is hoped that the study of feasible theories will ultimately yield useful new complexity results.

Over the last dozen years there has been substantial progress in proof complexity and feasible arithmetic as the two have grown together and have also benefited from a very productive cross-fertilization of techniques with Boolean circuit complexity. This progress has significantly broadened and deepened many connections of proof complexity and feasible arithmetic with computational complexity and greatly enriched both areas.

The papers in this volume represent the proceedings of a workshop on "Feasible Arithmetic and Proof Complexity" held at the DIMACS Institute at Rutgers, New Jersey on April 21-24, 1996. The principal speakers at this workshop included J. Avigad, P. Beame, S. Bellantoni, S. Buss, J.-Y. Cai, A. Carbone, P. Clote, F. Ferreira, X. Fu, R. Impagliazzo, J. Johannsen, B. Kauffmann, J. Krajíček, D. Leivant,

A.A. Razborov, K. Regan, S. Riis, T. Pitassi, P. Pudlak, S. Rudich, G. Takeuti and D. Willard, and many of these speakers submitted talks to this volume. The papers primarily cover lower bounds on the complexity of propositional proofs and meta-mathematical results on feasible theories of arithmetics. All papers in this volume are refereed.

We wish to thank the DIMACS institute for their generous support in hosting and financing the workshop and the DIMACS staff for their help in organizing the meeting. We also thank the referees for the rigorous and thorough review of the articles.

Paul Beame and Sam Buss

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

Plausibly Hard Combinatorial Tautologies

Jeremy Avigad

ABSTRACT. We present a simple propositional proof system which consists of a single axiom schema and a single rule, and use this system to construct a sequence of combinatorial tautologies that, when added to any Frege system, p-simulates extended-Frege systems.

1. Introduction

As was pointed out in [6], the conjecture that $NP \neq coNP$ can be construed as the assertion that there is no proof system (broadly interpreted) in which there are short (polynomial-length) proofs of every propositional tautology. Though showing $NP \neq coNP$ seems to be difficult, the above formulation suggests an obvious restriction, namely the assertion that specific proof systems are inefficient. One of the nicest results of this form to date is the fact that there are no short proofs of tautologies representing the pigeonhole principle in a fixed-depth Frege system (see, for example, [1, 2]). This approach to demonstrating a proof system's inefficiency seems natural: choose a suitable sequence of propositional formulas that express some true combinatorial assertion, and then show that these tautologies can't be proven efficiently by the system in question.

Unfortunately, in the case of Frege systems, there is a shortage of good candidates. Bonet, Buss, and Pitassi [3] consider a number of combinatorial principles with short extended-Frege proofs, and conclude that most of them seem to require at most quasipolynomial Frege proofs (see also [7] for further discussion). This isn't to say that there are *no* examples of tautologies whose Frege proofs are likely to require exponential length: Cook [5] has shown that propositional tautologies $Con_{EF}(n)$, which assert the partial consistency of extended-Frege systems, have polynomial extended-Frege proofs; whereas Buss [4] has shown that if a Frege system is augmented by these tautologies, it can polynomially simulate any extended-Frege system. In short, if there is any separation between Frege and extended-Frege proof systems, the assertions $Con_{EF}(n)$ witness this separation.

1991 *Mathematics Subject Classification.* Primary 03F20; Secondary 03B05.

Key words and phrases. proof complexity, propositional logic, Frege system, hard tautologies.

The obvious complaint is that such consistency assertions can hardly be called "combinatorial," since they involve the coding of formulas, proofs, axioms, and rules of inference. However, the existence of such tautologies suggests that the problem is primarily one of esthetics: we have some plausibly hard tautologies, only they are not as natural as we would like them to be.

The point of this paper is to make the assertions $Con_{EF}(n)$ look a bit more combinatorial than they might seem at first glance. In the next section we present a surprisingly simple proof system, p-equivalent to any Frege system, which relies on a single logical connective, a single axiom, and a single rule. The consistency of this proof system then "translates" to somewhat combinatorial assertions regarding the hereditarily finite sets or directed acyclic graphs. It is well known that extended-Frege proof systems are essentially Frege systems in which formulas are represented by nodes of a circuit, and in Section 4 we show that the DAG formulation of the combinatorial assertion yields a sequence of tautologies that behave much like the propositions $Con_{EF}(n)$.

2. A Simple Proof System

We start by reviewing some definitions from [6] (see also [8]). A proof system F is said to be *implicationally complete* if whenever

$$\varphi_1, \varphi_2, \ldots, \varphi_n \models \psi$$

then ψ is derivable from $\varphi_1, \ldots, \varphi_n$ in F. A *Frege system* is an implicationally complete propositional proof system based on finitely many rules and axiom schemata. If F_1 and F_2 are two propositional proof systems then F_1 *p-simulates* F_2 if there is a polynomial time algorithm that translates any F_1 proof of a propositional formula ψ to an F_2 proof of the same formula. (This last definition can be modified to cover situations in which F_1 and F_2 are based on different sets of logical connectives.) In particular, if F_1 p-simulates F_2 then there is a polynomial bound to the increase in length of proof given by the translation.

The notion of a Frege system is extremely robust, in that any two Frege systems p-simulate each other, even if the underlying sets of connectives are different and connectives of variable arities are allowed. (This in stated in [6] and proven in [10]; for an alternative proof see [4].)

In this paper we will work with the single connective $nand(\varphi_1, \ldots, \varphi_k)$, where k is arbitrary. This connective asserts that not all of its arguments are true, or, equivalently, that at least one of them is false. It might be more accurate to think of *nand* as being a unary operator on sets, since, for example, we will treat $nand(\varphi_1, \varphi_2, \varphi_3)$, $nand(\varphi_3, \varphi_1, \varphi_2)$, and $nand(\varphi_1, \varphi_1, \varphi_2, \varphi_3)$ as the same formula. We will often use $\vec{\varphi}$ to denote a sequence of formulas $\varphi_1, \ldots, \varphi_k$.

It should be clear that *nand* is a complete connective, since one can define $\bot$ as $nand()$ and $\neg\varphi$ as $nand(\varphi)$, and then define the rest of the logical connectives in the usual way. Notice that propositions of the form

$$(Ax) \qquad nand(\vec{\psi}, \vec{\varphi}, nand(\vec{\varphi}))$$

are tautologically true, since if none of the φ_i are false, then $nand(\vec{\varphi})$ is. Similarly, the rule

$$(Cut) \qquad \frac{nand(\vec{\psi}, \vec{\varphi}) \quad nand(\vec{\psi}, nand(\vec{\varphi}))}{nand(\vec{\psi})}$$

is sound, since either all the φ_i are true, in which case the first premise guarantees the conclusion, or at least one of the φ_i is false, in which case $nand(\vec{\varphi})$ is true and the second premise guarantees the conclusion. Finally, the rule

$$(Weak) \qquad \frac{nand(\vec{\psi})}{nand(\vec{\psi}, \vec{\varphi})}$$

is sound, since the premise guarantees that at least one of the ψ_i is false. Notice that all the above rules are still valid even if the sets $\{\vec{\psi}\}$ and $\{\vec{\varphi}\}$ are not disjoint, or if either of these two sets is empty.

Let F_1 be the proof system consisting of (Ax), (Cut), and $(Weak)$, and let F_2 be the proof system consisting of just (Ax) and (Cut). We now have the following surprising

THEOREM 2.1. *F_1 is a Frege-system, and F_2 p-simulates F_1.*

The proof of this theorem is subsumed by the following sequence of lemmas.

LEMMA 2.2. *The rule*

$$(A) \qquad \frac{nand(\vec{\psi}, nand(\varphi_1)) \quad \ldots \quad nand(\vec{\psi}, nand(\varphi_l))}{nand(\vec{\psi}, nand(\varphi_1, \ldots, \varphi_l))}$$

is a derived rule of F_1.

PROOF. Roughly speaking, rule (A) asserts that one can derive $\theta \vee (\varphi_1 \wedge \ldots \wedge \varphi_l)$ from $\theta \vee \varphi_1$ through $\theta \vee \varphi_l$. For simplicity let's consider the case $l = 2$, namely

$$\frac{nand(\vec{\psi}, nand(\varphi_1)) \quad nand(\vec{\psi}, nand(\varphi_2))}{nand(\vec{\psi}, nand(\varphi_1, \varphi_2))}.$$

Using an axiom, the first premise, weakening, and cut, conclude

$$\frac{nand(\vec{\psi}, \varphi_1, \varphi_2, nand(\varphi_1, \varphi_2)) \quad \dfrac{nand(\vec{\psi}, nand(\varphi_1))}{nand(\vec{\psi}, nand(\varphi_1), \varphi_2, nand(\varphi_1, \varphi_2))}}{nand(\vec{\psi}, \varphi_2, nand(\varphi_1, \varphi_2))}.$$

By weakening the second premise, conclude

$$\frac{nand(\vec{\psi}, nand(\varphi_2))}{nand(\vec{\psi}, nand(\varphi_2), nand(\varphi_1, \varphi_2))}.$$

The desired conclusion,

$$nand(\vec{\psi}, nand(\varphi_1, \varphi_2))$$

follows from a cut. The general case is similar and left to the reader. □

LEMMA 2.3. *The following rule*

$$(B) \qquad \frac{nand(\vec{\psi}, \vec{\varphi})}{nand(\vec{\psi}, nand(nand(\vec{\varphi})))}$$

is a derived rule of F_1.

PROOF. Roughly speaking, rule (B) asserts that one can derive $\theta \vee (\varphi_1 \vee \ldots \vee \varphi_k)$ from $\theta \vee \varphi_1 \vee \ldots \vee \varphi_k$. By weakening the premise, conclude

$$nand(\vec{\psi}, \vec{\varphi}, nand(nand(\vec{\varphi}))).$$

The conclusion follows from a cut on the axiom

$$nand(\vec{\psi}, nand(\vec{\varphi}), nand(nand(\vec{\varphi}))).$$

□

LEMMA 2.4. *F_1 is complete.*

PROOF. Rules (A) and (B) are not only sound, but have the further property that if the conclusions are tautologies, then so are all the premises. The usual method of proving completeness by "working backwards" then applies: suppose α is a tautology of the form $nand(\vec{\varphi})$. If one of the φ_i is of the form $nand(\psi_1, \ldots, \psi_l)$ for $l > 1$ we can use the first rule to reduce the task of proving α to proving formulas in which the subformula φ_i is replaced by a unary *nand*. On the other hand, if one of the φ_i of the form $nand(\psi)$ and ψ is not a variable, then ψ must be of the form $nand(\vec{\theta})$ and we can use the second rule backwards to remove two layers of *nand*'s. Ultimately we are reduced to proving tautologies of the form $nand(\vec{\varphi})$, where each φ_i is either $nand()$, a variable, or the negation of a variable. But it is easy to see that such an assertion is a tautology iff either $nand()$ appears among the φ_i or some particular variable appears along with its negation, and in both cases the resulting tautology is an axiom. (Note that, in particular, $nand(nand())$ is an axiom.) □

LEMMA 2.5. *F_1 is implicationally complete.*

PROOF. Suppose $\varphi_1, \ldots, \varphi_k \models \psi$ where each φ_i is of the form $nand(\vec{\theta}_i)$ and ψ is of the form $nand(\vec{\eta})$. Then

$$nand(\vec{\theta}_1) \wedge \ldots \wedge nand(\vec{\theta}_k) \rightarrow nand(\vec{\eta})$$

is a tautology. Some fiddling shows that this formula is equivalent to

$$nand(nand(\vec{\theta}_1), \ldots, nand(\vec{\theta}_k), \vec{\eta}),$$

and the fact that F_1 is complete implies that this is derivable in F_1. The conlusion $nand(\vec{\eta})$ then follows from the assumptions

$$nand(\vec{\theta}_1), \ldots, nand(\vec{\theta}_k)$$

using k applications of weakening and cut. □

We can't quite say that F_2, which doesn't allow weakening, is implicationally complete; notice that in the previous lemma we had to weaken the assumptions $\varphi_1, \ldots, \varphi_k$. However, we have the following

LEMMA 2.6. *F_2 p-simulates F_1.*

PROOF. We need to give a polynomial time algorithm that removes instances of weakening from any F_2 proof d. Since F_1 is a Frege system, we can assume that its derivations are tree-like (see [7]). Notice that

1. any two successive applications of weakening can be collapsed to a single one;
2. if φ follows from ψ_1 and ψ_2 by the cut rule, then any weakening of φ follows from the appropriate weakenings of ψ_1 and ψ_2; and
3. any weakening of an axiom is again an axiom.

As a result we can effectively eliminate instances of weakening in d, working from the bottom up. Notice that if the original proof d has length n, then it has at most n steps. The new proof will also have at most n steps, none of which is any longer than the length of d itself. A moment's reflection should convince the reader that, under the assumption that the original proof is tree-like, the algorithm can be made to run in polynomial time. □

In Section 4 we will need to assume that our proofs have a nice normal form.

LEMMA 2.7. *Theorem 2.1 holds even if we assume that in the application of (Cut), the set $\{\vec{\varphi}\}$ is not contained in $\{\vec{\psi}\}$.*[1]

PROOF. If $\{\vec{\varphi}\}$ is contained in $\{\vec{\psi}\}$, the conclusion of the rule is the same as the left hypothesis. In that case we can just eliminate this instance of the rule, along with its right subtree. □

From now on we will assume that F_2 proofs are of this form. Notice that as a result, the arity of the conclusion of any cut rule is strictly less than the arity of either premise.

[1]The editor has pointed out that this lemma can be strengthened to require $\{\vec{\varphi}\}$ and $\{\vec{\psi}\}$ to be disjoint, albeit at the expense of a polynomial increase in the length of the proof.

3. A Theorem About Hereditarily Finite Sets

The hereditarily finite sets are defined inductively as follows: the empty set $\emptyset$ is a hereditarily finite set, and if $a_1, \ldots, a_k$ are hereditarily finite sets, then so is $\{a_1, \ldots, a_k\}$. There is a natural bijection between hereditarily finite sets and variable-free formulas built up with *nand*, given by the map f which takes sets $\{a_1, \ldots, a_k\}$ to formulas $nand(f(a_1), \ldots, f(a_k))$.

Andreas Blass has pointed out that every hereditarily finite set a corresponds to a two-player game, as follows. Player I starts by chosing any element $b \in a$, and player II must respond by chosing some element $c \in b$. Player I then plays some $d \in c$, and so on. The game continues until one player cannot move because the empty set has just been played, at which point this player has lost. The reader can verify that a closed formula φ is true if and only if Player I has a winning strategy in the corresponding game.

Formulas with variables can, of course, be identified with hereditarily finite sets over atoms $x_1, \ldots, x_k$. With this identification, the three rules from the previous section can be stated as follows:

1. Axiom: $a \cup b \cup \{b\}$
2. Cut: from $a \cup b$ and $a \cup \{b\}$ conclude a
3. Weakening: from a conclude $a \cup \{b\}$.

This observation allows us to cook up a somewhat combinatorial theorem about hereditarily finite sets.

DEFINITION 3.1. *Say a hereditarily finite set c is* good *if it is of the form $a \cup b \cup \{b\}$; that is, there is some $b \in c$ such that $b \subset c$.*

So good sets correspond to axioms.

THEOREM 3.2. *Let C be a hereditarily finite set, such that for every a in C, either*

1. *a is good, or*
2. *for some b not contained in a, $a \cup b$ and $a \cup \{b\}$ are both in C.*

Then the empty set is not in C.

PROOF. By hypothesis, if a is in C then either a is good or a is "derived" from two strictly larger sets $a \cup b$ and $a \cup \{b\}$. Given any set a in C we can then work backwards and construct a "derivation" of a from good sets. But such a derivation of the empty set would then translate back to a proof of $\bot$ in F_2.

To phrase the proof slightly differently, let m be the maximum of the cardinalities of the sets in C, and show by induction on i that every set of cardinality $m - i$ in C corresponds to a true formula. But the empty set φ corresponds to $nand()$, which is false. □

The task of translating Theorem 3.2 into a sequence of propositional tautologies will be addressed in the next section. Roughly speaking, if we code hereditarily finite sets as strings, our tautologies will "express" the consistency of the Frege system F_1. On the other hand, to "express" the consistency of a corresponding *extended* Frege system, we will want to code hereditarily finite sets as nodes of a directed acyclic graph, as follows.

If G is a directed acyclic graph and a is a vertex of G, define the *neighborhood* of a to be

$$N(a) = \{b \mid \text{there is an edge from } a \text{ to } b\}.$$

We can think of each node a as coding a hereditarily finite set, consisting of the elements coded by the neighborhood of a. (If we wanted our representation to be canonical, we could demand that G be extensional, but this extra requirement is unnecessary for our purposes.)

Translating Theorem 3.2 to this new language yields the following

THEOREM 3.3. *Let G be a directed acyclic graph, and suppose C is a subset of the vertices of C such that for every a in C, one of the following two conditions holds:*

1. *Either there is a vertex b in $N(a)$ such that $N(b) \subseteq N(a)$, or*
2. *there are vertices d and e in C, and a vertex b, such that*
   (a) *$N(d) = N(a) \cup \{b\}$,*
   (b) *$N(e) = N(a) \cup N(b)$, and*
   (c) *$N(e) \neq N(a)$.*

Then every element of C is nonterminal, that is, has at least one outgoing edge.

PROOF. The last condition is the analog of the requirement that b is not a subset of a, and guarantees that the cardinality of $N(e)$ and $N(d)$ are stricly bigger than that of $N(a)$. As in the case of Theorem 3.2, a counterexample would unwind to a proof of a contradiction in F_1. □

Note that a graph G and subset C with satisfying the hypothesis of the theorem above could lead to an F_1 proof whose length is exponential in the size of the graph, since the DAG representation of formulas allows one to "reuse" subformulas efficiently. On the other hand, small graphs G *do* translate to small *extended*-Frege proofs. As it turns out, in the next section we will only need the converse of this fact, i.e. that small extended-Frege proofs translate to small graphs.

The following lemma provides a more direct proof of Theorem 3.3.

LEMMA 3.4. *Let G be a directed acyclic graph. Then there is a set of vertices S such that for every a in G,*

1. *if a is in S, then $N(a) \cap \bar{S} \neq \emptyset$, and*
2. *if a is in $\bar{S}$, then $N(a) \subseteq S$.*

PROOF. Intuitively, S consists of the nodes of G that correspond to true formulas. The construction of S proceeds in stages, just as in the assignment of truth values to the nodes of a circuit: we start by putting all the terminal nodes of G into $\bar{S}$, and once all the elements of $N(a)$ have been put into S or $\bar{S}$, we decide what to do with a based on clauses 1 and 2 of the lemma. □

The theorem then follows from the lemma, just as in the proof of Theorem 3.2, by noting that elements which satisfy clause 1 of the theorem are guaranteed to be in S, and if c and d are in S, so is any a satisyfing the condition set by clause 2. At the same time, no terminal node can be in S.

In the next section we show how to translate Theorem 3.3 to propositional tautologies that behave much like the tautologies $Con_{EF}(n)$. It would be nice if something resembling this theorem could be found somewhere in the graph theory literature. Though the odds are slim, the following reformulation might be more suggestive.

THEOREM 3.5. *Let G be a directed graph, and suppose C is a set of nonterminal vertices of G satisfying the hypothesis of Theorem 3.3. Then G contains a cycle.*

4. Some Plausibly Hard Tautologies

Recall that an extended-Frege system (see [6, 7]) is obtained from a Frege system by allowing one to introduce constants A_φ at any point in a proof to abbreviate formulas φ, using the "extension axiom"

$$A_\varphi \leftrightarrow \varphi.$$

Note that φ may include other constants that have previously been introduced, and if "$\leftrightarrow$" is not among the basic connectives of the proof system, one can use any reasonable equivalent. The expectation is that these abbreviations "should" allow us to prove tautologies more efficiently, much the way that circuits "should" be able to represent boolean functions more efficiently than formulas. In fact, one can think of extended-Frege proofs as reasoning about formulas that are represented by nodes of a circuit, in a way we'll make explicit below.

To start with, let's fix an extended-Frege system EF_1, which consists of F_1 augmented by the extension axioms above. Define EF_2 to consist of F_2 augmented by axioms

$$nand(nand(A_\varphi), \varphi, \vec{\psi})$$

and

$$nand(nand(\varphi), A_\varphi, \vec{\psi}).$$

When added to F_1, these two axioms allow for short proofs of any weakening of an extension axiom of F_2, so just as in Theorem 2.1 we have that EF_2 p-simulates EF_1, and hence any extended-Frege system.

Given a proof in EF_1 (or EF_2), one can construct a directed acyclic graph that represents its formulas, in the following way.

DEFINITION 4.1. *Let d be an EF_1- or EF_2-proof with variables x_1 to x_k, and let Γ denote the set of all subformulas of formulas in d. Say that a directed acyclic graph G with distinguished terminal nodes $\hat{x}_1, \ldots, \hat{x}_k$* represents the subformulas of d *if there is a map h from Γ to G with the following properties:*

1. $h(x_i) = \hat{x}_i$;
2. $h(nand(\varphi_1, \ldots, \varphi_m))$ *is a node a such that*
$$N(a) = \{h(\varphi_1), \ldots, h(\varphi_n)\};$$
3. $h(A_\varphi) = h(\varphi)$.

If G represents the formulas of d via h, say that the subset

$$C = \{h(\varphi) \mid \varphi \text{ is a line of } d\}$$

represents the proof d *in* G.

It should be clear that for any proof d we can find a G representing its subformulas, such that the size of G is polynomial in the length of d. Notice that h maps extension axioms of EF_2 to "good" nodes, that is, nodes satisfying Clause 1 of Theorem 3.3.

Now it is not difficult to cook up, for each n, a tautology that expresses Theorem 3.3 for graphs of size n, such that the length of these tautologies is bounded by a polynomial in n. We can use variables p_{ij} for $i < j$ to represent the assertion that there is an edge from i to j (the condition $i < j$ guarantees that the graph is acyclic), and variables q_i to express the assertion that i is in C. The hypothesis of the theorem is then a conjunction

$$\bigwedge_i (q_i \to \varphi_1(i) \vee \varphi_2(i))$$

where $\varphi_1(i)$ is the assertion

$$\bigvee_j \left(p_{ij} \wedge \bigwedge_k (p_{jk} \to p_{ik}) \right)$$

and $\varphi_2(i)$ is the assertion

$$\bigvee_{j,k,l,m} \left(q_k \wedge q_l \wedge p_{kj} \wedge \bigwedge_{n \neq j} (p_{kn} \leftrightarrow p_{in}) \wedge \bigwedge_n (p_{ln} \leftrightarrow (p_{in} \vee p_{jn})) \wedge \neg p_{im} \wedge p_{jm} \right).$$

Here all the variables in the large conjunctions and disjunctions range from 1 to n, and by p_{ij} for $i \geq j$ we really mean $\bot$. The conclusion of the theorem, that is, the assertion that there is no terminal edge in C, translates to

$$\bigwedge_i (q_i \to \bigvee_j p_{ij}).$$

Call the resulting tautology $T(n)$.

We now come to the main theorems in this section.

THEOREM 4.2. *The tautologies $T(n)$ have polynomial-size extended-Frege proofs.*

PROOF. This can be proven in much the same way that one proves the equivalent result for $Con_{EF}(n)$, as in [5] or [7]; the set S of Lemma 3.4 can be defined in PV or V_1^1. □

THEOREM 4.3. *Let F be any Frege system, and let $\hat{F}$ be the proof system obtained by additonally allowing any substitution instance of a tautlogy $T(n)$. Then $\hat{F}$ p-simulates any extended-Frege system.*

PROOF. Our proof is modeled on the equivalent result for $Con_{EF}(n)$, which appears in [4, Main Theorem 3]. Both arguments rely heavily on the fact that Frege systems have an adequate formalization of the notion for truth for propositional formulas, which is the main innovation in the paper just cited.[2]

Given an extended-Frege proof d of $\varphi(x_1, \ldots, x_n)$, we need to show how to construct a proof of $\varphi(x_1, \ldots, x_n)$ in $\hat{F}$, whose length is polynomially bounded in the length of d.

Since EF_2 p-simulates any extended-Frege system, we can efficiently construct, from d, a "DAG proof" of $nand(nand(\varphi))$; that is, a directed acyclic graph G and a set C representing an EF_2 proof of that formula.

Given any assignment E of truth-values (that is, constants $nand()$ and $nand(nand())$ representing false and true, respectively) to the variables $x_1, \ldots, x_n$, let φ^E denote the closed formula $\varphi(E(\vec{x}))$. We now argue in F. Letting E be arbitrary, we can replace each node $\hat{x}_i$ of G by a graph representing $E(x_i)$, and obtain a DAG proof of $nand(nand(\varphi^E))$. On the other hand, if φ evaluated at E is false, we can construct (as in [4]) a DAG proof of $nand(\varphi^E)$; more precisely, there is a sequence of formulas defining a DAG proof of this formula, whose length is polynomial in the length of d, such that F can verify that this DAG proof satisfies the hypothesis of Theorem 3.3.[3] Gluing these two together with a cut, we obtain DAG proof of $nand()$, that is, a counterexample to Theorem 3.3.

In short, we can show with a polynomial-size proof in F that if φ^E is false, then a substitution instance of one of the tautologies $T(n)$ fails. As a result, $\hat{F}$ proves that φ is true under an arbitrary assignment of truth values to its variables. By the adequacy of the truth predicate, $\hat{F}$ can then conclude φ. □

COROLLARY 4.4. *If there is a superpolynomial separation between Frege systems and extended-Frege systems, then there are no polynomial size proofs of the tautologies $T(n)$ in any Frege system.*

[2]For the argument below, one can either adapt the truth predicates of [4] to handle *nand*'s of arbitrary arity, or just work with their infix equivalents in F.

[3]The argument is actually a bit more delicate than the one outlined in [4]. Since EF_2 doesn't allow weakening, we have to "anticipate" the weakening rules that we would have used in the proof described there. Working on $nand(\varphi)$ from the "top down," we can assign to each suformula ψ appropriate sequences $\vec{\theta}_{\psi,1}$ and $\vec{\theta}_{\psi,2}$; then, from the "bottom up," prove the following in F: "if ψ^E is true, there is a DAG proof of $nand(nand(\psi^E), \vec{\theta}^E_{\psi,1})$, and if ψ^E is false, there is a DAG proof of $nand(\psi^E, \vec{\theta}^E_{\psi,2})$."

PROOF. If a Frege system F had polynomial-size proofs of the tautologies $T(n)$, then substituting formulas for variables would yield polynomial-size proofs of arbitrary substitution instances of these tautologies. By the previous theorem, F would then p-simulate any extended-Frege system. □

5. Final Comments

The system F_2 is not the first axiomatization of propositional logic based on a single axiom and rule. In 1913 Sheffer showed that the binary nand "$\varphi \mid \psi$" is a complete connective, and soon after, Jean Nicod [9] showed that the axiom schema

$$\{[p \mid (q \mid r)] \mid [t \mid (t \mid t)]\} \mid \{[s \mid q] \mid [(p \mid s) \mid (p \mid s)]\}$$

combined with the rule

$$\frac{p \mid (r \mid q) \quad p}{q}$$

provide a complete axiomatization of propositional logic. (Here "complete" means that one can derive the axioms and rules of Russell and Whitehead's *Principia Mathematica*; the modern notion of completeness for propositional logic first appeared independently in the work of Bernays and Post, a few years later.) In the second edition of the *Principia*, which appeared in 1925, the authors cite this work, and, oddly enough, call Sheffer's reduction "the most definite improvement resulting from work in mathematical logic during the past fourteen years."

The fact that the proof system F_2 presented here p-simulates any Frege system tells us that without loss of generality we can think of any Frege proof as a tree, with a tautology at the bottom, cuts at the branches, and axioms ("the good sets") at the nodes. If would be nice if this simple formulation could be put to good use in proving lower bounds.

References

[1] Ajtai, Miklos, "The complexity of the pigeonhole principle," *Proceedings of the IEEE 29th Annual Symposium on Foundations of Computer Science* (1988), pp. 346-355.

[2] Beame, Paul, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, Pavel Pudlák, and Alan Woods, "Exponential lower bounds for the pigeonhole principle," *Proceedings of the 24th Annual ACM Symposium on Theory of Computing* (1992), pp. 200-221.

[3] Bonet, Maria, Samuel Buss, and Toniann Pitassi, "Are there hard examples for Frege systems?" in Clote and Remmel eds., *Feasible Arithmetic II*, Birkhauser, 1995.

[4] Buss, Samuel, "Propositional consistency proofs," *Annals of Pure and Applied Logic*, vol. 52 (1991), pp. 3-29.

[5] Cook, Stephen, "Feasibly constructive proofs and the propositional calculus," *Proceedings of the 7th Annual ACM Symposium on Theory of Computing* (1975), pp. 83-97.

[6] Cook, Stephen and Robert Reckhow, "The relative efficiency of propositional proof systems," *Journal of Symbolic Logic*, vol. 44 (1979), pp. 36-50.

[7] Krajíček, Jan, "On Frege and extended-Frege proof systems," in Clote and Remmel eds., *Feasible Arithmetic II*, Birkhauser, 1995.

[8] Krajíček, Jan, *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, Cambridge University, 1995.
[9] Nicod, Jean, "A reduction in the number of the primitive propositions of logic," Proceedings of the Cambridge Philosophical Society, vol. 19 (1917), pp. 32-41.
[10] Reckhow, Robert, *On the Lengths of Proofs in the Propositional Calculus*, Ph. D. thesis, University of Toronto, 1976.

Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA 15213
E-mail address: avigad+@cmu.edu

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

More on the Relative Strength of Counting Principles

Paul Beame and Søren Riis

Abstract. We give exponential size lower bounds for bounded-depth Frege proofs of variants of the bijective ('onto') version of the pigeonhole principle, even given additional axiom schemas for modular counting principles. As a consequence we show that for bounded-depth Frege systems the general injective version of the pigeonhole principle is exponentially more powerful than its bijective version. Furthermore this yields a slightly simpler proof of exponential separations between modular counting principles in bounded-depth Frege systems.

1. Introduction

Over the last several years, substantial progress has been made in the study of the complexity of propositional proof systems. A particularly noteworthy development in this effort has been the significant cross-fertilization between research on circuit complexity and research on propositional proof complexity. One area of application of circuit-complexity techniques in proof complexity has been in the study of constant-depth Frege systems, proof systems in the conventional axiom schema/inference rule format each of whose constituent propositional formulas has constant depth.

Ajtai [2] introduced circuit-complexity techniques to the study of constant-depth Frege systems and provided the first bound showing that the propositional pigeonhole principle did not have efficient constant-depth proofs. His arguments were simplified by Bellantoni, Pitassi, and Urquhart [9] and the complexity lower bound improved to exponential by Beame, Impagliazoo, Krajíček, Pitassi, Pudlák, and Woods [7, 16, 14].

Given these results, a natural question to ask is: What is the power of the proof system if it is augmented by axiom schemas for some family of tautologies that does not have efficient proofs? Tautologies for several combinatorial principles have been studied. Ajtai [3] showed that if the pigeonhole principle is added as an axiom schema then the $Count_2$ tautologies still do not have efficient proofs, where the $Count_p$ tautologies express the fact that there is no perfect partition of a set

1991 *Mathematics Subject Classification.* 03F20.
Key words and phrases. proof complexity, bounded-depth proof systems, pigeonhole principle, counting principles, Hilbert's Nullstellensatz.
Research supported by a Carlsberg grant and by NSF grant CCR-9303017.

M into blocks of size p, if $|M| \not\equiv 0 \pmod p$. Again, this bound was improved to exponential in [8, 18] and can be extended to arbitrary values of p. These arguments used the usual circuit-complexity techniques augmented by specialized combinatorial techniques to handle the new axiom schemas.

A natural next question (asked originally in [15]) was to examine the relative strength of the $Count_p$ principles for different values of p. Ajtai [1] first showed that, when p and q are distinct primes, proofs of $Count_q$ are not efficient even when given $Count_p$ as axiom schemas. To handle the $Count_p$ axioms Ajtai used a reworking of several ideas from the theory of representations of the symmetric group.

A different approach for handling the $Count_p$ axioms was taken by two other groups of researchers independently. This approach was a natural extension of the methods in [8, 18]. Riis [18] laid out a framework involving showing that certain 'exceptional forests' of decision trees do not exist. This problem was left unsolved in [18]. Working along similar lines, Beame, Impagliazzo, Krajíček, Pitassi, and Pudlák [6] introduced the notion of a Nullstellensatz proof system and reduced the existence question for objects similar to Riis' exceptional forests to the degree required for certain proofs in this system. They also showed lower bounds on this degree using Ramsey theory, and thus extended Ajtai's results to a somewhat wider class of p, q combinations. Riis [19] applied similar Ramsey theory arguments directly to the forests themselves.

One important contribution of [6] was to show that exponential lower bounds for $Count_q$ given $Count_p$ would follow from improved degree bounds for the Nullstellensatz proofs. These improved degree bounds were shown by Buss, Impagliazzo, Krajíček, Pudlák, Razborov, and Sgall [10] who introduced a nice inductive method for producing such bounds.

The present paper uses similar techniques to give a further refinement of our understanding of the strength of these combinatorial principles. Thus far, we have referred to 'the' pigeonhole principle. However, there are a number of variations of the pigeonhole principle depending on the sizes of the domain and range of the map and on whether or not the map is required to be 'onto' (which is a weaker version). The lower bounds mentioned above have applied to either version equally and have assumed that the domain is one element larger than the range. We show that the onto version of the pigeonhole principle from $n + p^{\lfloor \epsilon \log n \rfloor}$ points to n points, $onto\text{-}PHP_n^{n+p^{\lfloor \epsilon \log n \rfloor}}$, requires exponential size constant-depth proofs even given $Count_p$ as axiom schemas. The key feature of our argument is a new degree lower bound for Nullstellensatz proofs of $onto\text{-}PHP_n^{n+p^k}$. (Our results strengthen the results in [20] and are based on a substantially different presentation.)

Since $onto\text{-}PHP_n^{n+p^{\lfloor \epsilon \log n \rfloor}}$ does follow efficiently from axiom schemas for the general PHP_n^{n+1}, and additional axiom schemas for $Count_p$ do yield efficient proofs of $onto\text{-}PHP_n^{n+1}$, it follows that PHP_n^{n+1} requires exponential size constant depth proofs given axiom schemas for $onto\text{-}PHP_n^{n+1}$. Also, since additional axiom schemas for $Count_q$, for appropriate $q \neq p$ do give short proofs of $onto\text{-}PHP_n^{n+p^{\lfloor \epsilon \log n \rfloor}}$, the exponential separations between $Count_q$ and $Count_p$ principles are also corollaries of our results. The idea of examining the relationship between $onto\text{-}PHP_n^{n+p^{\lfloor \epsilon \log n \rfloor}}$ and $Count_p$ was originally used implicitly by Riis as the basis of the approach in [18] towards proving a separation between $Count_p$ and $Count_q$.

There has been a substantial improvement in the precision and presentation of the methods for proving lower bounds on constant-depth Frege systems with additional axiom schemas and the papers above do not give entirely self-contained explanations of the best of current techniques. In this paper we attempt to give as complete a presentation as possible.

We now outline the structure of the argument, giving references for the key techniques. We use the notion of a k-evaluation due to Krajíček, Pudlák, and Woods [14], incorporating the matching decision trees of Pitassi, Beame, Impagliazzo [16], and built for any small Frege proof using a switching lemma proved with the methods of Beame [4]. Then, as in the argument of Riis [18] and Beame and Pitassi [8], we show that having a k-evaluation implies the existence of a certain forest of matching decision trees. Following this we show, using a reduction analogous to that of Beame, Impagliazzo, Krajíček, Pitassi, and Pudlák [6], that the existence of such a forest implies a small degree Nullstellensatz refutation of an associated family of polynomials. Finally, the proof that such a small degree refutation does not exist is analogous to that of Buss, Impagliazzo, Krajíček, Pudlák, Razborov, and Sgall [10]. This last is the main new technical contribution and the reader who is familiar with the other aspects of this paper may wish to skip directly to section 8. In section 9 we combine the arguments from the previous sections to show our main results.

2. Frege Proofs and Counting Principles

A *Frege system* is a sound and implicationally complete propositional proof system with a finite number of axiom schemas and inference rules. The *size* of a proof in a Frege system is the total number of subformulas appearing in the proof. We consider formulas over the basis $\vee$ (binary) and $\neg$ with propositional variables and the constants 0 (false) and 1 (true) as atoms. We use $F \wedge G$ as a shorthand for $\neg(\neg F \vee \neg G)$ and $\bigvee_{i=1}^{r} F_i$ as a shorthand for an arbitrarily parenthesized tree of binary $\vee$'s with $F_1, \ldots, F_r$ at the leaves.

The *depth* of a formula F is the maximum number of runs of consecutive $\vee$ connectives on any path from an atom of F to the main connective of F. A *depth d Frege system* is a restriction of a Frege system to proofs all of whose formulas have depth at most d.

(One can define Frege systems over any basis of binary connectives and the sizes of proofs in these systems are polynomially related to each other. For constant depth d Frege systems this is also true, given the depth measure above, provided that one excludes connectives $\oplus$ and $\leftrightarrow$. We concentrate on connectives $\neg$ and $\vee$ for ease of presentation.)

Pigeonhole Principles: Let D and R be disjoint sets and consider propositional variables P_{ij}, $i \in D$, $j \in R$. There are 3 natural variations of the Pigeonhole Principle:

"There is no total injective relation on $D \times R$:"

$$PHP_R^D = (\bigvee_{i \in D} \neg \bigvee_{j \in R} P_{ij}) \vee \bigvee_{j \in R} \bigvee_{i \neq i' \in D} (P_{ij} \wedge P_{i'j}).$$

"There is no 1-1 function from D to R:"

$$fun\text{-}PHP_R^D = PHP_R^D \vee \bigvee_{i \in D} \bigvee_{j \neq j' \in R} (P_{ij} \wedge P_{ij'}).$$

These two are polynomially equivalent as may be seen by imposing an order on R and setting:

$$P'_{ij} = P_{ij} \wedge \bigvee_{j'<j} \neg P_{ij'}.$$

However, the following variant, as we will see, is strictly weaker than PHP^D_R:
"There is no 1-1 onto function from D to R:"

$$\textit{onto-PHP}^D_R = \textit{fun-PHP}^D_R \vee (\bigvee_{j\in R} \neg \bigvee_{i\in D} \neg P_{ij}).$$

When $|D| > |R|$ all the variations are tautologies and for the purposes of this paper we will assume that $|D| \geq |R|$. Clearly, each pigeonhole principle variation only depends on the sizes of D and R so we will usually refer to $PHP^{|D|}_{|R|}$, etc.

Counting Principles: Let M be any set and p be a positive integer. Let $M^{(p)}$ denote the set of all p-element subsets of M. The mod p counting principle over M is defined on the set of propositional variables Y_e, $e \in M^{(p)}$.
"There is no perfect p-partition of M."

$$\textit{Count}^M_p = (\bigvee_{v\in M} \bigwedge_{v\in e\in M^{(p)}} \neg Y_e) \vee \bigvee_{e,f\in M^{(p)},\ e\perp f} (Y_e \wedge Y_f)$$

where we write $e \perp f$ if $e \neq f$ and $e \cap f \neq \emptyset$. If $|M| \not\equiv 0 \pmod p$ then $\textit{Count}^M_p$ is a tautology. Again, the tautology really only depends on $|M|$, so we can refer to $\textit{Count}^m_p$. We will use $\textit{Count}_p$ to refer to the family of tautologies $\textit{Count}^m_p$ with $m \not\equiv 0 \pmod p$.

3. Restrictions and Matching Decision Trees

The main argument in this paper is a lower bound for the lengths of bounded-depth Frege proofs of *onto-PHP*, given *Count* formulas as axiom schemas. Therefore the propositional variables with which will primarily be concerned are those that appear in the *onto-PHP* formula. We introduce some notation for discussing formulas involving these variables.

Let $\mathcal{M}_{D\times R}$ be the set of all partial matchings on $D \times R$. A *matching term* A is $\bigwedge_{\langle i,j\rangle\in\pi} P_{ij}$ for some matching $\pi \in \mathcal{M}_{D\times R}$. A *matching disjunction* F is $\bigvee_i A_i$ where A_i are matching terms.

We say that i and j are the *endpoints* of P_{ij}. If Y is a term or a set of variables, then $v(Y)$ denotes the set of endpoints of variables in Y. We use $\mathrm{dom}(Y) = v(Y) \cap D$ and $\mathrm{range}(Y) = v(Y) \cap R$.

For $|R| = n$ and $|D| = n+m$, define $\mathcal{M}^\ell_{D\times R}$ to be the set of all partial matchings on $D \times R$, ρ which match all but ℓ nodes of R.

Every ρ in $\mathcal{M}^\ell_{D\times R}$ determines a unique partial assignment or *restriction*, r,

$$r(P_{ij}) = \begin{cases} 1 & \text{if } \langle i,j\rangle \in \rho \\ 0 & \text{if there is an } e' \in \rho \text{ such that } |v(e') \cap \{i,j\}| = 1 \\ * & \text{otherwise} \end{cases}$$

where $*$ indicates that P_{ij} is not assigned a value. If r is the restriction obtained from ρ, we will refer to both the restriction and the partial matching by ρ. For a Boolean formula F in the variables over $D \times R$ and a partial matching ρ, F restricted by ρ will be the formula in the variables unset by ρ that remains after

assigning values to the variables set by ρ; we denote this by $F\restriction_\rho$. Given a set $S \subseteq D \cup R$, let $S\restriction_\rho$ denote $S \setminus v(\rho)$.

Our key interest in the set of restrictions given by the partial matchings $\rho \in \mathcal{M}^\ell_{D\times R}$, is that for any such ρ, $\textit{onto-PHP}^D_R\restriction_\rho = \textit{onto-PHP}^{D\restriction_\rho}_{R\restriction_\rho}$.

We say that two partial matchings σ and τ are *compatible* if $\sigma \cup \tau$ is also a partial matching. When viewed as restrictions, we use the notation $\sigma\tau$ to denote the restriction defined by the partial matching $\sigma \cup \tau$.

DEFINITION 3.1. A *matching decision tree over* $D \times R$ is a rooted directed tree T whose internal nodes are labelled by elements of $D \cup R$, and whose leaves may be labelled by elements of some label set L so that:

1. (a) If the root of T is labelled by $i \in D$ then for each $j \in R$ there is one out-edge from the root labelled $\langle i,j\rangle$.
   (b) If the root of T is labelled by $j \in R$ then for each $i \in D$ there is one out-edge from the root labelled $\langle i,j\rangle$.
   (c) There are no other out-edges from the root of T.
2. $T^{\langle i,j\rangle}$ is a matching decision tree over $D' \times R'$ where $D' = D \setminus \{i\}$, $R' = R \setminus \{j\}$, and $T^{(i,j)}$ is the tree whose root is the node connected to the root of T by the edge labelled $\langle i,j\rangle$.

Define $\mathrm{Br}(T)$ to be the set of branches (root-leaf paths) in T and $\mathrm{Br}_a(T)$ to be the set of those branches in T with leaf label $a \in L$. The set of edge labels along any branch of T forms a partial matching. We identify a branch with its matching so we view $\mathrm{Br}(T)$ and $\mathrm{Br}_a(T)$ as sets of partial matchings. We say that $T \equiv a$ if and only if $\mathrm{Br}(T) = \mathrm{Br}_a(T)$.

LEMMA 3.2. *Let π be a matching and T be a matching decision tree over $D \times R$ such that $|\pi| + \mathrm{height}(T) \leq \min(|D|, |R|)$. Then*

(i) *there is a $\sigma \in \mathrm{Br}(T)$ compatible with π.*
(ii) *the tree $T\restriction_\pi$ obtained by contracting all edges of T whose label is in π and deleting all edges of T (and their associated subtrees) whose labels are not compatible with π is a matching decision tree over $D\restriction_\pi \times R\restriction_\pi$.*

PROOF. We prove part (ii) first by induction on the height of T: The base case when T is a single labelled vertex is trivial.

If the label of the root of T is touches π in edge $\langle i,j\rangle$ then $T\restriction_\pi = T^{(i,j)}\restriction_{\pi'}$ where $\pi = \pi' \cup \langle i,j\rangle$. We apply the inductive hypothesis to $T^{(i,j)}$ and π' over $(D \setminus \{i\}) \times (R \setminus \{j\})$ to obtain the desired result.

If the label $i \in D$ of the root of T is not touched by π then the tree $T\restriction_\pi$ consists of the root of T with an outedge labelled by $\langle i,j\rangle$ for each $j \in R\restriction_\pi$ and this reaches subtree $T^{\langle i,j\rangle}\restriction_\pi$. Apply the inductive hypothesis to each such $T^{\langle i,j\rangle}$ over $(D \setminus \{i\}) \cup (R \setminus \{j\})$ to obtain the desired result.

The case when the root label of T is $j \in R$ and is not touched by π is similar. Part (i) follows by observing that any branch in T that is contracted to a branch in $T\restriction_\pi$ suffices. □

DEFINITION 3.3. For any matching decision tree T with label set $L = \{0,1\}$, let T^c be the same tree as T except that the leaf labels 0 and 1 are reversed, i.e. $\mathrm{Br}_1(T^c) = \mathrm{Br}_0(T)$ and $\mathrm{Br}_0(T^c) = \mathrm{Br}_1(T)$. Matching decision tree T *represents* boolean formula or function f iff:

$$\forall \pi \in \mathrm{Br}(T),\ f\restriction_\pi \equiv \text{the leaf label of } \pi \text{ in } T.$$

Given matching decision tree T, the matching disjunction given by T is

$$Disj(T) = \bigvee_{\pi \in \mathrm{Br}_1(T)} \bigwedge_{(i,j)\in\pi} P_{ij}$$

Note that T represents $Disj(T)$ and that if T has height $\leq k$ then $Disj(T)$ has terms of size $\leq k$. Observe that $Disj(T^c)$ is *not* equivalent to the negation of $Disj(T)$ but that if T represents f then the tree T^c does represent $\neg f$.

4. k-Evaluations

DEFINITION 4.1. Let Γ be a set of formulas closed under subformulas. A *k-evaluation*, $\mathbb{T}$, of Γ is an association of a matching decision tree $\mathbb{T}(A) = T_A$ of height $\leq k$ with each formula $A \in \Gamma$ such that

(1) T_0 and T_1 are single nodes labelled 0 and 1, respectively, and $T_{P_{ij}}$ is the unique tree of height 1 querying i that represents P_{ij},
(2) $T_{\neg A} = T_A^c$.
(3) If the major connective of A is $\vee$ then write $A = \bigvee_{i\in I} A_i$ where the major connective of each A_i is not $\vee$. It must be the case that T_A represents $\bigvee_{i\in I} Disj(T_{A_i})$.

Let $\mathbb{T}$ be a k-evaluation of a set of formulas containing formula A. We say that *A k-evaluates to true (false) under* $\mathbb{T}$ if and only if $T_A \equiv 1$ (respectively $T_A \equiv 0$).

Let the *size* of an axiom/rule in a Frege system $\mathcal{F}$ be the maximum number of distinct subformulas in it.

LEMMA 4.2. *Let P be a proof in Frege system $\mathcal{F}$ whose rules have size at most s, augmented by $Count_p$ axiom schemas. Suppose that $sk \leq |R| \leq |D|$ and let $\mathbb{T}$ be a k-evaluation of the set of subformulas of P. If every $Count_p$ axiom in $\mathcal{P}$ k-evaluates to true under $\mathbb{T}$ then all formulas in $\mathcal{P}$ k-evaluate to true under $\mathbb{T}$.*

PROOF. By induction on the number of Frege axioms and inferences in P.

Consider a Frege axiom/inference in P:

$$\frac{A_1(B_1/p_1,\cdots,B_m/p_m),\ldots,A_\ell(B_1/p_1,\cdots,B_m/p_m)}{A_0(B_1/p_1,\cdots,B_m/p_m)}$$

where the inference rule $\mathcal{R}$ is:

$$\frac{A_1(p_1,\cdots,p_m),\ldots,A_\ell(p_1,\cdots,p_m)}{A_0(p_1,\cdots,p_m)}$$

and assume that each $A_i(B_1/p_1,\cdots,B_m/p_m)$ for $1 \leq i \leq \ell$ k-evaluates to true under $\mathbb{T}$. We now show that this also holds for $A_0(B_1/p_1,\ldots,B_m/p_m)$:

Let $\mathcal{A}$ be the set of distinct subformulas of $\mathcal{R}$ and let Γ be $\mathcal{A}(B_1/p_1,\cdots,B_m/p_m)$. By assumption $|\Gamma| \leq s$, say $\Gamma = \{A_0,\ldots,A_\ell,\ldots,A_j\}$ for $j < s$.

Let $\pi_0 \in \mathrm{Br}(T_{A_0})$. Since $sk \leq n$ we can apply the Lemma 3.2 to inductively find $\pi_i \in \mathrm{Br}(T_{A_i})$ compatible with $\pi_0\cdots\pi_{i-1}$ for $1 \leq i \leq j$. Therefore all the π_i are mutually compatible. Let $\pi = \pi_0\pi_1\cdots\pi_j \in M_n$.

Observe that for any $A_i \in \Gamma$, $Disj(T_{A_i})\restriction_\pi$ is the constant 0 or 1 and define $V : \Gamma \to \{0,1\}$ by $V(A) = Disj(T_{A_i})\restriction_\pi$. By the definition of k-evaluations, V is a consistent truth evaluation and by assumption

$$V(A_1) = \cdots = V(A_\ell) = 1.$$

Since the rule $\mathcal{R}$ is sound it follows that $V(A_0) = 1$, i.e.

$$Disj(T_{A_0})\restriction_\pi = 1.$$

Since π extends branch π_0 of T_{A_0}, the leaf label of π_0 must be 1 as required. □

On the other hand we show that the tree associated with the goal formula of the proof cannot k-evaluate to true.

LEMMA 4.3. *If $k+1 \leq |R| < |D|$ and $\mathbb{T}$ is a k-evaluation of a set of formulas closed under subformulas and containing onto-PHP_R^D then onto-PHP_R^D does not k-evaluate to true under $\mathbb{T}$.*

PROOF. In fact we show that every leaf of $T_{\text{onto-}PHP_R^D}$ has label 0. By definition of a k-evaluation it is necessary and sufficient to show that $\mathrm{Br}_1(T_A) = \emptyset$ for each disjunct A in PHP_R^D.

Case 1: $A = (P_{ij} \wedge P_{i'j}) = \neg(\neg P_{ij} \vee \neg P_{i'j}) = \neg B$

Let $\pi \in \mathrm{Br}(T_A)$. It is also in $\mathrm{Br}(T_B)$. Since T_B represents $Disj(T_{\neg P_{ij}}) \vee Disj(T_{\neg P_{i'j}})$, it suffices to show that π is compatible with some element in $\mathrm{Br}_1(T_{\neg P_{ij}})$ or in $\mathrm{Br}_1(T_{\neg P_{i'j}})$.

By definition $T_{\neg P_{ij}}$ has height 1 with root label i and all its leaves are labelled 1 except the one below the out-edge with label $\langle i,j\rangle$.

Since $k+1 \leq n$, $T_{\neg P_{ij}}\restriction_\pi$ is well-defined and consists of contractions of all branches compatible with π. If π does not contain $\langle i,j\rangle$ then some branch of $T_{\neg P_{ij}}$ other than $\langle i,j\rangle$ remains and this has leaf label 1.

If π does contain $\langle i,j\rangle$ then it does not contain $\langle i',j\rangle$ and we apply the same argument to $T_{\neg P_{i'j}}$.

Case 2: $A = \neg\bigvee_{j\in R} P_{ij}$

Similar to the previous case. Here, we show that $\pi \in \mathrm{Br}(T_A)$ is compatible with some element of $\mathrm{Br}_1(T_{P_{ij}})$ for some $j \in R$.

If π contains $\langle i,j\rangle$ for some $j \in R$ then every branch in $T_{P_{ij}}$ compatible with π will be in $\mathrm{Br}_1(T_{P_{ij}})$.

If π does not contain $\langle i,j\rangle$ for any $j \in R$ then let $j' \in R$ be unmatched by π (such a j' must exist). Since π matches neither i nor j' and $k+1 \leq |R| < |D|$, π is compatible with the $\langle i,j'\rangle$ branch of $T_{P_{ij'}}$ which is what we need.

Case 3: The other onto-PHP_R^D disjuncts are handled exactly as in Case 1. □

5. Building a k-evaluation

Given an Frege proof $\mathcal{P}$ of limited size and depth we wish to find a restriction ρ such that after ρ is applied we have a suitable k-evaluation for all the subformulas in $\mathcal{P}$. This is too hard to do in a single step. Instead, we inductively build restrictions and k-evaluations for all depth i subformulas in $\mathcal{P}$ for $i = 0,\ldots,d$. The following lemma permits us to build upon previous k-evaluations.

LEMMA 5.1. *If Γ is a set of formulas closed under subformulas and $\mathbb{T}$ is a k-evaluation of Γ over $D \times R$ and ρ is a restriction on S with $|\rho| + k \leq |R|$ then the map $\mathbb{T}^\rho$ given by*

$$T_F^\rho = \begin{cases} T_F\restriction_\rho & \text{if } \mathrm{Br}_1(T_F) \neq \emptyset \\ T_0 & \text{if } \mathrm{Br}_1(T_F) = \emptyset \end{cases}$$

is a k-evaluation of $\Gamma\restriction_\rho$ over $(D \times R)\restriction_\rho$.

PROOF. Note that for any matching decision tree T and formula F, if T represents F over $D \cup R$ then $T\restriction_\rho$ represents $F\restriction_\rho$ over $(D \cup R)\restriction_\rho$. Also note that for any matching decision tree T,

$$Disj(T)\restriction_\rho = Disj(T\restriction_\rho).$$

From this the Lemma follows easily by induction. (The extra condition when $\mathrm{Br}_1(T_F) = \emptyset$ is to make sure that $T_{P_{ij}}\restriction_\rho = T_0$ when $P_{ij}\restriction_\rho = 0$.) □

The construction of decision trees for the higher level formulas of the proof uses the probabilistic method. The following so-called 'Switching Lemma' is the basis for that construction. We prove it as Lemma 5.4 below.

LEMMA 5.2. *Let F be an r-disjunction over $D \times R$ with $|R| = n$ and $|D| = n + m$. If $s \geq 0$ and $10m \leq \ell \leq (n/r)^{1/2}/10$ then, for ρ chosen uniformly at random from $\mathcal{M}^\ell_{D\times R}$, the probability that there does not exist a decision tree T over $(D \times R)\restriction_\rho$ of height less than s representing $F\restriction_\rho$ is less than $(1.5\ell^2\sqrt{r/n})^s$.*

LEMMA 5.3. *Let $|R| = n$, $|D| = n + m$. Let $n_0 = n$, $n_{i+1} = (n_i/9\log_2 S)^{1/4}$ for $i \geq 0$ and suppose that $n_d \geq \max\{10m, \log_2 S\}$. For any Frege proof $\mathcal{P}$ of size at most S and depth at most d in the pigeonhole variables on $D \times R$ there exists a restriction $\rho \in \mathcal{M}^{n_d}_{D\times R}$ such that there is a $\log_2 S$-evaluation $\mathbb{T}$ of the set of subformulas of $\mathcal{P}\restriction_\rho$ over $(D \times R)\restriction_\rho$.*

PROOF. Let $k = \log_2 S$. We construct a sequence of restrictions $\rho_0, \ldots, \rho_d = \rho$ and maps $\mathbb{T}^0, \ldots, \mathbb{T}^d = \mathbb{T}$ such that for each $i = 0, \ldots, d$, $|R\restriction_{\rho_i}| = n_i$ and $\mathbb{T}^i$ is a k-evaluation of the set of formulas in $\mathcal{P}_i\restriction_{\rho_i}$, where $\mathcal{P}_i$ is the set of subformulas of depth at most i in $\mathcal{P}$. We only specify trees for unnegated formulas at each depth since negations do not add to depth and if we have a tree T_F then we easily have a tree $T_{\neg F} = T_F^c$ of the same height.

Base Case: $i = 0$. Let ρ_0 be the empty restriction. The only nodes of depth 0 are inputs and their negations. For each literal P_{ij}, let $T^0_{P_{ij}}$ be a tree of height 1 that queries i and has its only leaf label 1 on the node reached by edge labelled $\langle i, j\rangle$. Let T^0_b be a single node labeled b for $b = 0, 1$.

Induction Step: Now suppose that after ρ_i is applied we have a k-evaluation $\mathbb{T}^i$ of $\mathcal{P}_i\restriction_{\rho_i}$. We wish to find a π such that $\rho_{i+1} = \rho_i\pi$ and extend T^i to a k-evaluation $\mathbb{T}^{i+1}$ of $\mathcal{P}_{i+1}\restriction_{\rho_{i+1}}$.

Now for any choice of $\pi \in \mathcal{M}^{n_{i+1}}_{(D\times R)\restriction_{\rho_i}}$ and any $A \in \mathcal{P}_i$, using Lemma 5.1 we can define $T^{i+1}_A = (T^i)^\pi_A$. Observe that for such $A \in \mathcal{P}_i$, $Disj(T^{i+1}_A) = Disj(T^i_A\restriction_\pi)$ which is a k-disjunction.

It remains to choose π and define T^{i+1}_A for $A \in \mathcal{P}_{i+1} \setminus \mathcal{P}_i$ of the form $A = \bigvee_j A_j$ where $A \in \mathcal{P}_{i+1} \setminus \mathcal{P}_i$. and each $A_j \in \mathcal{P}_i$. We consider π chosen at random from $\mathcal{M}^{n_{i+1}}_{(D\times R)\restriction_{\rho_i}}$. By Lemma 5.2, the probability that π does not admit a choice for T^{i+1}_A is

$$< 1.5n^2_{i+1}\sqrt{(\log_2 S)/n_i})^{\log_2 S} = 2^{-\log_2 S} = 1/S.$$

Since $|\mathcal{P}_{i+1} \setminus \mathcal{P}_i| \leq S$ the probability that some choice of π works for all formulas in $\mathcal{P}_{i+1} \setminus \mathcal{P}_i$ is strictly less than 1. We choose this π, fix $\rho_{i+1} = \rho_i\pi$ and set T^{i+1}_A according to that π for all $A \in \mathcal{P}_{i+1} \setminus \mathcal{P}_i$. The conditions for $\mathbb{T}^{i+1}$ are clearly satisfied. □

We assume that there is a total order on the elements of $D \cup R$ with all elements of D preceding those of R. Let $K \subseteq D \cup R$ and define

$$\mathcal{M}_{D\times R}(K) = \{\pi \mid K \subseteq v(\pi) \text{ and } \forall e \in \pi.v(e) \cap K \neq \emptyset\},$$

i.e., all minimal partial matchings over $D \times R$ which involve all of the elements of K.

We define the *complete matching tree* for $K \subseteq D \cup R$ over $D \times R$ as a matching decision tree over $D \times R$ with no leaf labels. It is the unique tree T such that $\mathrm{Br}(T) = \mathcal{M}_{D\times R}(K)$ and the query at each node v is the smallest element of K that is not an endpoint of the matching associated with the path from the root to v.

Given a disjunction F over D, assume that F has a total order on its terms and an order on the variables within each term. A restriction ρ is applied to F in order, so that $F\restriction_\rho$ is the disjunction whose terms consist of those terms of F that are not falsified by ρ, each shortened by removing any variables that are satisfied by ρ, and taken in the order of occurrence of the original terms on which they are based.

The *canonical decision tree for F over $D \times R$*, $T_{D\times R}(F)$ is defined inductively as follows:

1. If F is the constant function 0 or 1 (contains no terms or has an empty first term, respectively) then $T_{D\times R}(F)$ consists of a single leaf node labelled by the appropriate constant value.
2. If the first term C_1 of F is not empty then let F' be the remainder of F so that $F = C_1 \vee F'$. Let $K = v(C_1)$. We start with the complete matching tree for K. The paths of this tree correspond exactly to elements of $\mathcal{M}_{D\times R}(K)$. Let v_σ be the leaf node corresponding to the path labelled by $\sigma \in \mathcal{M}_{D\times R}(K)$. To obtain $T_{D\times R}(F)$, for each σ we replace the leaf node, v_σ, by the subtree $T_{(D\times R)\restriction_\sigma}(F\restriction_\sigma)$. (Note that for the unique element $\sigma \in \mathcal{M}_{D\times R}(K)$ which satisfies C_1 the leaf label of v_σ will be 1. For all other choices of σ, $T_{(D\times R)\restriction_\sigma}(F\restriction_\sigma) = T_{(D\times R)\restriction_\sigma}(F'\restriction_\sigma)$.)

$T_{D\times R}(F)$ clearly represents F over $D \times R$. We'll show that for appropriately chosen restriction ρ the height of $T_{D\times R}(F\restriction_\rho)$, $|T_{D\times R}(F\restriction_\rho)|$, is small with high probability. This lemma is a switching lemma in the spirit of [13] because it will allow us to obtain a disjunction that approximates the negation of F by representing F by a matching decision tree T and then taking $Disj(T^c)$.

LEMMA 5.4. *Let F be an r-disjunction over $D \times R$ with $|R| = n$ and $|D| = n + m$. If $s \geq 0$ and $10m \leq \ell \leq (n/r)^{1/2}/10$ then*

$$\frac{|\{\rho \in \mathcal{M}^\ell_{D\times R} \;:\; |T_{(D\times R)\restriction_\rho}(F\restriction_\rho)| \geq s\}|}{|\mathcal{M}^\ell_{D\times R}|} \leq (1.5\ell^2\sqrt{r/n})^s.$$

PROOF. We only need to consider $s > 0$. Let $S \in \mathcal{M}^\ell_{D\times R}$ be the set of restrictions ρ such that $|T_{(D\times R)\restriction_\rho}(F\restriction_\rho)| \geq s$. As in [4] we obtain a bound on $|S|/|\mathcal{M}^\ell_{D\times R}|$ by defining a 1-1 map from S to a small set.

Let $stars(r, s)$ to be the set of all sequences $\beta = (\beta_1, \ldots, \beta_k)$ such that for each j, $\beta_j \in \{*, -\}^r \setminus \{-\}^r$ and such that the total number of *'s in all the β_j is s. We will define a 1-1 map

$$S \;\to\; \bigcup_{s/2\leq j\leq s} \mathcal{M}^{\ell-j}_{D\times R} \times stars(r, j) \times [1, \ell + m]^s$$

The map: Let $F = C_1 \vee C_2 \vee \ldots$. Suppose that $\rho \in S$ and let π be the partial matching labelling the lexicographically first path in $T_{(D\times R)\restriction_\rho}(F\restriction_\rho)$ that has length $\geq s$. Trim the last few edges of π along the path from the root so that $|\pi| = s$. We use the formula F and π to determine the image of ρ. Let C_{ν_1} be the first term of F that is not set to 0 by ρ. Then $C_{\nu_1}\restriction_\rho$ will be the first term in $F\restriction_\rho$. Since $|\pi| > 0$, such a term must exist and is not the empty term. Let $K = v(C_{\nu_1}\restriction_\rho)$ and let σ_1 be the unique partial matching in $\mathcal{M}_{(D\times R)\restriction_\rho}(K)$ that satisfies $C_{\nu_1}\restriction_\rho$. Let π_1 be the portion of π that touches K. We have two cases based on whether or not $\pi_1 = \pi$.

1: If $\pi_1 \neq \pi$ then by the construction of π, $\pi_1 \in \mathcal{M}_{(D\times R)\restriction_\rho}(K)$. Note also that $C_{\nu_1}\restriction_{\rho\sigma_1} = 1$ but since $\pi_1 \neq \pi$, $\pi_1 \neq \sigma_1$, and thus $C_{\nu_1}\restriction_{\rho\pi_1} = 0$.
2: If $\pi_1 = \pi$ then it is possible that $v(\pi)$ does not contain all of K. In this case we shorten σ_1 so that it is the unique element of $\mathcal{M}_{(D\times R)\restriction_\rho}(K')$ that does not falsify $C_{\nu_1}\restriction_\rho$ where $K' = v(\pi_1) \cap K$.

Note that in either case $|\pi_1| \leq 2|\sigma_1|$.

We define β_1 to be a vector of length r based on the fixed ordering of the variables in term f_{ν_1}. The j-th component of β_1 is $*$ if and only if the j-th variable in C_{ν_1} is in $v(\sigma_1)$. Note that since $C_{\nu_1}\restriction_\rho$ is not the empty term then there is at least one $*$ in β_1. From C_{ν_1} and β_1 we can reconstruct σ_1.

Now, by the definition of $T_{(D\times R)\restriction_\rho}(F\restriction_\rho)$, $\pi \setminus \pi_1$ labels a path in the canonical tree $T_{(D\times R)\restriction_{\rho\pi_1}}(F\restriction_{\rho\pi_1})$. If $\pi_1 \neq \pi$, we repeat the above argument, with $\pi \setminus \pi_1$ in place of π, $\rho\pi_1$ in place of ρ and find a term C_{ν_2} which is the first term of F not set to 0 by $\rho\pi_1$. Based on this we generate π_2, σ_2, β_2, as before. We repeat this process until the round k in which $\pi_1\pi_2...\pi_k = \pi$.

For each i, π_i matches all elements of $v(\sigma_i)$, so the $\sigma_1, \ldots, \sigma_k$ are mutually compatible and thus $\sigma_1...\sigma_k = \sigma_1 \cup \cdots \cup \sigma_k$. The image of ρ under the 1-1 map we define is a triple, $\langle \rho\sigma_1...\sigma_k, (\beta_1, ..., \beta_k), \delta\rangle$ where δ is defined below. Let $\sigma = \sigma_1...\sigma_k$ and $j = |\sigma|$. Clearly $\rho\sigma = \rho\sigma_1...\sigma_k \in \mathcal{M}^{\ell-j}_{D\times R}$ and $(\beta_1, ..., \beta_k) \in stars(r, j)$.

We now define the information δ. This will encode the relationships between all the σ_i and π_i. The set $v(\pi_i)$ contains $v(\sigma_i)$ possibly together with some nodes unset by $\rho\sigma$, each of which must be connected by π_i to some element of $v(\sigma_i)$. We list the edges of π_i using the total order induced on $v(\sigma_i)$ by the order on the elements of $D \cup R$. For each vertex of $\mathrm{dom}(\sigma_i)$ in order, we list the name of the other node to which it is matched in pi_i. This endpoint can be one of $|\sigma_i| + \ell - j \leq \ell$ possibilities (all of which are known) so a number between 1 and ℓ is sufficient to encode this using the order induced on these vertices. After this, for each vertex of $\mathrm{range}(\sigma_i)$ in order, that is not matched so far, we can similarly give a number between 1 and $\ell - j + m \leq \ell + m$ to indicate its mate in D. The information δ is then simply the vector of these numbers, one per edge of π and is thus contained in $[1, \ell+m]^s$. Thus the image of the map is as required.

Inverting the map: It remains to show that the map we have just defined is indeed 1-1. To do this we show how to recover ρ from its image. The reconstruction is iterative. In the general stage of the reconstruction we will have recovered $\pi_1, ..., \pi_{i-1}$, $\sigma_1, ..., \sigma_{i-1}$, and will have constructed $\rho\pi_1...\pi_{i-1}\sigma_i...\sigma_k$. Recall that for $i < k$, $C_{\nu_i}\restriction_{\rho\pi_1...\pi_{i-1}\sigma_i} = 1$ and $C_j\restriction_{\rho\pi_1...\pi_{i-1}\sigma_i} = 0$ for all $j < \nu_i$. This clearly also holds when we append $\sigma_{i+1}...\sigma_k$ to the restriction. When $i = k$, something similar occurs except the only guarantee is that $C_{nu_i}\restriction_{\rho\pi_1...\pi_{k-1}\sigma_k} \neq 0$. Thus we can recover ν_i as the index of the first term of F that is not set to 0 by $\rho\pi_1...\pi_{i-1}\sigma_i...\sigma_k$.

Now, based on C_{ν_i} and β_i we can determine σ_i. Since we know $\sigma_1, ..., \sigma_i$ we can examine the entries in the vector δ associated with each of the vertices in $v(\sigma_i)$. At this point, although $\sigma_{i+1}, ..., \sigma_k$ are still undetermined, π_i can still be determined since π_i does not touch any of the vertices these restrictions touch.

We can now change $\rho\pi_1...\pi_{i-1}\sigma_i...\sigma_k$ to $\rho\pi_1...\pi_{i-1}\pi_i\sigma_{i+1}...\sigma_k$ using the knowledge of π_i and σ_i. Finally, given all the values of the π_i we can reconstruct ρ.

The numbers: Now we compute the value $|S|/|\mathcal{M}^\ell_{D\times R}|$. We can describe an element of $\mathcal{M}^\ell_{D\times R}$ by choosing ℓ elements of R and then, for each of the $n-\ell$ remaining vertices in turn, choosing an element of D with which it is to be matched. Thus $|\mathcal{M}^\ell_{D\times R}| = \binom{n}{\ell}(n+m)^{(n-\ell)} = \frac{n^{(\ell)}(n+m)^{(n-\ell)}}{\ell!}$ and

$$\begin{aligned}\frac{|\mathcal{M}^{\ell-j}_{D\times R}|}{|\mathcal{M}^\ell_{D\times R}|} &= \frac{n^{(\ell)}(n+m)^{(n-\ell)}(\ell-j)!}{(n-j)^{(\ell-j)}(n+m)^{(n-\ell+j)}\ell!} \\ &= \frac{(\ell+m)^{(j)}\ell^{(j)}}{(n-\ell)^{(j)}} \\ &\leq \left(\frac{(\ell+m)\ell}{n-\ell}\right)^j\end{aligned}$$

There is an easy bound of $|stars(r,s)| \leq 2^{s-1}r^s$ but we can also prove:

Claim: $|stars(r,s)| < (r/\ln 2)^s$.

For convenience in the proof we shall include the empty string in $stars(r,0)$ which would otherwise be empty. We shall show by induction on s that $|stars(r,s)| \leq \gamma^s$ for $(1+1/\gamma)^r = 2$; the statement of the lemma follows by using $1 + x < e^x$ for $x \neq 0$.

The base case $s = 0$ follows trivially. Now suppose that $s > 0$. It is easy to see from the definition that for any $\beta \in stars(r,s)$, if β_1 has $i \leq s$ $*$'s then $\beta = (\beta_1, \beta')$ where $\beta' \in stars(r, s-i)$. (For $i = s$ we have used our augmentation of $stars(r,0)$.) There are $\binom{r}{i}$ choices of β_1 so

$$\begin{aligned}|stars(r,s)| &= \sum_{i=1}^{\min(r,s)}\binom{r}{i}|stars(r,s-i)| \\ &\leq \sum_{i=1}^{r}\binom{r}{i}\gamma^{s-i} \\ &= \gamma^s\sum_{i=1}^{r}\binom{r}{i}(1/\gamma)^i \\ &= \gamma^s[(1+1/\gamma)^r - 1] \\ &= \gamma^s\end{aligned}$$

by the inductive hypothesis and the definition of γ. Thus the claim is proved.

Now applying our bounds we obtain

$$\begin{aligned}\frac{|S|}{|\mathcal{M}^\ell_{D\times R}|} &\leq \sum_{s/2\leq j}\frac{|\mathcal{M}^{\ell-j}_{D\times R}|}{|\mathcal{M}^\ell_{D\times R}|}\cdot|stars(r,j)|\cdot(\ell+m)^s \\ &\leq (\ell+1)^s\sum_{j\geq s/2}\left(\frac{r\ell(\ell+m)}{(n-\ell)\ln 2}\right)^j\end{aligned}$$

Since $1/\ln 2 < 1.4427$ and $10m \le \ell \le \sqrt{n/r}/10$, we have $(\ell + m) \le 1.1\ell$ and $n \le 1.02(n - \ell)$. Thus the series is at most

$$(1.1\ell)^s \sum_{j \ge s/2} (1.7r\ell^2/n)^j.$$

This is a geometric series with ratio $< .02$. Therefore it is at most

$$1.03(1.1\ell)^s(1.7r\ell^2/n)^{s/2} \le (1.5\ell^2\sqrt{r/n})^s.$$

□

6. Generic systems and Exceptional Forests

Suppose that we have a k-evaluation $\mathbf{T}$ of the subformulas in a Frege proof $\mathcal{P}$ of *onto-PHP*$_R^D$ with *Count*$_p$ axiom schemas. By Lemmas 4.2 and 4.3, there is some instance F of a *Count*$_p$ axiom in $\mathcal{P}$ that does not evaluate to true under $\mathbf{T}$. Therefore there is some $\pi \in \mathrm{Br}_0(T_F)$.

By Lemma 5.1, the map, $\mathbf{T}'$, given by

$$T'_A = \begin{cases} T_A \restriction_\pi & \text{if } \mathrm{Br}_1(T_A) \neq \emptyset \\ T_0 & \text{otherwise} \end{cases}$$

is also a k-evaluation of the formulas in F over $(D \times R)\restriction_\pi$ and T'_F is "false".

Let F_e for $e \in M^{(p)}$, $|M| \not\equiv 0 \pmod p$ be the formulas that substitute for Y_e in F and let $T_e = T'_{F_e}$. Using the T_e, we will see that if they 'locally' appear to define something that is a p-partition of M then $T'_F \equiv 1$. Then we show that it is impossible for the T_e to describe something that locally does appear to be a partition of M into blocks of size p. This latter is achieved by a reduction to a problem over polynomials.

LEMMA 6.1. *Suppose that some $\pi \in \mathrm{Br}_0(T_F)$ exists and $3k \le |R| < |D|$.*

(a) *If $e, e' \in M^{(p)}$ with $e \perp e'$ there are no compatible branches $\sigma_e \in \mathrm{Br}_1(T_e)$ and $\sigma_{e'} \in \mathrm{Br}_1(T_{e'})$.*

(b) *For any restriction τ such that $|\tau| + k \le |R|$, τ is compatible with some element of $\bigcup_{e \in M^{(p)}} \mathrm{Br}_1(T_e)$.*

PROOF. For part (a), suppose that there are compatible branches $\sigma_e \in \mathrm{Br}_1(T_e)$ and $\sigma_{e'} \in \mathrm{Br}_1(T_{e'})$. Let $\sigma = \sigma_e \cup \sigma_{e'}$ and apply it to all formulas in the k-evaluation $\mathbf{T}'$. We see that it will make $T'_{\neg F_e \vee \neg F_{e'}}\restriction_\sigma \equiv 0$. Thus $T'_{\neg(\neg F_e \vee \neg F_{e'})}\restriction_\sigma \equiv 1$ so $T'_F\restriction_\sigma = T'_F \equiv 1$ which is a contradiction.

For part (b), apply τ to all formulas in the k-evaluation $\mathbf{T}'$. If τ is incompatible with all elements of $\bigcup_{e \in M^{(p)}} \mathrm{Br}_1(T_e)$. then $T'_{\bigvee_{e \in M^{(p)}} F_e}\restriction_\tau \equiv 0$ so $T'_{\neg \bigvee_{e \in M^{(p)}} F_e}\restriction_\tau \equiv 1$ and thus $T'_F\restriction_\tau = T'_F \equiv 1$ which is a contradiction. □

Now for each $v \in M$ let $\mathcal{B}_v = \bigcup_{v \in e} \mathrm{Br}_1(T_e)$. Lemma 6.1 implies that any $\mathcal{B}_v$ consists of mutually incompatible elements and it contains an element compatible with any fixed τ with $|\tau| \le |R| - k$. In the terminology of [20] this is a $(|R|-k)$-*basis of height* $\le k$.

DEFINITION 6.2. A (p, M)-*generic system of height h over $D \times R$* is a collection of matching decision trees over $D \times R$: T_v, $v \in M$, with leaf labels that are p-subsets of M such that:

(1) each T_v has height at most h;
(2) each branch in T_v with leaf label e has $v \in e$;
(3) for all $e \in M^{(p)}$, for all $v, w \in e$, $\mathrm{Br}_e(T_v) = \mathrm{Br}_e(T_w)$.

LEMMA 6.3. *If F is an instance of a $Count_p^M$ axiom schema and there is some $\pi \in \mathrm{Br}_0(T_F)$ then if $n' = |R| < |D|$, $ph \le N$, $N \le \sqrt{(n'-k)/k}/10$, and $(1.5N^2\sqrt{k/(n'-k)})^h \le 1/|M|$, there is restriction $\rho \in \mathcal{M}^N_{(D\times R)\restriction_\pi}$ such that there is a (p, M)-generic system over $(D \times R)\restriction_{\pi\rho}$ of height at most ph.*

PROOF. For each $v \in M$, let $\mathcal{B}_v$ be as above and define

$$G_v = \bigvee_{\sigma \in \mathcal{B}_v} \bigwedge_{\langle i,j \rangle \in \sigma} P_{ij}.$$

By Lemma 5.4, for a ρ chosen uniformly at random from $\mathcal{M}^N_{(D\times R)\restriction_\pi}$, the probability that $G_v\restriction_\rho$ fails to have a canonical matching decision tree of height at most h is less than $1/|M|$. Therefore the probability that a ρ fails to do this for all $v \in M$ is less than one. Choose some ρ that achieves this for all $v \in M$ and let T_v^* be the tree associated with $G_v\restriction_\rho$.

By Lemma 6.1, if τ is a branch of T_v^* then $\rho\tau$ is compatible with some $\sigma \in \mathcal{B}_v$, i.e. some $\sigma \in \mathrm{Br}_1(T_e)$ for some e with $v \in e$, and thus the leaf label of τ must be 1. Therefore, since T_v^* is a canonical decision tree for $G_v\restriction_\rho$, σ must be contained in $\rho\tau$. Since the elements of $\mathcal{B}_v$ are mutually incompatible, the choice of σ must be unique. Therefore, each leaf of T_v^* is associated with a unique $e \in M^{(p)}$ with $v \in e$. Relabel the leaves of the T_v^* by their associated $e \in M^{(p)}$. Lemma 6.1 implies that for any $v, v' \in M$, if τ and τ' are compatible branches in T_v^* and $T_{v'}^*$ then their leaf labels e, e' are compatible.

In order to create the trees T_v, for each branch σ of T_v^* with leaf label e, extend σ in T_v^* by appending trees $T_w^*\restriction_\sigma$ for each $w \in e$ in turn and labelling all leaves of the resulting branches by e. This at most multiplies the height of the trees by p. Observe that for $e = \{v_1, \ldots, v_p\}$ the branches with leaf label e in T_{v_j} are all elements of the form

$$\{\pi_1 \cdots \pi_p \mid \pi_i \in \mathrm{Br}_e(T_{v_i}^*) \text{ for } i = 1, \ldots, p\}$$

and are thus independent of the choice of j. □

The above construction of a generic system is essentially from [6]. Alternatively, one could create the trees T_v^* by constructing the canonical decision trees for the G_v without applying any restriction. Using an argument like the one showing that any Boolean formula that has CNF clause size c and DNF term size d has Boolean decision tree of height $\le cd$, one can show that the canonical decision tree constructed for G_v has height at most $k(k+1)$. This latter approach is very much like the one in [19].

The key property that we use about any (p, M)-generic system of height h over $D \times R$ is that it is a forest $\mathcal{T}$ of matching decision trees over $D \times R$ of height $\le h$ such that each branch appears 0 (mod p) times in $\mathcal{T}$ and such that the total number of trees is $\not\equiv 0 \pmod p$. In the terminology of [18] this is a *p-exceptional forest of (D, R)-labelled trees.*

7. Nullstellensatz Proofs

DEFINITION 7.1. Given multivariate polynomials

$$Q_1(\vec{x}), \ldots, Q_m(\vec{x}) \in R[x_1, \ldots, x_n]$$

there is no solution to

$$\begin{aligned} Q_1(\vec{x}) &= 0 \\ \cdots &= \cdot \\ Q_m(\vec{x}) &= 0 \end{aligned}$$

over $\{0,1\}$ if $\exists P_1(\vec{x}), \ldots, P_m(\vec{x}) \in R[x_1, \ldots, x_n]$ such that $\sum_{i=1}^m P_i(\vec{x}) \cdot Q_i(\vec{x}) \equiv r \not\equiv 0 \pmod p$ in $R[x_1, \ldots, x_n]/(x_1^2 - x_1, \cdots, x_n^2 - x_n)$. We say that $P_1, \ldots, P_m$ are a *Nullstellensatz r-refutation* of $\{Q_1, \ldots, Q_m\}$. (We drop the r when $r = 1$. If p is a prime then the exact value of r is irrelevant. Also, if p is prime then a Nullstellensatz refutation is guaranteed to exist by Hilbert's Nullstellensatz whenever there is no $\{0,1\}$ solution.) The *degree* of the r-refutation is the maximum degree of the P_i

DEFINITION 7.2. Let *onto-*$\mathcal{PHP}_R^D$ be the following system of polynomial equations in variables $x_{i,j}$ with $i \in D$, $j \in R$:

(1) $Q_i^D(\vec{x}) = (\sum_{j \in R} x_{i,j}) - 1 = 0$ one for each $i \in D$, and
(2) $Q_j^R(\vec{x}) = (\sum_{i \in D} x_{i,j}) - 1 = 0$ one for each $j \in R$, and
(3) $Q_{i,jk}(\vec{x}) = x_{i,j} \cdot x_{i,k} = 0$ one for each $i \in D$, $j,k \in R$, $j \neq k$, and
(4) $Q_{ij,k}(\vec{x}) = x_{i,k} \cdot x_{j,k} = 0$ one for each $i \neq j$, $i,j \in D$, $j \in R$.

Again we use *onto-*$\mathcal{PHP}_{|R|}^{|D|}$ to emphasize that the sizes of D and R are all that matter.

DEFINITION 7.3. We relate monomials and sets of edges in $D \times R$ as follows: Given a set of edges $\pi \in D \times R$, define $X_\pi = \prod_{\langle i,j \rangle \in \pi} x_{i,j}$ and given a monomial $X = x_{i_1,j_1}^{e_1} \cdots x_{i_k,j_k}^{e_k}$ with $e_1, \ldots, e_k \geq 1$, define $\pi_X = \{\langle i_1, j_1 \rangle, \ldots, \langle i_k, j_k \rangle\}$.

LEMMA 7.4. *If $|M| \equiv r \pmod p$ and a (p,M)-generic system of height h over $D \times R$ exists then there is a Nullstellensatz r-refutation of onto-$\mathcal{PHP}_R^D$ of degree at most $h-1$ over $\mathbb{Z}_p$.*

The basic idea is to consider the polynomial whose monomials are the products of the variables associated with each branch of the trees in the generic system. That is, with each tree T_v we get a polynomial

$$P_{T_v} = \sum_{\pi \in \mathsf{Br}(T_v)} X_\pi.$$

We first show that each P_{T_v} is $1 + L_v$ where L_v a linear combination of the Q polynomials of degree at most $h-1$.

LEMMA 7.5. *Let T be a matching decision tree over $D \times R$. Then $P_T = \sum_{\pi \in \mathsf{Br}(T)} \prod_{e \in \pi} x_e$ is of the form $1 + L$ where L is a linear combination of the onto-$\mathcal{PHP}_R^D$ polynomials with coefficient polynomials of degree $\leq h-1$.*

PROOF. Proof by induction on the number of internal vertices of T.

Base Case: If T has no internal vertices then it has one branch of height 0, $P_T(\vec{x}) = 1$ and all coefficient polynomials are 0 which gives degree -1.

Induction Step: Suppose that T has at least one internal vertex and has height h. Then it has one such vertex v all of whose children are leaves. Let T' be the matching decision tree obtained by removing all the children of v. Let π be the matching given along the path from the root to v.

If the query at v is $i \in D$, then

$$\begin{aligned} P_T(\vec{x}) &= P_{T'}(\vec{x}) + X_\pi - \sum_{j \in R \setminus \mathsf{range}(\pi)} X_\pi \cdot x_{i,j} \\ &= P_{T'}(\vec{x}) + X_\pi \cdot (1 - \sum_{j \in R \setminus \mathsf{range}(\pi)} x_{i,j}) \\ &= P_{T'}(\vec{x}) + X_\pi \cdot (1 - \sum_{j \in R} x_{i,j}) + X_\pi \cdot \sum_{k \in \mathsf{range}(\pi)} x_{i,k} \\ &= P_{T'}(\vec{x}) - X_\pi \cdot Q_i^D + X_\pi \cdot \sum_{k \in \mathsf{range}(\pi)} x_{i,k}. \end{aligned}$$

X_π has degree at most $h-1$, the last term is a degree $h-2$ combination of the $Q_{ij,k}$, and applying the induction hypothesis to $P_{T'}$ yields the desired result.

The case when the query is $j \in R$ is analogous. □

PROOF OF LEMMA 7.4. Consider $\sum_{v \in M} P_{T_v}$ in $\mathbb{Z}_p$. On the one hand it is

$$\sum_{v \in M} (L_v + 1) = r + \sum_{v \in M} L_v.$$

On the other hand, every branch in the generic system appears some multiple of p times. Therefore over $\mathbb{Z}_p$,

$$\sum_{v \in M} P_{T_v} = 0.$$

We derive $r + \sum_{v \in M} L_v = 0$ and obtain the Nullstellensatz refutation by reversing signs. □

8. A Nullstellensatz degree lower bound for *onto-*$\mathcal{PHP}_N^{N+p^\ell}$

In this section we prove the following theorem which is of independent interest.

THEOREM 8.1. *Let $r \not\equiv 0 \pmod p$. If $N \geq ((p+2)^\ell - p^\ell)/2$ then any Nullstellensatz r-refutation of onto-$\mathcal{PHP}_N^{N+p^\ell}$ over $\mathbb{Z}_p$ must have degree at least $2^\ell - 1$.*

DEFINITION 8.2. A *d-design for $D \times R$* is a mapping $\mathcal{D}$ from the partial matchings of size $\leq d$ on $D \times R$ into $\mathbb{Z}_p$ such that

(a) $\mathcal{D}(\emptyset) = 1$ for the empty matching $\emptyset$,
(b) For each partial matching π with $|\pi| < d$ and $i \in D \setminus \mathrm{dom}(\pi)$

$$\sum_{j \in R \setminus \mathsf{range}(\pi)} \mathcal{D}(\pi \cup \langle i,j \rangle) \equiv \mathcal{D}(\pi) \pmod p$$

(c) For each partial matching π with $|\pi| < d$ and $j \in R \setminus \mathrm{range}(\pi)$

$$\sum_{i \in D \setminus \mathsf{dom}(\pi)} \mathcal{D}(\pi \cup \langle i,j \rangle) \equiv \mathcal{D}(\pi) \pmod p$$

LEMMA 8.3. *Let $r \not\equiv 0 \pmod p$. If there is a d-design for $D \times R$ then any r-refutation of onto-$\mathcal{PHP}_R^D$ over $\mathbb{Z}_p$ requires degree at least d.*

PROOF. We extend the d-design $\mathcal{D}$ to be a function from the set of polynomials to $\mathbb{Z}_p$. For any monomial X in variables $x_{i,j}$ with $i \in D$ and $j \in R$ define

$$\mathcal{D}(X) = \begin{cases} \mathcal{D}(\pi_X) & \text{if } \pi_X \text{ is a matching with } |\pi_X| \leq d \\ 0 & \text{otherwise} \end{cases}$$

and extend $\mathcal{D}$ linearly over $\mathbb{Z}_p$ to a map $\mathcal{D} : \mathbb{Z}_p[\vec{x}] \to \mathbb{Z}_p$ by setting $\mathcal{D}(P_1 + P_2) = \mathcal{D}(P_1) + \mathcal{D}(P_2)$ for $P_1, P_2 \in \mathbb{Z}_p[\vec{x}]$ and $\mathcal{D}(aP) = a\mathcal{D}(P)$ for $a \in \mathbb{Z}_p$ and $P \in \mathbb{Z}_p[\vec{x}]$.

Clearly $\mathcal{D}(1) = \mathcal{D}(\emptyset) = 1$ by part (a) of the design definition. We consider the polynomials in the definition of *onto-*$\mathcal{PHP}_R^D$ and show that for any $P \in \mathbb{Z}_p[\vec{x}]$ of degree $< d$,

$$\mathcal{D}(P \cdot Q_i^D) = \mathcal{D}(P \cdot Q_j^R) = \mathcal{D}(P \cdot Q_{i,jk}) = \mathcal{D}(P \cdot Q_{ij,k}) = \mathcal{D}(P \cdot (x_{i,j}^2 - x_{i,j})) = 0. \tag{$*$}$$

We see that $(*)$ is sufficient by observing that it implies if $0 \neq r = \Sigma_i P_i Q_i$ is an r-refutation of *onto-*$\mathcal{PHP}_R^D$ over $\mathbb{Z}_p$ of degree $< d$ then $0 \neq r = \mathcal{D}(r) = \mathcal{D}(\Sigma_i P_i Q_i) = \Sigma_i \mathcal{D}(P_i Q_i) = 0$ which is a contradiction.

To prove $(*)$, by the linearity of $\mathcal{D}$ it clearly suffices to prove it when P is simply a monomial X of degree $< d$. Furthermore, if μ_X is not a partial matching then $\mathcal{D}(X) = 0$ so, by the linearity of $\mathcal{D}$, we can assume that μ_X is a partial matching.

Since $Q_{i,jk}$ and $Q_{ij,k}$ are monomials and both $\mu_{Q_{i,jk}}$ and $\mu_{Q_{ij,k}}$ are not partial matchings we immediately have $\mathcal{D}(X \cdot Q_{i,jk}) = \mathcal{D}(X \cdot Q_{ij,k}) = 0$ for any monomial X.

Also, since $\mu_{X \cdot x_{i,j}^2} = \mu_{X \cdot x_{i,j}}$, the linearity of $\mathcal{D}$ implies that $\mathcal{D}(X \cdot (x_{i,j}^2 - x_{i,j})) = 0$.

For $Q_i^D(\vec{x}) = \sum_{j \in R} x_{i,j} - 1 = 0$ we have two cases depending on whether or not $i \in \mathrm{dom}(\pi_X)$. If $i \notin \mathrm{dom}(\pi_X)$ then

$$\begin{aligned} \mathcal{D}(X \cdot Q_i^D(\vec{x})) &= \mathcal{D}(X \cdot (\sum_{j \in R} x_{i,j} - 1)) \\ &= \sum_{j \in R} \mathcal{D}(X \cdot x_{i,j}) - \mathcal{D}(X) \\ &= \sum_{j \in R \setminus \mathrm{range}(\pi_X)} \mathcal{D}(X \cdot x_{i,j}) - \mathcal{D}(X) \\ &= \sum_{j \in R \setminus \mathrm{range}(\pi_X)} \mathcal{D}(\mu_X \cup \langle i,j \rangle) - \mathcal{D}(\mu_X) \\ &= 0 \end{aligned}$$

over $\mathbb{Z}_p$ by part (b) of the definition of a d-design since $|\pi_X| < d$. If $i \in \mathrm{dom}(\pi_X)$ then let $\langle i, j^* \rangle \in \pi_X$. In this case

$$\begin{aligned} \mathcal{D}(X \cdot Q_i^D(\vec{x})) &= \mathcal{D}(X \cdot (\sum_{j \in R} x_{i,j} - 1)) \\ &= \sum_{j \in R} \mathcal{D}(X \cdot x_{i,j}) - \mathcal{D}(X) \\ &= \mathcal{D}(X \cdot x_{i,j^*}) - \mathcal{D}(X) \\ &= \mathcal{D}(\mu_X \cup \langle i, j^* \rangle) - \mathcal{D}(\mu_X) \\ &= 0 \end{aligned}$$

since $\mu_{X \cdot x_{i,j}}$ is not a matching for $j \neq j^*$ and $\mu_X \cup \langle i, j^* \rangle = \mu_X$.

The result for $Q_j^R(\vec{x})$ follows similarly using part (c) of the definition of a d-design. □

LEMMA 8.4. *If there is a d-design $\mathcal{D}$ for $D \times R$ over $\mathbb{Z}_p$ then there is a $2d+1$-design $\mathcal{D}'$ for $D' \times R'$ over $\mathbb{Z}_p$ where $|D'| = (p+1)|D| + |R|$ and $|R'| = |D| + (p+1)|R|$. (Observe that $|D'| - |R'| = p(|D| - |R|)$.)*

Before we prove Lemma 8.4, we show how it implies Theorem 8.1.

PROOF OF THEOREM 8.1. For $\ell \geq 0$ let $N_\ell = ((p+2)^\ell - p^\ell)/2$. We show by induction that there is a $2^\ell - 1$-design for $[1, N_\ell + p^\ell] \times [1, N_\ell]$ over $\mathbb{Z}_p$. The theorem then will follow by Lemma 8.3.

For $\ell = 0$, letting $\mathcal{D}(\emptyset) = 1$ is sufficient to satisfy the conditions for a 0-design.

Suppose we have a $2^\ell - 1$-design $\mathcal{D}$ for $[1, N_\ell + p^\ell] \times [1, N_\ell]$. Observe that $N_{\ell+1} + p^{\ell+1} = (p+1)(N_\ell + p^\ell) + N_\ell$ and $N_{\ell+1} = (N_\ell + p^\ell) + (p+1)N^\ell$. Applying Lemma 8.4 we get a $2(2^\ell - 1) + 1 = 2^{\ell+1} - 1$-design $\mathcal{D}'$ for $[1, N_{\ell+1} + p^{\ell+1}] \times [1, N_{\ell+1}]$ over $\mathbb{Z}_p$ as required. □

PROOF OF LEMMA 8.4. Let $\mathcal{D}$ be a d-design for $D \times R$ over $\mathbb{Z}_p$. Let $D' = \{i_1, \ldots, i_{|D'|}\}$ and $R' = \{j_1, \ldots, j_{|R'|}\}$. Divide D' into $|R|$ individual points $i_1, \ldots, i_{|R|}$ and $|D|$ blocks $D_1, \ldots D_{|D|}$ each of size $p+1$ and divide R' into $|D|$ individual points $j_1, \ldots, j_{|D|}$ and $|R|$ blocks, $R_1, \ldots, R_{|R|}$ each of size $p+1$. Following [10], we also fix a cyclic ordering on the elements within each block, e.g. as a permutation $\sigma : D' \cup R' \to D' \cup R'$ which maps each of the individual points to itself and whose other orbits are the blocks of size $p+1$. We say that $\langle i, j \rangle$ is *parallel* to $\langle i', j' \rangle$ iff there is some r such that $i' = \sigma^r(i)$ and $j' = \sigma^r(j)$. Observe that this forms an equivalence relation on edges.

In matchings on $D' \times R'$, we say that an edge is a *cross edge* if it is in $D_a \times R_b$ for some a and b and is a *rung* if it is in $b \times R_b$ or $D_a \times a$ for some a or b, i.e. it joins some individual point to its corresponding block. Given $\pi \subseteq D' \times R'$, let $Im(\pi) = \{\langle a, b \rangle \mid \pi \cap D_a \times R_b \neq 0\}$, i.e. $Im(\pi)$ is the projection of the cross edges in π onto $D \times R$.

DEFINITION 8.5. For each choice, V, of a set of $|D| + |R|$ representative elements, $u_i \in D_i$ for $i = 1, \ldots, |D|$ and $v_j \in R_j$ for $j = 1, \ldots, |R|$ and matching π on $D' \times R'$, we say that π *respects* V if

(A) the only edges of π are rungs or cross edges,
(B) each rung in π matches a representative element given by V, i.e. is of the form $\langle u_i, j_i \rangle$ for $i \leq |D|$ or $\langle i_j, v_j \rangle$ for $j \leq |R|$,
(C) for any a and b, each cross edge of π in $D_a \times R_b$ is parallel to $\langle u_a, v_b \rangle$ but not equal to it.

For each V as above, we can define a map $\mathcal{D}^V$ from the set of partial matchings of size $\leq d$ on $D' \times R'$ to $\mathbb{Z}_p$.

$$\mathcal{D}^V(\pi) = \begin{cases} \mathcal{D}(Im(\pi)) & \text{if } \pi \text{ respects } V \text{ and } Im(\pi) \text{ is a matching of size} \leq d \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we define $\mathcal{D}'(\pi) = \sum_V \mathcal{D}^V(\pi)$.

Claim: $\mathcal{D}'$ is a $2d+1$-design for $D' \times R'$ over $\mathbb{Z}_p$.

Clearly $\mathcal{D}'(\emptyset) = \sum_V \mathcal{D}^V(\emptyset) = (p+1)^{|D|+|R|} = 1$ over $\mathbb{Z}_p$ since there are exactly $(p+1)^{|D|+|R|}$ different choices of V and for each of these $\mathcal{D}^V(\emptyset) = 1$. Thus condition (a) for a design is satisfied.

We now show that condition (b) for a $2d+1$-design is satisfied. The proof for condition (c) is analogous. Let $|\pi| \leq 2d$ be a matching on $D' \times R'$ and $i' \in D' \setminus \mathrm{dom}(\pi)$.

We can assume without loss of generality that $Im(\pi)$ is a matching of size $\leq d$ since otherwise $\mathcal{D}'(\pi) = 0$ and $\mathcal{D}'(\pi \cup \langle i', j' \rangle) = 0$ for all $j' \in R'$.

Now

$$\begin{aligned}\sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}'(\pi \cup \langle i', j' \rangle) &= \sum_{j' \in R' \setminus \mathrm{range}(\pi)} \sum_V \mathcal{D}^V(\pi \cup \langle i', j' \rangle) \\ &= \sum_V \sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}^V(\pi \cup \langle i', j' \rangle).\end{aligned}$$

We have several cases:

Case 1: if $i' = i_j$ for $1 \leq j \leq |R|$ then $Im(\pi \cup \langle i', j' \rangle) = Im(\pi)$ and, if $\pi \cup \langle i', j' \rangle$ respects V, it must be the case that $j' = v_j$. Thus

$$\begin{aligned}\sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}'(\pi \cup \langle i', j' \rangle) &= \sum_V \sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}^V(\pi \cup \langle i', j' \rangle) \\ &= \sum_V \mathcal{D}^V(\pi \cup \langle i_j, v_j \rangle) \\ &= \sum_V \mathcal{D}^V(\pi) \\ &= \mathcal{D}'(\pi)\end{aligned}$$

as required.

Case 2: $i' \in D_a$ for some a. We split this case into several subcases based on the structure of π. Let $\mathcal{V}_{i'}$ be the set of those V such that $u_a \neq i'$. For each subcase we first observe that we only need to consider those $V \in \mathcal{V}_{i'}$ such that π respects V. If π does not respect V then $\pi \cup \langle i', j' \rangle$ does not respect V, so $\mathcal{D}^V(\pi) = \mathcal{D}^V(\pi \cup \langle i', j' \rangle) = 0$. If V has $u_a = i'$ then $j' = a$ is the only value such that V respects $\pi \cup \langle i', j' \rangle$ and for this value, $Im(\pi \cup \langle i', j' \rangle) = Im(\pi)$. Thus for each $V \notin \mathcal{V}_{i'}$,

$$\sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}^V(\pi \cup \langle i', j' \rangle) = \mathcal{D}^V(\pi).$$

Subcase (a): π has a cross edge $\langle i^*, j^* \rangle$ with $i^* \in D_a$ and $j^* \in R_b$ for some b. If π respects $V \in \mathcal{V}_{i'}$ then there is exactly one j' such that $Im(\pi \cup \langle i' j' \rangle)$ is a matching and $\pi \cup \langle i', j' \rangle$ respects V. This is the unique $j' \in R_b$ such that such that $\langle i', j' \rangle$ is parallel to $\langle i^*, j^* \rangle$, i.e. $i' = \sigma^r(i^*)$ and $j' = \sigma^r(j^*)$ for some r. For this value of j', $Im(\pi \cup \langle i', j' \rangle) = Im(\pi)$, so for each $V \in \mathcal{V}_{i'}$,

$$\sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}^V(\pi \cup \langle i', j' \rangle) = \mathcal{D}^V(\pi).$$

Subcase (b): π has no cross edges touching D_a and $|Im(\pi)| < d$.
In this case, given $V \in \mathcal{V}_{i'}$ such that π respects V, for each $b \in R \setminus \mathrm{range}(Im(\pi))$ there is one choice of $j' \in D' \setminus \mathrm{range}(\pi)$ such that $Im(\pi \cup \langle i', j' \rangle)$ is a matching

and $\pi \cup \langle i', j' \rangle$ respects V. This j' is the unique member of R_b such that $\langle i', j' \rangle$ is parallel to $\langle u_a, v_b \rangle$ and for this value, $Im(\pi \cup \langle i', j' \rangle) = Im(\pi) \cup \langle a, b \rangle$. Thus

$$\begin{aligned}\sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}^V(\pi \cup \langle i', j' \rangle) &= \sum_{b \in R \setminus \mathrm{range}(Im(\pi))} \mathcal{D}(Im(\pi) \cup \langle a, b \rangle) \\ &= \mathcal{D}(Im(\pi)) \\ &= \mathcal{D}^V(\pi)\end{aligned}$$

using the fact that $\mathcal{D}$ satisfies condition (b) for a d-design over $D \times R$.

Subcase (c): π has no cross edges or rung edges touching D_a and $|Im(\pi)| = d$. In this case, $|Im(\pi \cup \langle i', j' \rangle)| > d$ for any j' so $\mathcal{D}^V(Im(\pi \cup \langle i', j' \rangle)) = 0$ for all V. We show that the sum of $\mathcal{D}^V(\pi)$ for all $V \in \mathcal{V}_{i'}$ is also 0. We can group such V that π respects into equivalence classes based on their choices other than u_a. Observe that the choice of $u_a \neq i'$ does not affect the value of $\mathcal{D}^V(\pi)$ since π has no cross edges touching D_a. Within each equivalence class there are exactly p choices of $u_a \neq i'$, so for each such class C, $\sum_{V \in C} \mathcal{D}^V(\pi)$ is a multiple of p and thus equal to 0 in $\mathbb{Z}_p$. Therefore, the sum of $\mathcal{D}^V(\pi)$ for all $V \in \mathcal{V}_{i'}$ is 0, and thus $\sum_{V \in \mathcal{V}_{i'}} \sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}^V(\pi \cup \langle i', j' \rangle) = \sum_{V \in \mathcal{V}_{i'}} \mathcal{D}^V(\pi)$.

Subcase (d): π has a rung edge but no cross edges touching D_a and $|Im(\pi)| = d$. As in the previous case, $|Im(\pi \cup \langle i', j' \rangle)| > d$ for any j' such that $\pi \cup \langle i', j' \rangle$ is a matching so $\mathcal{D}^V(Im(\pi \cup \langle i', j' \rangle)) = 0$ for all V. Again we show that the sum of $\mathcal{D}^V(\pi)$ for all $V \in \mathcal{V}_{i'}$ is also 0. In this case, π has at least d cross edges and at most $2d$ total edges, one of which is a rung edge that does not touch the same block as any cross edge. Thus there is some cross edge of π, $\langle i^*, j^* \rangle \in D_e \times R_f$ for some e and f, such that no other edges of π touch D_e or R_f. We group all $V \in \mathcal{V}_{i'}$ that π respects into equivalence classes based on their choices of points other than u_e and v_f. Since π has no other edges touching D_e or R_f, the value of $\mathcal{D}^V(\pi)$ is the same for all V in each equivalence class. Within each equivalence class there are exactly p choices of V since π respects each V and there are exactly p choices of u_e and v_f such that $\langle u_e, v_f \rangle$ is parallel to $\langle i^*, j^* \rangle$ but $u_e \neq i^*$. Therefore for each class C', $\sum_{V \in C'} \mathcal{D}^V(\pi) = 0$ over $\mathbb{Z}_p$. It follows that the sum of $\mathcal{D}^V(\pi)$ for all such V is 0, and thus $\sum_{V \in \mathcal{V}_{i'}} \sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}^V(\pi \cup \langle i', j' \rangle) = \sum_{V \in \mathcal{V}_{i'}} \mathcal{D}^V(\pi)$.

Summarizing Case 2, we have $\sum_V \sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}^V(\pi \cup \langle i', j' \rangle) = \sum_V \mathcal{D}^V(\pi)$, i.e.

$$\sum_{j' \in R' \setminus \mathrm{range}(\pi)} \mathcal{D}'(\pi \cup \langle i', j' \rangle) = \mathcal{D}'(\pi)$$

as required. □

When p is prime, using a construction from [18], we can see that the degree of a Nullstellensatz refutation is not too much larger than the above lower bound. To see this we first need the following:

PROPOSITION 8.6. *If p is prime and $m - p^\ell < p^\ell a \leq m$ then $\binom{m}{p^\ell} \equiv a \pmod{p}$.*

PROOF. For each r, $1 \leq r \leq p^\ell$, let $m(r)$ be the unique integer between $m - p^\ell + 1$ and m that is congruent to r modulo p^ℓ. Observe that

$$\binom{m}{p^\ell} \equiv \prod_{r=1}^{p^\ell} \frac{m(r)}{r} \pmod{p}.$$

Since $m(p^\ell) = p^\ell a$, by assumption, $\frac{m(p^\ell)}{p^\ell} \equiv a \pmod p$.

For $1 \le r < p^\ell$, write $r = p^k r'$ where $\gcd(p, r') = 1$ and $k < \ell$. Since $m(r) \equiv r \pmod{p^\ell}$, there is some m' such that $m(r) = p^\ell m' + r = p^k(p^{k-\ell} + r')$. Therefore $\frac{m(r}{r} = \frac{p^{k-\ell}+r'}{r'}$. Since $p^{k-\ell} + r' \equiv r' \pmod p$, we derive that $\frac{m(r)}{r} \equiv 1 \pmod p$ from which the proposition follows. □

LEMMA 8.7. *If p is prime and $p^\ell \le N$, there is a Nullstellensatz refutation of onto-$\mathcal{PHP}_N^{N+p^\ell}$ of degree $p^\ell - 1$.*

PROOF. Let $D = [1, N + p^\ell]$ and $R = [1, N]$. Consider the polynomial

$$\sum_{A \subset D,\ |A|=p^\ell\ \pi,\ \mathrm{dom}(\pi)=A} \ \sum X_\pi - \sum_{B \subset R,\ |B|=p^\ell\ \pi,\ \mathrm{range}(\pi)=B} \ \sum X_\pi.$$

Since each X_π with $|\pi| = p^\ell$ appears exactly once in each sum, the value of the polynomial is 0. On the other hand, notice that $\sum_{\pi,\ \mathrm{dom}(\pi)=A} X_\pi$ is the polynomial P_T for a matching decision tree of height h that queries each element of A along each path. Therefore by Lemma 7.5, $\sum_{\pi,\ \mathrm{dom}(\pi)=A} X_\pi = 1 + L_A$ where L_A is a linear combination of the onto-$\mathcal{PHP}_R^D$ polynomials of degree $\le p^\ell - 1$ over $\mathbb{Z}_p$. Similarly, $\sum_{\pi,\ \mathrm{range}(\pi)=B} X_\pi = 1 + L_B$ where L_B is a combination of degree $\le p^\ell - 1$.

Therefore

$$\begin{aligned} 0 &= \sum_{A \subset D,\ |A|=p^\ell\ \pi,\ \mathrm{dom}(\pi)=A} \ \sum X_\pi - \sum_{B \subset R,\ |B|=p^\ell\ \pi,\ \mathrm{range}(\pi)=B} \ \sum X_\pi \\ &= \binom{N+p^\ell}{p^\ell} - \binom{N}{p^\ell} + L \end{aligned}$$

where L is a combination of the onto-$\mathcal{PHP}_R^D$ polynomials of degree at most $p^\ell - 1$. Since $\binom{N+p^\ell}{p^\ell} - \binom{N}{p^\ell} \equiv 1 \pmod p$ by Proposition 8.6, we obtain a Nullstellensatz refutation of onto-$\mathcal{PHP}_R^D$ of degree at most $p^\ell - 1$. □

9. Putting it all Together

THEOREM 9.1. *For $\ell \le \epsilon \log_2 n$ with $1/\epsilon = 3 \cdot 4^{d+1}(\frac{1}{2} + \log_2(p+2))$, any depth d proof of onto-$PHP_n^{n+p^\ell}$ in a Frege system augmented by $Count_p$ axiom schemas requires size at least $n^{2^\ell/(4^{d+1}p)}$.*

PROOF. Suppose that ℓ satisfies the conditions of the statement and that $\mathcal{P}$ is a depth d Frege proof with $Count_p$ axioms of onto-$PHP_n^{n+p^\ell}$ of size $S < n^{2^\ell/(4^{d+1}p)}$.

Let $k = \log_2 S$, and $N = n^{1/(2\cdot 4^{d+1})}/\sqrt{k}$.

Since $\ell \le \epsilon \log_2 n$,

$$\begin{aligned} 10(p+2)^\ell\sqrt{k} &\le 10(p+2)^\ell 2^{\ell/2}\sqrt{\log_2 n} \\ &\le 10n^{\epsilon(\frac{1}{2}+\log_2(p+2))}\sqrt{\log n} \\ &\le 10n^{1/(3\cdot 4^{d+1})}\sqrt{\log_2 n} \\ &< n^{1/(2\cdot 4^{d+1})}. \end{aligned}$$

for n sufficiently large relative to d. Therefore $10(p+2)^\ell < N$.

Define $n_0, \ldots n_d$ as in the statement of Lemma 5.3. Then $n_d = n^{1/4^d}/(9k)^{\delta_d}$ where $\delta_d = \sum_{i=1}^d 4^{-i} < 1/3$ and thus $n_d \ge n^{1/4^d}/(9k)^{1/3} > N$. It follows that $n_d \ge k = \log_2 S$ and $n_d \ge 10p^\ell$.

Therefore by Lemma 5.3, there is a restriction $\rho \in \mathcal{M}_{D\times R}^{n_d}$ and a k-evaluation $\mathbb{T}$ of the set of subformulas of $\mathcal{P}\restriction_\rho$ over $(D \times R)\restriction_\rho = D' \times R'$ where $|D'| = |R'| + p^\ell = n_d + p^\ell$.

By Lemmas 4.2 and 4.3, there must be some instance F of a $Count_p^M$ axiom schema in $\mathcal{P}\restriction_\rho$ and $\pi \in \mathrm{Br}_0(T_F)$. We now let $h = 4^{d+1}\log_n S$. Observe that by assumption about S, $h < 2^\ell/p$ and that

$$\begin{aligned} (1.5N^2\sqrt{k/(n_d-k)})^h &< (3N^2\sqrt{k}(9k)^{1/6}/n^{1/(2\cdot 4^d)})^h \\ &< (n^{1/4^{d+1}-1/(2\cdot 4^d)})^h \\ &= n^{h/4^{d+1}} \le 1/S \le 1/|M| \end{aligned}$$

and apply Lemma 6.3 to obtain a (p, M)-generic system of height $ph < 2^\ell$ over $D'' \times R''$ where $|D''| = |R''| + p^\ell = N + p^\ell$. Applying Lemma 7.4, we obtain a Nullstellensatz $|M|$-refutation of onto-$\mathcal{PHP}_N^{N+p^\ell}$ of degree less than $2^\ell - 1$ which contradicts Theorem 8.1. □

Riis [18], by considering all possible domain and range subsets of size p^ℓ, as in Lemma 8.7, has shown that one can prove onto-$\mathcal{PHP}_n^{n+p^\ell}$ from $Count_p$ using a constant-depth proof of size $n^{O(p^\ell)}$ so the above bound is relatively tight.

COROLLARY 9.2. *Any depth d Frege proof of PHP_n^{n+1} requires size $n^{\Omega(n^{1/(30\cdot 4^d)})}$ even if axiom schemas for onto-PHP_n^{n+1} are permitted.*

PROOF. Apply Theorem 9.1 with $p = 2$, $\ell = (\log_2 n)/(30 \cdot 4^d) - 1$, and $n' = n + 1 - p^\ell$. (It is not hard to check that the conditions hold.) This implies that any depth d Frege proof of onto-$PHP_{n'}^{n'+p^\ell}$ using axiom schemas for $Count_2$ requires size $n^{\Omega(n^{1/(30\cdot 4^d)})}$. Now it is easy to see that onto-PHP_n^{n+1} is an immediate consequence of $Count_2^{2n+1}$ so the same lower bound applies to the size of the proofs with onto-PHP_n^{n+1} schemas instead of $Count_2$ axiom schemas. Finally, observe that onto-$PHP_{n'}^{n'+p^\ell}$ is an immediate consequence of $PHP_{n'+p^\ell-1}^{n'+p^\ell}$, i.e. of PHP_n^{n+1}. □

COROLLARY 9.3. [10] *If p and q are positive integers such that q contains a prime factor not dividing p then any depth d Frege proof of $Count_q$ requires size $2^{n^{\Omega(1/4^d)}}$ even if axiom schemas for $Count_p$ are permitted.*

More generally:

COROLLARY 9.4. *If p and $q_1, \ldots, q_k$ are positive integers such that each q_i contains a prime factor not dividing p then any depth d Frege proof of $\bigvee_{i=1}^k Count_{q_i}$ requires size $2^{n^{\Omega(1/4^d)}}$ even if axiom schemas for $Count_p$ are permitted.*

PROOF. If q_i contains a prime factor not dividing p then there is an easy proof of onto-$PHP_n^{n+p^\ell}$ from $Count_{q_i}$ by counting the number of edges touching the domain and range, respectively, and observing that these must be different modulo q_i. The implementation of this as a proof of size $(2n+p^\ell)^{O(q_i)}$ is quite straightforward. The $\Omega(\cdot)$ in the lower bound depends on the sizes of p and the q_i but does not depend on n or d. The overall argument is easily handled by cases. □

Following standard connections between bounded-depth Frege systems and bounded arithmetic (see [15]) the results above also have implications for the relativized system of bounded arithmetic $S_2(R)$, defined by Buss [11], in which R is an uninterpreted function symbol. In general, lower bounds for $S_2(R)$ follow from $2^{(\log n)^{\omega(1)}}$ size lower bounds. If we let $PHP_*^{*+1}(R)$ (respectively $\textit{onto-}PHP_*^{*+p^\ell}(R)$, $Count_p(R)$, etc.) denote the first-order version of the pigeonhole principle (etc.) for the relation R then the following are immediate corollaries of the above results.

COROLLARY 9.5.

(1) *Let $\ell(n)$ be an integer function of n such that $\ell(n) = \omega(\log\log n)$ and $\ell(n) = o(\log n)$. There is no proof of $PHP_*^{*+p^{\ell(\cdot)}}(R)$ in $S_2(R) + Count_p(R)$.*
(2) *There is no proof of $PHP_*^{*+1}(R)$ in $S_2(R) + \textit{onto-}PHP_*^{*+1}(R)$.*
(3) [10] *If q contains a prime factor not dividing p then there is no proof of $Count_q(R)$ in $S_2(R) + Count_p(R)$.*

10. Remarks

It is interesting to compare the degree lower bound for the Nullstellensatz refutations of $\textit{onto-}\mathcal{PHP}_N^{N+p^\ell}$ with the degree lower bound for $\mathcal{PHP}_N^{N+s}$ using the quite different construction in [5]. If we take $p = 2$ and $N = (4^\ell - 2^\ell)/2$, then the degree lower bound from Theorem 8.1 is $d = 2^\ell - 1$ which satisfies $N = d(d+1)/2$, i.e. the same degree as in [5] despite the more stringent conditions required in Theorem 8.1. (For $p > 2$, Theorem 8.1 does not give as large a degree bound.)

Recently, Razborov [17] has shown an $\Omega(N)$ degree lower bound not only for Nullstellensatz refutations of $\mathcal{PHP}_N^{N+s}$ but also for more general polynomial refutations called Polynomial Calculus or Gröbner proofs [12, 10]. It is an open problem to prove non-trivial lower degree bounds for polynomial calculus proofs of $\textit{onto-}\mathcal{PHP}_N^{N+p^\ell}$; the sorts of characterizations of polynomials based on $\mathcal{PHP}_N^{N+s}$ that are critical for proving the lower bounds in [17] do not seem to extend easily to this problem.

Finally, it would be interesting to improve our lower bound and close the gap between $p^\ell - 1$ and $2^\ell - 1$ or to reduce the size of N required to achieve it.

Acknowledgements

The authors would like to thank Toni Pitassi and Russell Impagliazzo for discussions that clarified many of the issues in this paper, and Alasdair Urquhart for his insight as to how best to formulate these arguments.

References

[1] M. Ajtai. The independence of the modulo p counting principles. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, pages 402–411, Montréal, Québec, Canada, May 1994.
[2] Miklós Ajtai. The complexity of the pigeonhole principle. In *29th Annual Symposium on Foundations of Computer Science*, pages 346–355, White Plains, NY, October 1988. IEEE.
[3] Miklós Ajtai. Parity and the pigeonhole principle. In Samuel R. Buss and P. J. Scott, editors, *Feasible Mathematics*, pages 1–24, A Mathematical Sciences Institute Workshop, Ithaca, NY, 1990. Birkhäuser.
[4] Paul W. Beame. A switching lemma primer. Technical Report 95-07-01, Department of Computer Science and Engineering, University of Washington, November 1994.

[5] Paul W. Beame, Stephen A. Cook, Jeff Edmonds, and Russell Impagliazzo. The relative complexity of NP search problems. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 303–314, Las Vegas, NV, May 1995.
[6] Paul W. Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 794–806, Santa Fe, NM, November 1994. IEEE.
[7] Paul W. Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, Pavel Pudlák, and Alan Woods. Exponential lower bounds for the pigeonhole principle. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, pages 200–220, Victoria, B.C., Canada, May 1992.
[8] Paul W. Beame and Toniann Pitassi. An exponential separation between the matching principle and the pigeonhole principle. In *8th Annual IEEE Symposium on Logic in Computer Science*, pages 308–319, Montreal, Quebec, June 1993.
[9] S. Bellantoni, T. Pitassi, and A. Urquhart. Approximation and small depth Frege proofs. In *Proceedings, Structure in Complexity Theory, Sixth Annual Conference*, pages 367–391, Chicago, IL, June 1991. IEEE.
[10] S. Buss, R. Impagliazzo, J. Krajíček, P. Pudlák, A. A. Razborov, and J. Sgall. Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Computation Complexity*. Submitted.
[11] Samuel R. Buss. *Bounded Arithmetic*. Bibliopolis, Napoli, 1986. Volume 3 of Studies in Proof Theory.
[12] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Gröbner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty Eighth Annual ACM Symposium on Theory of Computing*, pages 174–183, Philadelphia, PA, May 1996.
[13] Johan Håstad. *Computational Limitations of Small-Depth Circuits*. MIT Press, 1987. ACM Doctoral Dissertation Award Series (1986).
[14] J. Krajíček, P. Pudlák, and A Woods. Exponential lower bounds to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures and Algorithms*, 7(1), 1995.
[15] J. Paris and A. Wilkie. Counting problems in bounded arithmetic. In *Methods in Mathematical Logic: Proceedings of the 6th Latin American Symposium on Mathematical Logic 1983*, volume 1130 of *Lecture notes in Mathematics*, pages 317–340, Berlin, 1985. Springer-Verlag.
[16] Toniann Pitassi, Paul W. Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3(2):97–140, 1993.
[17] A. A. Razborov. Lower bounds for the polynomial calculus. Manuscript, 1996.
[18] Søren Riis. *Independence in Bounded Arithmetic*. PhD thesis, Oxford University, 1993.
[19] Søren Riis. *Count(q)* does not imply *Count(p)*: revised version. Technical Report RS 94-21, BRICS, 1994.
[20] Søren Riis. *Count(q)* versus the pigeonhole principle. *Arch. Math Logic*, 37:157–188, 1997.

PAUL BEAME, COMPUTER SCIENCE AND ENGINEERING, UNIVERSITY OF WASHINGTON, BOX 352350, SEATTLE, WA 98195-2350, USA
E-mail address: beame@cs.washington.edu

SØREN RIIS, DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF AARHUS, NY MUNKEGADE, BUILDING 540, DK-300 AARHUS C, DENMARK
E-mail address: smriis@daimi.aau.dk

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

Ranking Arithmetic Proofs by Implicit Ramification

Stephen J. Bellantoni

ABSTRACT. Proofs in an arithmetic system are ranked according to a ramification hierarchy based on occurrences of induction. It is shown that this ranking of proofs corresponds exactly to a natural ranking of the primitive recursive functions based on occurrences of recursion. A function is provably convergent using a rank r proof, if and only if it is a rank r function. The result is of interest to complexity theorists, since rank one corresponds to polynomial time. Remarkably, this characterization of polynomial-time provability admits induction over formulas having arbitrary quantifier complexity.

1. Introduction

The primitive recursive functions can be assigned ranks, based on an examination of the structure of their derivations as built up from the initial functions by the rules of composition and recursion. A hierarchy defined in this way consists of the linear-space computable functions at level 1; if primitive recursion is replaced with recursion on notation, then the defined hierarchy has the polynomial-time computable functions at level 1. At higher levels, both hierarchies consist of certain of the Grzegorczyk classes [4]. The current work shows how to rank arithmetic proofs containing inductions, such that the ranking of proofs corresponds exactly to this ranking of recursions. It is shown that a function can be proven convergent by a rank r proof if and only if it is a rank r function.

The ranking of proofs is entirely syntactic, without any obvious reference to computation time, the size of quantified values, etc. The ranking of proofs is not a disguise for the ranking of function symbols appearing in the proofs. Remarkably, induction formulas are allowed to have arbitrary alternations of unbounded quantifiers. Yet the first level of the logical hierarchy proves convergence of exactly the polynomial time computable functions. Thus, the fragment of arithmetic consisting of rank 1 proofs, passes a key test for being considered a "feasible" system.

Compared to the bounded induction rules of earlier weak subsystems of arithmetic, the present definitions permit a radical reduction in the syntactic restrictions placed on the induction formula. The only remaining restriction is a relatively mild

Much of this work was carried out while the author was a visiting researcher at DIMACS, the Center for Discrete Mathematics and Theoretical Computer Science.

1991 Mathematics Subject Classification numbers: primary 03F30; secondary 68Q15, 03F20, 03D20, 03F07, 03B20, 03B45, 03B50, 03F50.

one concerning occurrences of 0. In exchange, the overall amount of induction in the proof is restricted, by measuring the rank of the proof.

The fragment of arithmetic consisting of rank 1 proofs is not closed under modus ponens. On the one hand, this certainly seems awkward from the viewpoint of traditional approaches to logic and earlier approaches to feasible arithmetic. On the other hand, it is also the feature described by Fagin et. al. as desirable for modal reasoning using limited computational resources [8] p. 309. The present work refers to modal concepts by reading "$x = y$" as "x is (feasibly, constructively) known to be equal to y". The logical connectives and quantifiers are treated classically. Proofs are trees of formulas with axioms at the leaves and instances of either modus ponens or induction at the interior nodes.

Modal concepts are incident on the idea of *ramification* that is central to this work. Generally, a ramified mathematical system is one in which objects are defined using levels such that the definition of an object in level i refers only to objects in levels strictly below i. Such definitions are "predicative" because they do not contain any circularity of reference. A prototypical example occurs in ramified set theory, where the universe of sets at level i is defined by comprehension over sets in the previous universes. In the present work, an 'object' is an integer together with an *depth of understanding* (or: amount of knowledge, strength of comprehension) of that integer. Ramification level r ($0 \leq r \leq \omega$) consists of the integers as known with a depth of understanding at least r. Each input position i of each primitive recursive function f is labeled with a 'rank' number $\rho(f, i)$ that, intuitively, tells how much understanding of the ith input is required in order to have any understanding at all of the output [4]. One defines $\rho(f) = \max_i \rho(f, i)$. Similarly, each variable x in each proof Π is labeled with a 'rank' number $\rho(\Pi, \ulcorner x \urcorner)$, that tells how much understanding is required of the integers as referred to by x, in order to understand the proof (i.e. to understand the truth of each step of the proof). Then one defines $\rho(\Pi) = \max_{\ulcorner x \urcorner} \rho(\Pi, \ulcorner x \urcorner)$. Induction is a principle that may increase the rank of the induction variable. Although each step of the induction can be understood using only the depth of understanding required for the antecedents $\alpha[0]$ and $\alpha[x] \supset \alpha[x+1]$, the *completion* of the induction typically requires a greater amount of knowledge. Thus, induction is a principle that generates the next ramification level.

Ramification levels are analogous to the 'tiers' that Leivant defined in order to control impredicativity [13]. In Leivant's work, the strength of the logic is constrained by the explicit syntactic presence of special unary predicates. Indeed, from the presence of modal concepts in the present work one might expect to have explicit unary predicates $K_i x$ in the logic indicating that "x is understood to a depth of i". However, this is not the approach taken here. The "depth of understanding about the integers that x ranges over, required in order to understand the proof" is not something that has to be manifested within the proof itself. It can be viewed as an aspect of the proof that we are only interested in when performing an analysis of the proof. The number of ramification levels required to understand the proof, is implicit in the structure of the proof — specifically in the layering of induction instances. The purpose of this paper is to measure this implicit ramification in the syntax, using $\rho(\Pi)$, and to relate this measure to computational complexity.

The results of this paper are obtained by defining a *standard ramified model* which contains countably many copies of the integers $\mathbf{N}^0, \mathbf{N}^1, \ldots, \mathbf{N}^\omega$, each copy indicating a different depth of understanding of the integers. Let $\mathbf{K} = \cup_i \mathbf{N}^i$; the universe of the model is $\mathbf{K} \cup \{\perp\}$. Defining $\mathbf{U}^r = \cup_{i \geq r} \mathbf{N}^i \cup \{\perp\}$, the subuniverse

$\mathbf{U}^r \cap \mathbf{K}$ corresponds to ramification level r: the integers as known with a depth of understanding at least r. The model is designed to satisfy: $\mathbf{f}(\vec{\mathbf{x}}) = \perp$ iff $\exists i$, $\mathbf{x}_i \notin \mathbf{U}^{\rho(f,i)} \cap \mathbf{K}$. This formalizes the intuition that one must have at least a depth $\rho(f, i)$ of understanding of the ith input, in order to know the output. The main result shows that the conclusion of a given proof Π holds in the standard ramified model when each variable x is relativized to $\mathbf{U}^{\rho(\Pi, \ulcorner x \urcorner)}$. This formalizes the intuition that the proof can be understood by having a depth of understanding only $\rho(\Pi, \ulcorner x \urcorner)$ of the integers that x ranges over. That is, proof Π involves a depth of understanding at most $\rho(\Pi)$. On the other hand, f requires a depth of understanding at least $\rho(f)$. By a simple semantic argument, it will follow that if Π proves the convergence of f, then $\rho(\Pi) \geq \rho(f)$. Therefore, low rank proofs have limited computational complexity: they cannot prove the convergence of high-rank functions.

The subuniverses $\mathbf{U}^r$ seem to be analogous to successive worlds of an intuitionistic model. However, their existence is not presupposed by the definition of the semantics itself. Indeed, the semantics are first-order semantics, and ordinary first-order models of arithmetic are models of the system used here. Ramification levels are implicit in an arithmetic proof, not explicitly required by the semantics.

The definition of $\rho(\Pi, \ulcorner x \urcorner)$ proceeds using an analysis of proofs, including an analysis of modus ponens and $\supset$, that is entirely type level one. This contrasts with all the proof analysis methods that are based on the Curry-Howard isomorphism of formulas as types, using higher-type functionals. The present analysis works by defining a directed weighted graph, the "contribution graph" $G(\Pi)$ of the proof, which contains information about occurrences of $\supset$ in the proof. Although the specific definition of $G(\Pi)$ is based on the specific proof system used here, similar constructs should be definable for other proof systems.

A review of earlier work in weak subsystems of arithmetic is given by Krajicek [10]. For relevant background in recursion theory and computational complexity, see Clote's survey [7].

2. Ranking Recursions

The development begins by defining a recursive function class which is equivalent to the class of primitive recursive functions. A function derivation is a tree structure with leaves labeled by initial functions and internal nodes appropriately labeled by functions defined using one of the derivation rules. We frequently speak of a derivation interchangeably with the corresponding function, although of course each derivable function has many derivations. Write m_f for the arity of function f. The class $\mathcal{PR}_2$ is the class of derivations defined by:

- Initial functions: constant 0; successors $s_0 x = 2x$ and $s_1 x = 2x + 1$; predecessor $p(x) = \lfloor x/2 \rfloor$; and conditional $c(x, y, z) =$ "if $x \bmod 2 = 0$ then y else z".
- Full composition: given h and $g_1, \ldots, g_m$ with $m = m_h \geq 1$, define f by $f(\vec{x}) = h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$, where each $\vec{x}^j$ is a vector consisting of elements from the vector $\vec{x}$ (possibly with repetitions, omissions, or changes in order).
- Full recursion on notation: given g and h define f by $f(0, \vec{y}) = g(\vec{y})$ and $f(x, \vec{y}) = h(x, \vec{y}, f(px, \vec{y}))$ for $x \neq 0$.

In the recursion scheme: position 1 in f is the *recursion position*; position m_h in h is the *critical position*; the *step* function is h.

To rank all of these functions, one assigns a *rank*, $\rho(f,i)$, for each derivation $f \in \mathcal{PR}_2$ and each position $1 \leq i \leq m_f$. The rank is supposed to be a measure of how hard it is to understand the function's use of a given input. One can think of rank as being in some sense a measure of how "opaque" an input position is. In the logical setting, substitutivity of equals will fail, with the ranks of input positions being related to the amount of induction that is required to know the output of the function.

In defining $\rho(f)$, one uses the idea that when a function f is derived from subfunctions $\vec{h}$, then there is some sense in which the rank of input positions of $\vec{h}$ contribute to the rank of input positions of f. For example, if $f(x) = h(0, g(x))$, then one requires $\rho(f,1) \geq \rho(h,2)$ and $\rho(f,1) \geq \rho(g,1)$. The key point is that recursion makes it strictly more difficult to understand how a value is used. If $f(x) = h(x, f(px))$, then one requires $\rho(f,1) \geq 1+\rho(h,2)$ as well as $\rho(f,1) \geq \rho(h,1)$. The definition of $\rho(f,i)$ is:

- If f is an initial function, then $\rho(f,i) = 0$ for $1 \leq i \leq m_f$.
- If f is defined by the composition $h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$, then $\rho(f,i) = \max(\{\rho(h,j) : x_i \in \vec{x}^j\} \cup \{\rho(g_j,k) : x_i \equiv (\vec{x}^j)_k\} \cup \{0\})$ for $1 \leq i \leq m_f$.
- If f is defined by recursion from step function h and base function g, then $\rho(f,1) = \max\{\rho(h,1), 1 + \rho(h,m_h)\}$ and $\rho(f,i) = \max\{\rho(h,i), \rho(h,m_h), \rho(g,i-1)\}$ for $2 \leq i \leq m_f$.

Define the *rank* of derivations by: $\rho(f) = \max(\{\rho(f,i) : 1 \leq i \leq m_f\} \cup \{0\})$.
Define $\mathcal{PR}_2^r = \{f \in \mathcal{PR}_2 : \rho(f) \leq r\}$.

Rank differs from "tiers" (Leivant [14]) because rank is not a type system. This is not merely a syntactic distinction, as the present system admits more instances of composition and recursion than the stricter system in [14]. In particular, $\mathcal{PR}_2$ includes all the primitive recursive functions, while the system in [14] collapses at level 2 to the polynomial-time computable functions.

The following results are of interest. Readers familiar with [5] might note the correspondence: rank 0 = 'safe', rank 1 = 'normal'.

THEOREM 2.1 (Bellantoni & Cook [5]). *$\mathcal{PR}_2^1$ consists of exactly the polynomial time computable functions.*

THEOREM 2.2 (Bellantoni & Niggl [4]). *For $r \geq 2$, $\mathcal{PR}_2^r = \mathcal{E}^{r+1}$, the Grzegorczyk class of level $r+1$. In particular, $\mathcal{PR}_2^2 =$ the Kalmar elementary functions.*

3. Synthetic Arithmetic

Now a proof system is developed that is suitable for counting the rank of inductions. We would like to be able to admit a function symbol for every primitive recursive function, without having convergence of all functions proved trivially by existential generalization from reflexivity instances $f(\vec{x}) = f(\vec{x})$. To accomplish this, we view '$=$' and '$\neq$' as relations that exclude each other but which do not satisfy $x = y \vee x \neq y$. These relations can be read constructively: "$x = y$" is "x is known to be the same as y" or "there is a constructive demonstration that x is the same as y", and "$x \neq y$" is similarly read as "x is known to be different from y" or "there is a constructive demonstration that x is different from y". Other than '$=$' and '$\neq$', the logic is entirely classical; and classical semantics are used. This approach is in some sense opposite to intuitionistic logic, in which '$\supset$' and other connectives are given a constructive semantics, and in which reflexivity and substitutivity of '$=$' are treated as analytic truths. Compared to free logic, the current approach has the advantage of classical quantifier rules together with correspondingly simpler semantics.

Let Kt be the formula $\exists w, (w = t \vee w \neq t)$, where w is the first variable not appearing in term t. Under the reading above, Kt asserts that there is something which is known to be equal to or known to be different from t. Shortening this phrase a little bit, "Kt" is read as "t is known". Convergence statements are statements of the form, $\forall \vec{x} \in K,\ Kf(\vec{x})$ (that is, $\forall \vec{x}, ((\wedge_i Kx_i) \supset Kf(\vec{x}))$).

The system developed here is called "synthetic" in respect of the synthetic (constructive, non-analytic) reading of "$t = t$". There already is such a thing as "epistemic" arithmetic; see Goodman [9] and Shapiro [20]. A constructibility predicate Kx was defined by Lifschitz in his contribution to Shapiro's volume [19]; the semantic meaning is given by a form of realizability. A special predicate letter is used to delimit the numbers in Peano's original formulation. Leivant extends this to a series of special predicate letters corresponding to ramification levels [13]. A typical constructive approach to non-convergence of functions is given in Beeson's logic of partial terms [1] p. 97. For free logic one can start with Lambert's collection [11], particularly the contributions by Garson (p. 111) and Posy (p. 49). The present development is, as far as we know, original.

Let $\mathcal{SFO}(F)$ (*synthetic first-order logic over F*) be the logic having the following axioms and rules, defined over the language with function symbols F, relation symbols '$=$' and '$\neq$', and logical connectives $\supset$, $\neg$, $\forall$. The other connectives are defined classically. Proofs are trees of formulas. Uniform substitution of term t for variable x in α is indicated by $\alpha[t/x]$. To improve readability a comma is used after quantifiers; and abbreviations such as "$\forall x \in K,\ \alpha$" for "$\forall x, (Kx \supset \alpha)$" are used. In the (FO) axioms below, $\vec{z}$ is a list consisting of some or all of the free variables in the subsequent formula. In (KS) below, $\vec{a}$ and $\vec{b}$ are terms with free variables $\vec{z}$.

The (FO) axioms are the axioms of classical first-order predicate calculus without equality (adapted from [16]). The other axioms are modifications of the usual axioms for equality.

(FO.1) $\forall \vec{z}, (\alpha \supset (\beta \supset \alpha))$
(FO.2) $\forall \vec{z}, ((\alpha \supset (\beta \supset \gamma)) \supset ((\alpha \supset \beta) \supset (\alpha \supset \gamma)))$
(FO.3) $\forall \vec{z}, ((\neg\alpha \supset \neg\beta) \supset (\beta \supset \alpha))$
(FO.4) $\forall \vec{z}, ((\forall x, (\alpha \supset \beta)) \supset ((\forall x, \alpha) \supset (\forall x, \beta)))$
(FO.5) $\forall \vec{z}, ((\forall x, \alpha) \supset \alpha[t/x])$ (t is free for x in α)
(FO.6) $\forall \vec{z}, (\alpha \supset \forall y, \alpha)$ (y is not free in α)

(E) $\forall x, y, (x = y \supset \neg(x \neq y))$
(¬E) $\forall x, y, (x \neq y \supset \neg(x = y))$
(KR=) $\forall x, (Kx \supset x = x)$
(S=) $\forall x, y, (x = y \supset y = x)$
(¬S=) $\forall x, y, (x \neq y \supset y \neq x)$
(T=) $\forall x, y, z, ((x = y \wedge y = z) \supset x = z)$
(¬T=) $\forall x, y, z, ((x = y \wedge y \neq z) \supset x \neq z)$
(KS) $\forall \vec{z}, ((\vec{a} = \vec{b} \wedge Kf(\vec{a}) \wedge Kf(\vec{b})) \supset f(\vec{a}) = f(\vec{b}))$

The only deduction rule of $\mathcal{SFO}(F)$ is (MP): $\alpha,\ \alpha \supset \beta \mid \beta$.

These axioms are meant to formalize, first of all, the idea that $x = y \vee x \neq y$ is not valid; and second of all, that deductions involving $=$ and $\neq$ can be understood as ones involving constructive knowledge. For example, if one knows that $a = b$, and one knows what $f(a)$ and $f(b)$ are, then in fact one knows that $f(a) = f(b)$. But simply from knowing $a = b$, one does not know $f(a) = f(b)$: function inputs are opaque contexts.

A very convenient feature of $\mathcal{SFO}(F)$ is that models of $\mathcal{SFO}(F)$ are just first-order models in which the interpretations of '$=$' and '$\neq$', and the interpretation of the function symbols, satisfy the second group of axioms above. The system $\mathcal{SFO}(F)$ is complete for these models because $\mathcal{SFO}(F)$ is an extension of first-order logic without equality (FO.1-6 above). Of course, models of $\mathcal{SFO}(F)$ may interpret '$=$' in a non-normal way i.e. $\mathrm{x} = \mathrm{y}$ with x and y being different points in the model.

For the purposes of this work, it is appropriate to include extra copies of the function symbol for the initial function 0. Zero is special because it appears explicitly in the induction rule. Let F_2 consist of one function symbol for each $\mathcal{PR}_2$ derivation, except that for the initial function 0 put countably many function symbols, $\{0_i : 0 \leq i < \omega\}$ into F_2. (Throughout, ω is the first limit ordinal.) One could use only a single symbol for zero if one were willing to accept that (ind) be restricted to induction over 0-free formulas; see the definition of (ind) below.

A base system $\mathcal{SQ}_2$ (*synthetic Q in base 2*) is defined by adding the following axioms to $\mathcal{SFO}(F_2)$, for $0 \leq i, j \leq \omega$. First are axioms for 0, s, p, and c:

(0.1)	$0_i = 0_j$	(s.6)	$\forall x, y \in K,\ (s_0 x = s_0 y \supset x = y)$
(s.1)	$s_0 0_i = 0_j$	(s.7)	$\forall x, y \in K,\ (s_1 x = s_1 y \supset x = y)$
(s.2)	$\forall x \in K,\ (s_1 x \neq 0_i)$	(p.1)	$p 0_i = 0_j$
(s.3)	$\forall x \in K,\ (s_0 x \neq s_1 0_i)$	(p.2)	$\forall x \in K,\ (p s_1 x = x)$
(s.4)	$\forall x \in K,\ (s_0 x \neq s_1 x)$	(p.3)	$\forall x \in K,\ (p s_0 x = x)$
(s.5)	$\forall x \in K,\ (s_0 x \neq s_1 x)$	(p.4)	$\forall x \in K,\ (x = s_0 p x \vee x = s_1 p x)$
		(c.1)	$\forall x, y, z \in K,\ (x = s_0 p x \supset c(x, y, z) = y)$
		(c.2)	$\forall x, y, z \in K,\ (x = s_1 p x \supset c(x, y, z) = z)$

The remaining axioms are for f defined by composition or recursion. They are paraphrased below.

(comp) $(\forall \vec{x} \in K,\ Kh(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m)))$
$\supset (\forall \vec{x} \in K,\ (f(\vec{x}) = h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))))$
where f is defined by composition from h and $g_1, \ldots, g_m$.

(rec.1) $((\forall \vec{y} \in K,\ Kg(\vec{y})) \wedge (\forall x, \vec{y}, z \in K,\ Kh(x, \vec{y}, z)))$
$\supset (\forall \vec{y} \in K,\ f(0_i, \vec{y}) = g(\vec{y}))$
where f is defined by recursion from h and g.

(rec.2) $((\forall \vec{y} \in K,\ Kg(\vec{y})) \wedge (\forall x, \vec{y}, z \in K,\ Kh(x, \vec{y}, z)))$
$\supset (\forall x, \vec{y} \in K,\ (x \neq 0_j \supset (Kf(px, \vec{y}) \supset f(x, \vec{y}) = h(x, \vec{y}, f(px, \vec{y})))))$
where f is defined by recursion from h and g.

The understanding of (comp) is: if one knows what $h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$ is for all known $\vec{x}$, then one knows that $f(\vec{x}) = h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$ for all known $\vec{x}$. For (rec.1): if the base function, g, is known, and the step function, h, is known, then for all known $\vec{y}$, one knows that $f(0, \vec{y})$ is equal to $g(\vec{y})$. For (rec.2): if the base function g is known, and the step function h is known, then for all values x that are known to be different from 0, and for all known $\vec{y}$, if one knows $f(px, \vec{y})$ then one knows that $f(x, \vec{y})$ equals $h(x, \vec{y}, f(px, \vec{y}))$.

Note. In the recursion axiom (rec.1), the literal value 0_i is specified in the base term $f(0_i, \vec{y})$. Since models may be non-normal, this formulation does not constrain f on all x such that $x = 0_i$. Definition: a *shadow of zero* is a point x in a model such that $\mathrm{x} = \mathbf{0}_i$ in the model, for some i. By (0.1) it doesn't matter which i one uses. On the syntactic side, the various function symbols 0_i, $i \geq 0$ are *variant names for* 0. By (0.1), every variant name for 0 denotes a shadow of 0. A shadow of 0 may be denoted by more than one variant name, and there may be shadows of 0 that are not denoted by any variant name. The use of variant names for zero does not constitute an explicit reference to ramification levels. The variant names of zero are interchangeable in the axioms. They are treated symmetrically rather than ordered into successive layers.

Finally one obtains $\mathcal{SA}_2$ (*synthetic arithmetic in base 2*) by adding an induction rule to $\mathcal{SQ}_2$. In this rule, one requires as usual that x and z do not already appear in α; furthermore, *it is required that* 0_i *does not already appear in* α. The (ind) rule is:

$$\frac{\alpha[0_i/y] \qquad \forall x \in K,\ (x \neq 0_j \supset (\alpha[px/y] \supset \alpha[x/y]))}{\forall z \in K,\ \alpha[z/y]}$$

Other than the restriction concerning 0_i, x and z, the induction formula can be arbitrary; for example, it can contain any number of unbounded alternating quantifiers.

Note. Of considerable interest in the induction rule is the fact that the conclusion of induction says that $\alpha[z/y]$ holds at all $z \in K$; in particular this includes all the shadows of zero. Thus, a use of induction constrains the truth value of α at all shadows of 0, even though in the premisses one is only required to have specified the truth value of α at one of the shadows of zero, namely 0_i. Compare with (rec.1). It is only through occurrences of induction that the value of a recursively defined function is constrained on all shadows of zero.

The restriction that 0_i does not already appear in α, is mitigated by the fact that one can use other variant names 0_j $(j \neq i)$ in α. In fact this was the purpose of introducing variant names for zero; the alternative would be to have only one symbol for 0, and not allow it into the induction formula $\alpha[y]$. The variant names for zero are equivalent in the sense that they are all treated in the same way by the axioms. The point, then, is that the variant name 0_i used to form the base case of the induction has in some sense a kind of independence from all the other names in α: there is no reason why 0_i has to denote identically the same element as 0_j, $j \neq i$. In other words it is the axiom system (0.1, rec, etc), rather than just a coincidence of names, that determines the relationship between, on the one hand, 'zero used as the base value for the induction variable' and, on the other hand, 'zero used to refer to a fixed element of the universe independent of the value of the induction variable'. This use of 0_i and 0_j contrasts with the ordinary unramified formulation in which such a distinction cannot be made.

4. Ranking Proofs

Now a rank, $\rho(\Pi)$, is assigned for each proof Π. But before defining $\rho(\Pi)$, something should be said about fairness. One trivial way to have rank r provability

correspond to rank r functions, would be to let the rank of a proof be the maximum of the ranks of the function symbols appearing in it. But this would be cheating, because then rank r proofs essentially could only use the definitions of rank r functions. Obviously such a system could not prove the convergence statement for a function symbol outside of $\mathcal{PR}_2^r$. One should exclude from consideration any definition of $\rho(\Pi)$ which refers to $\rho(f)$, or in which $\rho(\Pi)$ is known to be large simply due the use of $\mathcal{SQ}_2$ axioms for a lot of functions of high rank. To be fair, the use of an $\mathcal{SQ}_2$ defining axiom for a function should cost nothing in terms of the rank of the proof. This does not mean that the statement of the definition of $\rho(\Pi)$ has to be independent of how and where composition and recursion axioms are used; only that uses of composition and recursion axioms should not increase $\rho(\Pi)$, either by themselves or in combination with (MP) and the logical axioms. In other words, every $\mathcal{SQ}_2$ proof should have rank 0. Thus, induction must be critical to the definition of rank.

A graph $G(\Pi)$, the *constraint graph of* Π, is used to define $\rho(\Pi)$. The constraint graph represents auxiliary information about the proof, namely certain constraints that ρ must satisfy. Let V be the set of code numbers $\{\ulcorner x\urcorner, \ulcorner y\urcorner, \dots\}$ of variables in the language of $\mathcal{FO}$, and let $\{\ulcorner 0_i\urcorner : i \geq 0\}$ be code numbers for the variant names of 0. The nodes of $G(\Pi)$ are $V \cup \{\ulcorner 0_i\urcorner : i \geq 0\}$. A 'rank' will be assigned to each $\ulcorner 0_i\urcorner$ as well as to each $\ulcorner x\urcorner \in V$. Write $\langle \ulcorner u\urcorner, \ulcorner v\urcorner, d\rangle$ for an edge from $\ulcorner u\urcorner$ to $\ulcorner v\urcorner$ of weight d.

The edge $\langle \ulcorner u\urcorner, \ulcorner v\urcorner, d\rangle$ represents a constraint, that the rank of v must be at least as great as d plus the rank of u.

Let $\Pi_1, \Pi_2, \dots$ be the subproofs, in left to right order, leading to the premisses of the last step of Π. The notation $\ulcorner y\urcorner \in t$ or $\ulcorner 0_i\urcorner \in t$ means that the variable variable y or constant 0_i has an occurrence in the term t. Note that the quantified variables in (comp) and (rec.1-2) are not arbitrary but are in some cases the same for two quantifiers. This will save us a few edges in $G(\Pi)$, because edges of the form $\langle \ulcorner x\urcorner, \ulcorner x\urcorner, 0\rangle$ can be omitted; they would be redundant under the subsequent definition of ρ. The graph $G(\Pi)$ is defined by cases on the last rule in Π, as follows.

Ax Any axiom except (FO.5) or (rec.1).

(G.1) $G(\Pi) = \emptyset$

(FO.5) $\forall \vec{z}$, $((\forall x, \alpha) \supset \alpha[t/x])$, where t is free for x in α

(G.2) $G(\Pi) = \{\langle \ulcorner x\urcorner, \ulcorner y\urcorner, 0\rangle : \ulcorner y\urcorner \in t, \ulcorner y\urcorner \neq \ulcorner x\urcorner\} \cup \{\langle \ulcorner x\urcorner, \ulcorner 0_i\urcorner, 0\rangle : 0_i \in t\}$

(rec.1) $((\forall \vec{y} \in K,\ Kg(\vec{y})) \wedge (\forall x, \vec{y}, z \in K,\ Kh(x, \vec{y}, z)))$
$\supset (\forall \vec{y} \in K,\ f(0_i, \vec{y}) = g(\vec{y}))$

(G.3) $G(\Pi) = \{\langle \ulcorner x\urcorner, \ulcorner 0_i\urcorner, 0\rangle\} \cup \{\langle \ulcorner z\urcorner, \ulcorner 0_i\urcorner, 1\rangle\} \cup \{\langle \ulcorner z\urcorner, \ulcorner y_j\urcorner, 0\rangle : 1 \leq j \leq m_f - 1\}$

(MP) $\alpha,\ \alpha \supset \beta \mid \beta$

(G.4) $G(\Pi) = G(\Pi_1) \cup G(\Pi_2)$.

(ind) $(\alpha[0_i/y]),\ (\forall x \in K,\ (x \neq 0_j \supset (\alpha[px/y] \supset \alpha[x/y]))) \mid (\forall z \in K,\ \alpha[z/y])$

(G.5) $G(\Pi) = G(\Pi_1) \cup G(\Pi_2) \cup \{\langle \ulcorner 0_i\urcorner, \ulcorner z\urcorner, 0\rangle\} \cup \{\langle \ulcorner x\urcorner, \ulcorner z\urcorner, 0\rangle\}$

This completes the definition of $G(\Pi)$. Next we will define $\rho(\Pi, \ulcorner x\urcorner)$ for $\ulcorner x\urcorner \in V$. Approximately speaking, $\rho(\Pi, \ulcorner x\urcorner)$ is the supremum of the weights of all finite paths that end at $\ulcorner x\urcorner$. In this way, one has $\rho(\Pi, \ulcorner x\urcorner) \geq d + \rho(\Pi, \ulcorner y\urcorner)$ whenever there is a finite path from y to x of total weight d. Thus, ρ satisfies the constraints expressed

by the edges in $G(\Pi)$. In fact, it will be simplest to define ρ to be the minimum function satisfying these constraints.

One hopes that a suitable renaming of variables in Π would guarantee that $G(\Pi)$ contains no positively weighted cycles. But instead of attempting such a renaming, we simply ignore any proofs whose constraint graph contains a positively weighted cycle. This is accomplished by assigning rank ω when a cycle occurs. Evidently, not too many proofs are ignored this way: later it will be shown that there are finite-rank proofs of convergence of all the primitive recursive functions.

A *ranking function*, q, is a total function from $V \cup \{\ulcorner 0_i\urcorner : i \geq 0\}$ to $\mathbf{N} \cup \{\omega\}$, where $\mathbf{N}$ is the non-negative integers. Say that q is *good for* G, written $q \geq G$, iff: $\langle \ulcorner u\urcorner, \ulcorner v\urcorner, d\rangle \in G$ implies $q(\ulcorner v\urcorner) \geq d + q(\ulcorner u\urcorner)$. If q is good for G, then in fact q is good for every subgraph of G. For a nonempty set of ranking functions Q, the pointwise minimum 'min Q' is the function q' such that $q'(\ulcorner u\urcorner) = \min\{q(\ulcorner u\urcorner) : q \in Q\}$, for all $\ulcorner u\urcorner \in V \cup \{\ulcorner 0_i\urcorner : i \geq 0\}$. It happens that if every $q \in Q$ is good for G, then so is $\min Q$. By assigning $q(\ulcorner u\urcorner) = \omega$ in case $\ulcorner u\urcorner$ is in a positively weighted cycle in $G(\Pi)$, one can guarantee that $\{q :\ q \geq G(\Pi)\}$ is nonempty. For details, see parts (1a) and (2) of the next lemma below.

Writing $\rho(\Pi, \cdot)$ for $\lambda n.\rho(\Pi, n)$, define:

$$\rho(\Pi, \cdot) = \min\{q :\ q \geq G(\Pi)\}.$$

One has that $\rho(\Pi, \ulcorner x\urcorner) = 0$ for all $\ulcorner x\urcorner \notin \Pi$. See (1b) of the next lemma below. Define the *rank* of proof Π by:

$$\rho(\Pi) = \max\{\rho(\Pi, \ulcorner x\urcorner) : \ulcorner x\urcorner \in V\}.$$

Notice that $\rho(\Pi)$ does not depend on $\rho(\Pi, \ulcorner 0_i\urcorner)$ unless there is an edge from $\ulcorner 0_i\urcorner$ to some $\ulcorner x\urcorner \in V$.

The definition of $\rho(\Pi)$ is "fair" in the sense discussed earlier, even though weighted edges are associated with the defining axioms of the functions. Although it is true that a positively weighted edge is added for an instance of a function definition (rec.1, G.3), such an edge ends at a name for zero rather than at a variable. The weight of this edge does not contribute to $\rho(\Pi)$ until a path containing the edge is extended by a (G.5) edge to reach the induction variable in an (ind) conclusion. This is because the definition of $\rho(\Pi)$ only refers to $q(\ulcorner x\urcorner)$ for variables $\ulcorner x\urcorner \in V$, and an (ind) instance is the only place where an edge passes from some $\ulcorner 0_i\urcorner$ to some variable $\ulcorner x\urcorner \in V$. Uses of the axioms defining the functions do not increase the rank either by themselves or in combination with (MP) and the logical axioms: all $\mathcal{SQ}_2$ proofs have rank 0. Thus, induction is critical. Say that two occurrences of (ind) are *nested* if there is a path from any literal in the conclusion of one, to any literal in any antecedent of the other. If a proof has rank $r < \omega$, then it must contain instances of induction nested r deep: the graph $G(\Pi)$ contains a path passing through r different names for zero, corresponding to r interleavings of (G.3/rec.1) and (G.5/ind).[1]

[1] Consider a rank r proof, $r < \omega$. Its graph contains a path of weight r ending at a variable, in which each node of the path is labeled with a different element of $V \cup \{\ulcorner 0_i\urcorner : i \geq 0\}$. The graph doesn't contain any weighted cycles, because $r < \omega$. But the weighted edges in a graph are exactly the edges that pass from some variable $\ulcorner x\urcorner$ to some name for zero, $\ulcorner 0_i\urcorner$ (they are (G.3) edges). Consider any two weighted edges on the path, such that no weighted edge appears between them. In order to form a path, there must be a "return" subpath from the literal $\ulcorner 0_i\urcorner$ to the variable $\ulcorner y\urcorner$ that starts the next weighted edge. There also must be a similar "return" subpath from the last $\ulcorner 0_j\urcorner$ to the variable at the end of the path. These "return" subpaths must each start with a (G.5)

The converse is not true in general. In a rank 1 proof there may be arbitrarily many (ind) occurrences, and these may be arbitrarily deeply nested. This is analogous to the fact that a rank 1 function derivation may contain arbitrarily deeply "nested" recursions, for example if each recursion is on a different input. Not all occurrences of induction have new computational content. Viewed differently, an occurrence of induction on a variable x might happen to generate a ramification level that is less than the ramification levels already generated by some other inductions in other parts of the proof. We wouldn't want to increase the overall proof rank for such an occurrence of induction. Essentially, the presence of weighted edges in some of the non-induction parts of the proof, is a way of determining whether or not subsequent inductions have new computational content i.e. generate a new ramification level. An induction that does not use a new statement of fact (i.e. a defining axiom for a function) is not one that needs to be accounted for.

In summary, although the definition of proof rank uses the recursion axioms for the primitive recursive functions, proof rank is not a disguise for function rank. Proof rank relates to the layering of inductions. Function rank relates to the layering of recursions. These are two independent concepts. A low-rank proof can refer to high-rank functions, and a high-rank proof might refer only to low-rank functions.

The following lemma was promised above.

LEMMA 4.1. *For every proof Π, (1a) there is a ranking function q that is good for $G(\Pi)$, such that (1b) if x is a variable not in Π, then $q(\ulcorner x \urcorner) = 0$. Furthermore, (2) if Q is any nonempty set of ranking functions such that every $q \in Q$ is good for $G(\Pi)$, then $\min Q$ is also good for $G(\Pi)$.*

PROOF. (1) The required ranking function assigns the first limit ordinal, ω, to all the (codes of) variables that appear in the proof, and assigns 0 to all the (codes of) variables that do not appear in the proof. It is a ranking function because of the following. *Fact:* if $0 \leq d < \omega$ then $d + \omega = \omega$, although $\omega + d > \omega$. See Mendelson [17] p. 204.

Actually, if $G(\Pi)$ does not contain any cycles, then one can obtain a ranking function in which all ranks are finite: ω only needs to be assigned when the variable is in a cycle in the graph. However, a proof of this this is not formally required for the present work.

(2) Assume that Q is non-empty and that every $q \in Q$ is good for $G(\Pi)$. Let $q' = \min Q$ and consider any edge $\langle \ulcorner u \urcorner, \ulcorner v \urcorner, d \rangle \in G(\Pi)$. Let q be a function in Q minimizing $q(\ulcorner v \urcorner)$; then $q'(\ulcorner v \urcorner) = q(\ulcorner v \urcorner)$ by the definition of $\min Q$. Since q is good for $G(\Pi)$, one has $q(\ulcorner v \urcorner) \geq d + q(\ulcorner u \urcorner)$. By the definition of $\min Q$ again, $q(\ulcorner u \urcorner) \geq q'(\ulcorner u \urcorner)$. Therefore, $q'(\ulcorner v \urcorner) = q(\ulcorner v \urcorner) \geq d + q(\ulcorner u \urcorner) \geq d + q'(\ulcorner u \urcorner)$. It follows that q' is good for $G(\Pi)$. □

5. Standard Ramified Model

Now the development turns to semantic issues.

Let $\perp$ be a unique element, and let $\mathbf{N}^0, \mathbf{N}^1, \mathbf{N}^2, \ldots$ together with $\mathbf{N}^\omega$ be countably many copies of the non-negative integers.

Define a *depth* for each point in $\cup_j \mathbf{N}^j \cup \{\perp\}$ as follows: $\delta(\mathbf{x}) = j$ for all $\mathbf{x} \in \mathbf{N}^j$, and $\delta(\perp) = -1$. The intuitive meaning is that $\delta(\mathbf{x})$ is the depth of knowledge that one has of $\mathbf{x}$: the greater the depth, the greater one's understanding of $\mathbf{x}$, and the more able one is to perform induction with $\mathbf{x}$ or to compute functions with $\mathbf{x}$ as an input. The multiple copies of an integer do not represent different *values*, but different *amounts of understanding* of a single integer value.

A vector $\vec{\mathbf{x}} \in (\cup_j \mathbf{N}^j \cup \{\perp\})^{(m_f)}$ of length m_f is *adequate* for $f \in \mathcal{PR}_2$ if $\delta(\mathbf{x}_i) \geq \rho(f, i)$ for all i, $1 \leq i \leq m_f$. For example, for every function f, every vector consisting of m_f many elements from $\mathbf{N}^\omega$ is adequate for f. The empty vector is adequate for every zero-ary function. A vector containing $\perp$ is inadequate for every function.

Some more notation will be helpful. For a point $\mathbf{x} \in \cup_j \mathbf{N}^j$, let $\underline{\mathbf{x}}$ be the corresponding integer (i.e. the integer without indication of which set $\mathbf{N}^j$ it is in). For an integer $\mathbf{n}$, let $[\underline{\mathbf{n}}]^j$ be the copy of $\mathbf{n}$ in $\mathbf{N}^j$. For a function symbol f, one writes $\mathbf{f}$ for the interpretation of f in a given model, and $\underline{\mathbf{f}}$ for the standard function on the integers defined by the $\mathcal{PR}_2$ derivation. Given an assignment A and a model M, one writes $\mathbf{t}$ for the interpretation of term t under (M, A).

The *standard ramified model* is the model M_0 defined as follows. The universe of M_0 is $\cup_j \mathbf{N}^j \cup \{\perp\}$. Define $\mathbf{x} = \mathbf{y}$ iff: $\delta(\mathbf{x}) \geq 0$, and $\delta(\mathbf{y}) \geq 0$, and $\underline{\mathbf{x}}$ is identically $\underline{\mathbf{y}}$. Similarly, define $\mathbf{x} \neq \mathbf{y}$ iff: $\delta(\mathbf{x}) \geq 0$, and $\delta(\mathbf{y}) \geq 0$, and $\underline{\mathbf{x}}$ is different from $\underline{\mathbf{y}}$. The interpretation $\mathbf{f}$ of function symbol f is defined by:

$$\mathbf{f}(\vec{\mathbf{x}}) \equiv \begin{cases} [\underline{\mathbf{f}}(\underline{\vec{\mathbf{x}}})]^{\min(\{\delta(\mathbf{x}_i):\, 1 \leq i \leq m_f\} \cup \{\omega\})} & \text{if } \vec{\mathbf{x}} \text{ is adequate for } f \\ \perp & \text{otherwise.} \end{cases}$$

Under this definition, if $\vec{\mathbf{x}}$ is adequate for f and $m_f \geq 1$, then $\delta(\mathbf{f}(\vec{\mathbf{x}})) = \min \delta(\vec{\mathbf{x}})$. If f is zero-ary, then $\delta(\mathbf{f}()) = \omega$. For example, in M_0 every variant name 0_i is interpreted as $[\underline{\mathbf{0}}]^\omega$. For every j, the predecessor of $[\underline{\mathbf{0}}]^j$ is $[\underline{\mathbf{0}}]^j$; in fact $\delta(\mathbf{px}) = \delta(\mathbf{x})$ for every $\mathbf{x}$.

Recall $Kt \equiv \exists w, (w = t \vee w \neq t)$; the interpretation $\mathbf{K}$ of K is $\cup_j \mathbf{N}^j$. The model does not satisfy $\forall x, Kx$, because $\perp \notin \mathbf{K}$. Say that a function *diverges* on a given vector of points if the output of the function is not in $\mathbf{K}$; otherwise it *converges*. In these models, functions converge iff the input is adequate. For example, if any input is $\perp$ then the output is also $\perp$.

M_0 has been presented to make the basic structure apparent. In fact the following will be used instead. A *variant standard ramified model* is a model M that is the same as M_0 except the variant names 0_i are interpreted by any shadows of 0, not just by $[\underline{\mathbf{0}}]^\omega$. Other constant symbols are still given interpretations in $\mathbf{N}^\omega$. One must be careful to remember that the subscript i in 0_i refers to the variant name, not to the depth $\delta(\mathbf{0}_i)$ of the shadow $\mathbf{0}_i$ denoted by 0_i. Note that M_0 itself is a variant model.

Given a ranking function q, say that M is *good for* q, written $M \geq q$, iff M is a variant standard ramified model such that $\delta(\mathbf{0}_i) \geq q(\ulcorner 0_i \urcorner)$ for all i. Note $M_0 \geq q$ for every ranking function q. When one considers all possible models $M \geq q$ for a given q, the interpretation of 0_i is essentially relativized to $\{[\underline{\mathbf{0}}]^j : q(\ulcorner 0_i \urcorner) \leq j \leq \omega\}$.

None of the variant models is a model of the theorems of $\mathcal{SA}_2$. However, it will be shown that a suitable relativization of each $\mathcal{SA}_2$ theorem is satisfied.

In order to relativize variables, let U^j $(1 \leq j \leq \omega)$ be new unary predicate letters. One should realize that the symbols U^j are only being introduced in order to carry out the analysis, and they do not appear in the definition of $\mathcal{SA}_2^r$ or in the final statement of the results. Give U^j the interpretation $\mathbf{U}^j \equiv \cup_{i \geq j} \mathbf{N}^i \cup \{\perp\}$ in every variant model M. Thus, $j \geq k$ implies $\mathbf{U}^j \subseteq \mathbf{U}^k$. As j increases, the subuniverses

edge corresponding to an instance of induction, because (G.5) is the only place where an edge is added from a zero to a variable. These r instances of (ind) are nested, and they are distinct because they use different names 0_i in their base cases.

$\mathbf{U}^j$ get smaller and smaller, and more and more functions are total on $\mathbf{U}^j \cap \mathbf{K}$; in some sense, more and more is known over $\mathbf{U}^j \cap \mathbf{K}$. This is like the successive worlds of an intuitionistic Kripke model, where more and more propositions hold as one moves farther and farther along a sequence of successive worlds.

For a formula α, let α^q be obtained from α by relativizing each variable x to $U^{q(|x|)}$. The relativization replaces each quantifier $\forall x$, α with $\forall x$, $(U^{q(|x|)}x \supset \alpha)$, and prefixes the formula with "$U^{q(|x|)}x \supset$" for each free variable x.

The following closure property of $\mathbf{U}^j$ is used for analyzing substitution (FO.5).

LEMMA 5.1 (Closure of $\mathbf{U}^j$). *Given a term t and any $j \geq 0$, let A be an assignment for a variant M such that $\mathbf{y} \in \mathbf{U}^j$ for all $y \in t$ and $\mathbf{0}_i \in \mathbf{U}^j$ for all $0_i \in t$. Then $\mathbf{t} \in \mathbf{U}^j$.*

PROOF. It is proved by induction on the term structure of t. If t is just a variable y, then the statement of the lemma follows trivially by the assumption $\mathbf{y} \in \mathbf{U}^j$. If t is a zero-ary function symbol f other than some 0_i, then $\delta(\mathbf{t}) = \omega$ and the statement again follows. If t is some 0_i then it follows trivially because $\mathbf{0}_i \in \mathbf{U}^j$ by assumption.

Otherwise, t is $f(\vec{a})$ and the arity of f is $m_f \geq 1$. By the induction hypothesis on each term in $\vec{a}$, one has $\vec{\mathbf{a}} \in \mathbf{U}^j$, i.e. $\delta(\mathbf{a}_i) = -1$ or $\delta(\mathbf{a}_i) \geq j$ (for $1 \leq i \leq m_f$). Under the definition of M_0, if $\vec{\mathbf{a}}$ is inadequate for f then $\mathbf{f}(\vec{\mathbf{a}})$ is $\perp$, in which case $\mathbf{f}(\vec{\mathbf{a}}) \in \mathbf{U}^j$. On the other hand, if $\vec{\mathbf{a}}$ is adequate for f then $\delta(\mathbf{a}_i) \geq j$ for each i, because $\delta(\mathbf{a}_i) < j \Rightarrow \delta(\mathbf{a}_i) = -1 \Rightarrow \vec{\mathbf{a}}$ is inadequate for f. Therefore, if $\vec{\mathbf{a}}$ is adequate for f (with $m_f \geq 1$) one has $\delta(\mathbf{f}(\vec{\mathbf{a}})) = \min\{\delta(\mathbf{a}_i) : 1 \leq i \leq m_f\} \geq j$, again implying $\mathbf{f}(\vec{\mathbf{a}}) \in \mathbf{U}^j$. □

6. Implicit Ramification Theorem

Now we have arrived at the key "implicit ramification" theorem relating the ranking of induction to the ramification subuniverses U^j. The theorem says that if Π proves α and $q \geq G(\Pi)$, then $M \models \alpha^q$ for every $M \geq q$ (including M_0). A simple semantic argument (see §7) will then show that if Π proves the convergence of f, then $\rho(\Pi) \geq \rho(f)$.

The proof of the implicit ramification theorem is unfortunately long, since we must consider each axiom and rule of $\mathcal{SA}_2$. It is broken into two lemmas. The first lemma considers the simpler cases: all axioms other than (FO.5) and (rec.1). This lemma does involve some work, for (comp) and (rec.2). The second lemma handles the most interesting cases, finishing the theorem by treating the relativizations of (FO.5), (rec.1), (MP), and (ind).

In variant models, $(Kt)^q$ holds if and only if Kt holds: there is something equal or not equal to t, iff there is something in $\mathbf{N}^\omega$ that is equal or not equal to t. Therefore, when making semantic arguments one can ignore relativizations of the quantifier in K. In the following we drop the relativization and refer only to $\mathbf{K}$.

LEMMA 6.1. *Let Π be an instance of an $\mathcal{SA}_2$ axiom other than (FO.5) or (rec.1), let $q \geq G(\Pi)$ and let $M \geq q$. Then M satisfies the q-relativization of the axiom instance.*

PROOF. The relativizations of the axioms (FO.1), (FO.2), (FO.3), (FO.4), and (FO.6) follow by standard first-order semantic reasoning.

Consider the equality axioms of $\mathcal{SFO}$. The relativization of each of these axioms consists of the same axiom with the universal quantifiers relativized to a

subuniverse. The restriction to a subuniverse only weakens the assertion. So, if the unrelativized form is satisfied by M then the relativized form is also satisfied by M. By the construction, the unrelativized axioms are satisfied by M. Substitution (sub) is a typical example. Assume $\vec{a} = \vec{b}$ and $Kf(\vec{a})$ and $Kf(\vec{b})$ are satisfied. Since $\mathbf{Kf}(\vec{\mathbf{a}})$ and $\mathbf{Kf}(\vec{\mathbf{b}})$ hold, it must be that $\mathbf{f}(\vec{\mathbf{a}})$ is $[\underline{\mathbf{f}}(\vec{\mathbf{a}})]^j$ and $\mathbf{f}(\vec{\mathbf{b}})$ is $[\underline{\mathbf{f}}(\vec{\mathbf{b}})]^k$ for some j and k. Since $\vec{\mathbf{a}} = \vec{\mathbf{b}}$, it must be that the underlying integers are the same: $\underline{\vec{\mathbf{a}}}$ is identically $\underline{\vec{\mathbf{b}}}$. Therefore $\underline{\mathbf{f}}(\underline{\vec{\mathbf{a}}})$ is identically $\underline{\mathbf{f}}(\underline{\vec{\mathbf{b}}})$, implying $\mathbf{f}(\vec{\mathbf{a}}) = \mathbf{f}(\vec{\mathbf{b}})$ in M, as required by the conclusion of substitutivity.

Now consider the $\mathcal{SQ}_2$ axioms for the initial functions (0.1-2), (s.1-7), (p.1-3) and (c.1-2). Without belaboring details, these axioms are all satisfied by the construction of M, because: the initial functions 0_i, s_0, s_1, p and c converge everywhere in $\mathbf{K}$ and output a value in $\mathbf{K}$ whose underlying integer is the output of the standard function; the depth of the value in $\mathbf{K}$ does not matter to the '=' and '$\neq$' relations; and the corresponding statements are true in the standard model of arithmetic. The relativization of the universal quantifiers in these axioms only weakens the statement, so the relativizations also hold.

What remains is (comp) and (rec.2). There are no free variables in the relativized form of the (comp) axiom:

$$(\forall \vec{x} \in U^{q(|\vec{x}|)} \cap K,\ Kh(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))) \\ \supset (\forall \vec{x} \in U^{q(|\vec{x}|)} \cap K,\ f(\vec{x}) = h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m)))$$

The argument can be summarized as follows. The relativized antecedent asserts that vectors of depth $q(|\vec{x}|)$ are adequate for $\lambda\vec{x}.h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$. Under the definition of ρ, it will follow that vectors of depth $q(|\vec{x}|)$ are adequate for f. In M_0, the convergence of both sides of "$f(\vec{x}) = h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$" ensures that the equality holds, because f was defined in $\mathcal{PR}_2$ by composition from h and $g_1, \ldots, g_m$.

- To show that $(\text{comp})^q$ is satisfied in M, let A be any assignment. Assume that for all $\vec{\mathbf{x}} \in \mathbf{U}^{q(|\vec{x}|)} \cap \mathbf{K}$ one has $M, A|^{\vec{\mathbf{x}}}_{\vec{x}} \models Kh(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$. One must show, for $\vec{\mathbf{x}} \in \mathbf{U}^{q(|\vec{x}|)} \cap \mathbf{K}$, that $M, A|^{\vec{\mathbf{x}}}_{\vec{x}} \models f(\vec{x}) = h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$. Choose any $\vec{\mathbf{x}} \in \mathbf{U}^{q(|\vec{x}|)} \cap \mathbf{K}$. Let $\mathbf{G}_j$ be the interpretation of $g_j(\vec{x}^j)$ and let $\mathbf{F}$ be the interpretation of $f(\vec{x})$. Recall that $\rho(f,i) = \max(\{\rho(h,j) : x_i \in \vec{x}^j\} \cup \{\rho(g_j,k) : x_i \equiv (\vec{x}^j)_k\} \cup \{0\})$ for $1 \leq i \leq m_f$.

  Since $Kh(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m)))$ holds, it must be that $\vec{\mathbf{G}}$ is adequate for h, because h only converges on adequate inputs. It follows that $x_i \in \vec{x}^j \Rightarrow \delta(\mathbf{x}_i) \geq \rho(h,j)$ for all i and j, because: if $x_i \in \vec{x}^j$ and $\delta(\mathbf{x}_i) < \rho(h,j)$ then $\delta(\mathbf{G}_j) = \delta(\mathbf{g}_j(\vec{\mathbf{x}}^j)) = \min(\{\delta(\mathbf{x}_{i'}) : \mathbf{x}_{i'} \in \vec{\mathbf{x}}^j\} \cup \{\omega\}) \leq \delta(\mathbf{x}_i) < \rho(h,j)$, contradicting that $\vec{\mathbf{G}}$ is adequate for h. The adequacy of $\vec{\mathbf{G}}$ also implies $\vec{\mathbf{G}} \in \mathbf{K}$, which implies that for each j, $\vec{\mathbf{x}}^j$ is adequate for g_j. Therefore, $\delta(\mathbf{x}_i) \geq \rho(g_j,k)$ for all j and k such that $x_i \equiv (\vec{x}^j)_k$. One also has $\delta(\mathbf{x}_i) \geq 0$ for all i, because $\mathbf{x}_i \in \mathbf{K}$. Using the definition of $\rho(f,i)$, these three statements about $\delta(\mathbf{x}_i)$ imply that $\mathbf{x}_i$ is adequate for f. Therefore, $\mathbf{F} \in \mathbf{K}$ and by the definition of M, $\mathbf{F} = [\underline{\mathbf{f}}(\underline{\vec{\mathbf{x}}})]^{\delta(\mathbf{F})}$. Since $\vec{\mathbf{G}}$ is adequate for h and $\vec{\mathbf{x}}^j$ is adequate for g_j $(1 \leq j \leq m)$, the interpretation of $h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m)))$ is $[\underline{\mathbf{h}}(\underline{\mathbf{g}}_1(\underline{\vec{\mathbf{x}}}^1), \ldots, \underline{\mathbf{g}}_m(\underline{\vec{\mathbf{x}}}^m))]^d$ for some d. By the definition of M this is equal to $[\underline{\mathbf{f}}(\underline{\vec{\mathbf{x}}})]^{\delta(\mathbf{F})}$, completing the proof that $M, A|^{\vec{\mathbf{x}}}_{\vec{x}} \models f(\vec{x}) = h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$.

Finally, the relativized form of (rec.2) is:

$$((\forall \vec{y} \in U^{q(\ulcorner \vec{y} \urcorner)} \cap K,\ Kg(\vec{y})) \wedge (\forall x, \vec{y}, z \in U^{q(\ulcorner x \urcorner), q(\ulcorner \vec{y} \urcorner), q(\ulcorner z \urcorner)} \cap K,\ Kh(x, \vec{y}, z)))$$
$$\supset \forall x, \vec{y} \in U^{q(\ulcorner x \urcorner), q(\ulcorner \vec{y} \urcorner)} \cap K,\ (x \neq 0_j \supset (Kf(px, \vec{y}) \supset f(x, \vec{y}) = h(x, \vec{y}, f(px, \vec{y})))).$$

The argument is summarized as follows. The convergence statement $Kf(px, \vec{y})$ implies that $\langle \mathbf{px}, \vec{\mathbf{y}} \rangle$ is adequate for f. Since $\delta(\mathbf{x}) = \delta(\mathbf{px})$, this also means that $\langle \mathbf{x}, \vec{\mathbf{y}} \rangle$ is adequate for f. It will follow using the definition of ρ that $\langle \mathbf{x}, \vec{\mathbf{y}}, \mathbf{f}(\mathbf{px}, \vec{\mathbf{y}}) \rangle$ is adequate for h. As in the case of (comp), convergence of the two sides implies that the equality holds. The antecedent clauses actually are not required to show that $(\text{rec.2})^q$ is satisfied.

- Letting A be any assignment, assume the antecedent of $(\text{rec.2})^q$ is satisfied, and choose any $\mathbf{x}, \vec{\mathbf{y}} \in \mathbf{U}^{q(\ulcorner x \urcorner), q(\ulcorner \vec{y} \urcorner)} \cap \mathbf{K}$ such that $\mathbf{x} \neq \mathbf{0}_j$. One must show that $M, A|^{\mathbf{x}, \vec{\mathbf{y}}}_{x, \vec{y}} \models (Kf(px, \vec{y}) \supset f(x, \vec{y}) = h(x, \vec{y}, f(px, \vec{y}))$. Recall that $\rho(f, 1) = \max\{\rho(h, 1),\ 1 + \rho(h, m_h)\}$, and $\rho(f, i) = \max\{\rho(h, i),\ \rho(g, i-1),\ \rho(h, m_h)\}$ for $2 \leq i \leq m_f$.

  Assume that $M, A|^{\mathbf{x}, \vec{\mathbf{y}}}_{x, \vec{y}} \models Kf(px, \vec{y})$. Then $\langle \mathbf{px}, \vec{\mathbf{y}} \rangle$ is adequate for f, because functions in M only converge on adequate inputs. Therefore $\delta(\mathbf{f}(\mathbf{px}, \vec{\mathbf{y}})) = \min(\{\delta(\mathbf{px})\} \cup \{\delta(\mathbf{y}_i) : 1 \leq i < m_f - 1\}) \geq \min\{\rho(f, i) : 1 \leq i \leq m_f\} \geq \rho(h, m_f)$ by the adequacy of $\langle \mathbf{px}, \vec{\mathbf{y}} \rangle$ and the definition of $\rho(f)$. Also one has $\delta(\mathbf{x}) = \delta(\mathbf{px}) \geq \rho(f, 1) \geq \rho(h, 1)$ and $\delta(\mathbf{y}_i) \geq \rho(f, i+1) \geq \rho(h, i+1)$ for $1 \leq i \leq m_f - 1$, again using the adequacy of $\langle \mathbf{px}, \vec{\mathbf{y}} \rangle$ and the definition of $\rho(f)$. It follows that $\langle \mathbf{x}, \vec{\mathbf{y}}, \mathbf{f}(\mathbf{px}, \vec{\mathbf{y}}) \rangle$ is adequate for h. Easily, $\langle \mathbf{x}, \vec{\mathbf{y}} \rangle$ is adequate for f because $\langle \mathbf{px}, \vec{\mathbf{y}} \rangle$ is adequate for f and because $\delta(\mathbf{x}) = \delta(\mathbf{px})$. In the variant model M, under the definition of f for $\mathbf{x} \neq \mathbf{0}_j$ with f derived in $\mathcal{PR}_2$ by recursion from g and h, the adequacy of the inputs on each side of the equation implies that $f(x, \vec{y}) = h(x, \vec{y}, f(px, \vec{y}))$ holds.

□

Now we have reached the more interesting cases, involving edges of $G(\Pi)$.

THEOREM 6.2 (Implicit Ramification). *Let Π be an $\mathcal{SA}_2$ proof with conclusion β, let $q \geq G(\Pi)$, and let $M \geq q$. Then $M \models \beta^q$.*

PROOF. The proof is by structural induction on Π, considering each rule or axiom by which the conclusion of Π could have been reached. Applying the preceding lemma, what remains is (FO.5), (rec.1), (MP) and (ind). Let A be any assignment.

Consider (FO.5): $\forall \vec{z}, (\forall x, \alpha \supset \alpha[t/x])$. The relativization has the form

$$\gamma \supset \forall \vec{z} \in U^{q(\ulcorner \vec{z} \urcorner)},\ ((\forall x \in U^{q(\ulcorner x \urcorner)},\ \alpha') \supset \alpha'[t/x]),$$

where γ consists of the antecedents relativizing any free variables. To show that (M, A) satisfies the relativized statement, assume $M, A \models \gamma$, let $\vec{\mathbf{z}} \in \mathbf{U}^{q(\ulcorner \vec{z} \urcorner)}$, and assume $M, A|^{\vec{\mathbf{z}}}_{\vec{z}} \models \forall x \in U^{q(\ulcorner x \urcorner)},\ \alpha'$. One must show that $M, A|^{\vec{\mathbf{z}}}_{\vec{z}} \models \alpha'[t/x]$. Because of the edge $\langle \ulcorner x \urcorner, \ulcorner y \urcorner, 0 \rangle$ added in (G.2), one has $q(\ulcorner x \urcorner) \leq q(\ulcorner y \urcorner)$ for all variables $\ulcorner y \urcorner \in t$ (it is trivial if $\ulcorner y \urcorner = \ulcorner x \urcorner$). Similarly, the edge $\langle \ulcorner x \urcorner, \ulcorner 0_i \urcorner, 0 \rangle$ ensures $q(\ulcorner x \urcorner) \leq q(\ulcorner 0_i \urcorner)$ for all $0_i \in t$. Let $j = q(\ulcorner x \urcorner)$; one has $\mathbf{U}^j \supseteq \mathbf{U}^{q(\ulcorner y \urcorner)}$ and $\mathbf{U}^j \supseteq \mathbf{U}^{q(\ulcorner 0_i \urcorner)}$. Each variable $y \in t$ is either free or is among $\vec{z}$; in either case one has $\mathbf{y} \in \mathbf{U}^j$ because $(M, A|^{\vec{\mathbf{z}}}_{\vec{z}})$ satisfies a relativizing clause $U^{q(\ulcorner y \urcorner)} y$ that implies $U^j y$. Similarly, each $0_i \in t$ is interpreted by a shadow $\mathbf{0}_i \in \mathbf{U}^j$ because $M \geq q$ and $\mathbf{U}^j \supseteq \mathbf{U}^{q(\ulcorner 0_i \urcorner)}$. Now the closure lemma for $\mathbf{U}^j$ gives $\mathbf{t} \in \mathbf{U}^j$. Since $M, A|^{\vec{\mathbf{z}}}_{\vec{z}} \models \forall x \in U^j,\ \alpha'$, one has $M, A|^{\vec{\mathbf{z}}}_{\vec{z}} \models \alpha'[t/x]$ as required.

The relativized form of (rec.1) is:

$$((\forall \vec{y} \in U^{q(\ulcorner \vec{y} \urcorner)} \cap K,\ Kg(\vec{y})) \wedge (\forall x, \vec{y}, z \in U^{q(\ulcorner x \urcorner), q(\ulcorner \vec{y} \urcorner), q(\ulcorner z \urcorner)} \cap K,\ Kh(x, \vec{y}, z)))$$
$$\supset (\forall \vec{y} \in U^{q(\ulcorner \vec{y} \urcorner)} \cap K,\ (f(0_i, \vec{y}) = g(\vec{y})))$$

The argument can be summarized as follows. The relativized antecedents assert that points at the depths given by q, are adequate for g and h. This, together with the (G.3) edges and the definition of $\rho(f)$, ensures that appropriate vectors $\langle 0_i, \vec{\mathbf{y}} \rangle$ are adequate for f. Using the definition of M_0, the adequacy of inputs on both sides of the equality "$f(0_i, \vec{y}) = g(\vec{y})$" ensures that the equality holds, because f is defined in $\mathcal{PR}_2$ by a recursion having g as the base function.

- Assume the antecedent of $(\text{rec.1})^q$ is satisfied, and choose any $\vec{\mathbf{y}} \in \mathbf{U}^{q(\ulcorner \vec{y} \urcorner)} \cap \mathbf{K}$; one must show $M, A|^{\vec{\mathbf{y}}}_{\vec{y}} \models f(0_i, \vec{y}) = g(\vec{y})$. In M, functions only converge on adequate inputs; therefore, satisfaction of $(\forall \vec{y} \in U^{q(\ulcorner \vec{y} \urcorner)} \cap K,\ Kg(\vec{y}))$ (under any assignment) implies that $q(\ulcorner y_j \urcorner) \geq \rho(g, j)$ for $1 \leq j \leq m_g$. Similarly, the satisfaction of $(\forall x, \vec{y}, z \in U^{q(\ulcorner x \urcorner), q(\ulcorner \vec{y} \urcorner), q(\ulcorner z \urcorner)} \cap K,\ Kh(x, \vec{y}, z)))$ implies that $q(\ulcorner x \urcorner) \geq \rho(h, 1)$, and $q(\ulcorner y_j \urcorner) \geq \rho(h, j+1)$ for $1 \leq j < m_h - 2$, and $q(\ulcorner z \urcorner) \geq \rho(h, m_h)$.

  Now, the edges added in (G.3) for the (rec.1) axiom are

  $$\{\langle \ulcorner x \urcorner, \ulcorner 0_i \urcorner, 0 \rangle\} \cup \{\langle \ulcorner z \urcorner, \ulcorner 0_i \urcorner, 1 \rangle\} \cup \{\langle \ulcorner z \urcorner, \ulcorner y_j \urcorner, 0 \rangle : 1 \leq j \in m_f - 1\}$$

  Since $q \geq G(\Pi)$, these give $q(\ulcorner 0_i \urcorner) \geq q(\ulcorner x \urcorner) \geq \rho(h, 1)$ and, critically, $q(\ulcorner 0_i \urcorner) \geq 1 + q(\ulcorner z \urcorner) \geq 1 + \rho(h, m_h)$. Under the definition of $\rho(f, 1) = \max\{\rho(h, 1), 1 + \rho(h, m_h)\}$ it follows that $q(\ulcorner 0_i \urcorner) \geq \rho(f, 1)$. We already have that $q(\ulcorner y_j \urcorner) \geq \max\{\rho(g, j), \rho(h, j+1)\}$ from the antecedents, and using the third set of edges one also has $q(\ulcorner y_j \urcorner) \geq q(\ulcorner z \urcorner) \geq \rho(h, m_h)$, for $1 \leq j \leq m_f - 1$. Since $\rho(f, j+1) = \max\{\rho(h, j+1), \rho(h, m_h), \rho(g, j)\}$ by definition, one has $q(\ulcorner y_j \urcorner) \geq \rho(f, j+1)$ for $1 \leq j \leq m_f - 1$. In summary, therefore, the conditions $M \geq q$ and $\vec{\mathbf{y}} \in \mathbf{U}^{q(\ulcorner \vec{y} \urcorner)} \cap \mathbf{K}$ (together with the antecedents) ensure that $\langle 0_i, \vec{\mathbf{y}} \rangle$ is adequate for f. Similarly, $\vec{\mathbf{y}}$ is adequate for g. Therefore, under the definition of M, one has $f(0_i, \vec{y}) = [\underline{\mathbf{f}}(\underline{\mathbf{0}_i}, \underline{\vec{\mathbf{y}}})]^{\delta(\mathbf{f}(\mathbf{0}_i, \vec{\mathbf{y}}))} = [\underline{\mathbf{g}}(\underline{\vec{\mathbf{y}}})]^{\delta(\mathbf{g}(\vec{\mathbf{y}}))} = g(\vec{y})$ as required.

For (MP), let Π_1 be a proof of α, and let Π_2 be a proof of $\alpha \supset \beta$. The relevant clause is (G.4): $G(\Pi) = G(\Pi_1) \cup G(\Pi_2)$. Since q is good for $G(\Pi)$, q is also good for the subgraphs: $q \geq G(\Pi_1)$ and $q \geq G(\Pi_2)$. Applying the induction hypothesis on Π_1 and Π_2 with M, one has $M, A \models \alpha^q$ and $M, A \models (\alpha \supset \beta)^q$. By standard first-order semantic reasoning, $M, A \models \beta^q$.

Finally we have arrived at (ind). Let Π_1 and Π_2 be the subproofs for the premisses of an induction with conclusion $\forall z \in K,\ \alpha[z/y]$. Recall the (G.5) edges:

$$G(\Pi) = G(\Pi_1) \cup G(\Pi_2) \cup \{\langle \ulcorner 0_i \urcorner, \ulcorner z \urcorner, 0 \rangle\} \cup \{\langle \ulcorner x \urcorner, \ulcorner z \urcorner, 0 \rangle\}.$$

Since $G(\Pi) \supseteq G(\Pi_1) \cup G(\Pi_2)$, the assumption $q \geq G(\Pi)$ gives $q \geq G(\Pi_1)$ and $q \geq G(\Pi_2)$. Therefore the induction hypothesis can be applied to Π_1 and Π_2. Letting γ be the relativizing antecedents for the free variables of α (other than y), and letting α' be obtained from α by relativizing the quantifiers only, the induction hypothesis implies that for all $M' \geq q$,

(1) $M', A \models \gamma \supset \alpha'[0_i/y]$
(2) $M', A \models \gamma \supset \forall x \in U^{q(\ulcorner x \urcorner)} \cap K,\ (x \neq 0_j \supset (\alpha'[px/y] \supset \alpha'[x/y]))$.

Note that γ does not contain x or z since these are not free in α. Let $d = q(\ulcorner z \urcorner)$. It is sufficient to show that, for the original M given in the statement of the theorem,

$$(3) \qquad M, A \models \gamma \supset \forall z \in U^d \cap K,\ \alpha'[z/y].$$

Since $q \geq G(\Pi)$, the existence of the edges $\langle \ulcorner 0_i \urcorner, \ulcorner z \urcorner, 0\rangle$ and $\langle \ulcorner x \urcorner, \ulcorner z \urcorner, 0\rangle$ implies $d = q(\ulcorner z \urcorner) \geq q(\ulcorner 0_i \urcorner)$ and similarly $d \geq q(\ulcorner x \urcorner)$.

It is proved by an induction on $\underline{\mathbf{z}}$ that $M, A \models (\gamma \wedge U^d z \wedge Kz) \supset \alpha'[z/y]$. This implies (3) as required. (Recall that $\underline{\mathbf{z}}$ is the underlying integer for the element $\mathbf{z}$ that A assigns for z. We can assume $\mathbf{z} \in \mathbb{K}$ because of the antecedent Kz.)

The induction is summarized as follows. When $\mathbf{z}$ is a shadow of zero of depth at least d, let M' be M but with 0_i interpreted as $[\underline{0}]^{\delta(\mathbf{z})}$. By (1), $\gamma \supset \alpha'[0_i/y]$ holds in M'. Since $\mathbf{z}$ is $\mathbf{0}_i$, this means that $\gamma \supset \alpha'[z/y]$ holds in M'. Since M and M' only differ on 0_i, which doesn't appear in $\alpha'[z/y]$, this means that M satisfies $\gamma \supset \alpha'[z/y]$. The base case of the induction on $\mathbf{z}$ follows. The step case follows by a slightly simpler argument using (2) with $M' \equiv M$.

- In the base case of this induction, $\underline{\mathbf{z}}$ is zero. If $\delta(\mathbf{z}) < d$ then $(U^d z \wedge Kz)$ is falsified and the entire formula is satisfied. Otherwise, $\mathbf{z}$ is a shadow of zero of depth at least $d \geq q(\ulcorner 0_i \urcorner)$. Consider the variant model M' obtained from M by setting the interpretation of 0_i equal to $[\underline{0}]^{\delta(\mathbf{z})}$. Since $M \geq q$ and $\delta(\mathbf{z}) \geq q(\ulcorner 0_i \urcorner)$, one has $M' \geq q$, enabling us to use the induction hypothesis (1). Since $\mathbf{z}$ is identically $\mathbf{0}_i$ under (M', A), one has $M', A \models (\gamma \wedge U^d z \wedge Kz) \supset \alpha'[z/y]$ if and only if $M', A \models (\gamma \wedge U^d 0_i \wedge K0_i) \supset \alpha'[0_i/y]$. The latter statement is implied by (1). Now we must replace M' with M again. By the definition of (ind), 0_i does not appear in α (nor in α', γ, $U^d z$ or Kz). But M and M' differ only in the interpretation of 0_i. Therefore $M, A \models (\gamma \wedge U^d z \wedge Kz) \supset \alpha'[z/y]$ if and only if $M', A \models (\gamma \wedge U^d z \wedge Kz) \supset \alpha'[z/y]$. The satisfaction by (M', A) has already been proved, finishing the base case of the induction on $\underline{\mathbf{z}}$.
- In the induction step case, the underlying integer $\underline{\mathbf{z}}$ is greater than zero. Again the case of interest is when $\delta(\mathbf{z}) \geq q(\ulcorner z \urcorner)$. Observe that $\underline{\mathbf{pz}} < \underline{\mathbf{z}}$. The subsidiary induction hypothesis is $M, A|^{\mathbf{pz}}_z \models (\gamma \wedge U^d z \wedge Kz) \supset \alpha'[z/y]$. By the definition of the (ind) rule, z is not free in α (or γ). So, the free occurrences of z are all displayed when we write "$(\gamma \wedge U^d z \wedge Kz) \supset \alpha'[z/y]$". Therefore, the subsidiary induction hypothesis is equivalent to $M, A \models (\gamma \wedge U^d pz \wedge Kpz) \supset \alpha'[pz/y]$. Since $\delta(\mathbf{x}) = \delta(\mathbf{px})$, this is equivalent to

  $$(4)\ M, A \models (\gamma \wedge U^d z \wedge Kz) \supset \alpha'[pz/y].$$

  At the same time, we have the main induction hypothesis (2) with $M' \equiv M$. Assume $M, A \models (\gamma \wedge U^d z \wedge Kz)$. By (4), $M, A \models \alpha'[pz/y]$. Then using (2) with $M' \equiv M$, one has $M, A \models \alpha'[z/y]$ (the condition $\mathbf{z} \neq \mathbf{0}_j$ is satisfied because $\underline{\mathbf{z}} > 0$). Therefore, $M, A \models (\gamma \wedge U^d z \wedge Kz) \supset \alpha'[z/y]$ as required to complete the subsidiary induction on $\underline{\mathbf{z}}$.

□

7. Correspondence of $\mathcal{SA}^r_2$ and $\mathcal{PR}^r_2$

One direction of the main result can now be completed. Recall that the convergence statement for f is $\forall \vec{x} \in K,\ Kf(\vec{x})$.

THEOREM 7.1. *Let $f \in \mathcal{PR}_2$ and let $r < \omega$. If $\mathcal{SA}^r_2$ proves the convergence of f, then $f \in \mathcal{PR}^r_2$.*

PROOF. Let $\Pi \in \mathcal{SA}^r_2$ be a proof of $\forall \vec{x} \in K,\ Kf(\vec{x})$. Recall that $\rho(\Pi, \cdot) \geq G(\Pi)$ by definition. Also, $M_0 \geq \rho(\Pi, \cdot)$. Using the preceding theorem (and the fact that K is equivalent to its own relativization), $M_0 \models \forall \vec{x} \in U^{\rho(\Pi, \ulcorner \vec{x} \urcorner)} \cap K,\ Kf(\vec{x})$. Choose any $\vec{\mathbf{x}}$ such that $\delta(\mathbf{x}_i) = \rho(\Pi, \ulcorner x_i \urcorner)$ for $1 \leq i \leq m_f$. Then $\vec{\mathbf{x}} \in \mathbb{U}^{\rho(\Pi, \ulcorner \vec{x} \urcorner)} \cap \mathbb{K}$. By the relativized convergence statement, $\mathbf{f}(\vec{\mathbf{x}}) \in \mathbb{K}$. But by the construction of M_0, if $\vec{\mathbf{x}}$ were not adequate for f then $\mathbf{f}(\vec{\mathbf{x}})$ would diverge (i.e. would not be in $\mathbb{K}$). Therefore, $\vec{\mathbf{x}}$ must be adequate for f, implying $\rho(\Pi, \ulcorner x_i \urcorner) = \delta(\mathbf{x}_i) \geq \rho(f, i)$ for $1 \leq i \leq m_f$. This implies $\rho(\Pi) \geq \rho(f)$, so $f \in \mathcal{PR}_2^{\rho(\Pi)}$. Here, the expression "$\mathcal{PR}_2^{\rho(\Pi)}$" is defined because $\rho(\Pi) \leq r < \omega$ by assumption. □

The other direction of the main result is not too difficult. A fact will pave the way.

Fact. Any first-order proof that uses substitution (FO.5) only to replace a quantified variable $\forall x$, with a free instance of the same variable x, has $G(\Pi) = \emptyset$. Specifically, the following derived rules leave the constraint graph unchanged: (UE) if $\vdash \forall x,\ \alpha$ then $\vdash \alpha$; and (UI) if $\vdash \alpha$ then $\vdash \forall x,\ \alpha$.

THEOREM 7.2. *Let $f \in \mathcal{PR}_2$, and let $r < \omega$. If $f \in \mathcal{PR}^r_2$, then there is an $\mathcal{SA}^r_2$ proof of the convergence of f.*

PROOF. In fact, there is a proof Π of $\forall \vec{x} \in K,\ Kf(\vec{x})$ such that $\rho(\Pi, \ulcorner x_i \urcorner) = \rho(f, i)$ and such that $G(\Pi)$ has no edge starting at $\ulcorner x_i \urcorner$, for $1 \leq i \leq m_f$. It is shown by induction on the structure of the derivation of f in $\mathcal{PR}_2$.

If f is one of the initial functions, then the convergence statement for f is proved using first-order logic from the defining axioms for f. For example, to prove convergence of c, one uses (p.4) together with (c.1-2). Other than (UI) and (UE), the only quantifier manipulation is of the form $t_1 = t_2 \supset Kt_1$ or $t_1 \neq t_2 \supset Kt_1$. It follows that the only edges in the constraint graph are weight 0 edges originating at the quantified variable in an occurrence of Kt (for some term t).

If f is defined by the composition $h(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$, then let Π_0 and $\Pi_1, \ldots, \Pi_m$ be the proofs of convergence obtained by the induction hypothesis on h and $g_1, \ldots, g_m$. Renaming the variables in Π_0, say Π_0 is a proof of $\forall \vec{y} \in K,\ Kh(\vec{y})$. Eliminating each $\forall y_j$ with $g_j(\vec{x}^j)$, using (UE) on each $\forall x_i$ in each Π_j, $j \geq 1$, and then making propositional moves followed by (UI), one obtains $\forall \vec{x} \in K,\ Kh(g_1(\vec{x}^1), \ldots, g_m(\vec{x}^m))$. In performing the replacement of y_j with $g_j(\vec{x}^j)$, one creates an edge of weight zero from $\ulcorner y_j \urcorner$ to $\ulcorner x_i \urcorner$ for each $x_i \in \vec{x}^j$. Essentially this forces $\rho(\Pi, x_i) \geq \rho(\Pi_0, y_j)$. The edges of $G(\Pi_j)$ $(1 \leq j \leq m)$ remain in the proof, as do the edges of $G(\Pi_0)$. For this construction, $\rho(\Pi, x_i) = \max\{\rho(\Pi_0, y_j) : x_i \in \vec{x}^j\} \cup \{\rho(\Pi_j, (\vec{x}^j)_k) : x_i \equiv (\vec{x}^j)_k\} \cup \{0\}$. Since $\rho(\Pi_0, y_j) = \rho(h, j)$ and $\rho(\Pi_j, (\vec{x}^j)_k) = \rho(g_j, k)$ by the induction hypothesis, under the definition of $\rho(f, i)$ this gives $\rho(\Pi, x_i) = \rho(f, i)$.

If f is defined by recursion from step function h and base function g, then let Π_h and Π_g be obtained by the induction hypothesis on h and g. Rename the variables as shown in the following outline sketch of Π. First-order deductions have been abbreviated here.

First,

$$\dfrac{\dfrac{\dfrac{\Pi_g}{\forall \vec{y} \in K,\ Kg(\vec{y})} \quad \dfrac{\Pi_h}{\forall x, \vec{y}, z \in K,\ Kh(x, \vec{y}, z)}}{\forall \vec{y} \in K,\ (f(0_i, \vec{y}) = g(\vec{y}))}\ \text{(rec.1) for } f}{\forall \vec{y} \in K,\ Kf(0_i, \vec{y})}$$

Second,

$$\dfrac{\dfrac{\dfrac{\Pi_g}{\forall \vec{y} \in K,\ Kg(\vec{y})} \quad \dfrac{\Pi_h}{\forall x, \vec{y}, z \in K,\ Kh(x, \vec{y}, z)}\ \text{(rec.2) for } f}{\forall x, \vec{y} \in K,\ (x \neq 0_j \supset (Kf(px, \vec{y}) \supset f(x, \vec{y}) = h(x, \vec{y}, f(px, \vec{y})))}}{\forall x, \vec{y} \in K,\ x \neq 0_j \supset (Kf(px, \vec{y}) \supset Kf(x, \vec{y}))}}{\forall x \in K,\ (x \neq 0_j \supset (\forall \vec{y} \in K,\ Kf(px, \vec{y}) \supset \forall \vec{y} \in K,\ Kf(x, \vec{y})))}$$

These two pieces are joined together by an (ind) instance,

$$\dfrac{\forall \vec{y} \in K,\ Kf(0_i, \vec{y}) \qquad \forall x \in K,\ (x \neq 0_j \supset (\forall \vec{y} \in K,\ Kf(px, \vec{y}) \supset \forall \vec{y} \in K,\ Kf(x, \vec{y})))}{\forall u \in K,\ \forall \vec{y} \in K,\ Kf(u, \vec{y})}$$

Examining this outline, one sees that a path of weight one is created from the critical variable $\ulcorner z \urcorner$ to the induction variable $\ulcorner u \urcorner$ through the intermediary $\ulcorner 0_i \urcorner$, due to the (ind) following the (rec.1). A path of weight zero is similarly created from the recursion variable $\ulcorner x \urcorner$ to $\ulcorner 0_i \urcorner$ and from there to the induction variable $\ulcorner u \urcorner$. For the other variables $\vec{y}$, the instances of (rec.1) for f result in edges of weight 0 from $\ulcorner z \urcorner$ to $\ulcorner y_i \urcorner$, for $1 \leq i \leq m_f - 1$.

These are the only significant new paths created in the constraint graph. Although some paths are created due to the implicit use of $f(x, \vec{y}) = h(x, \vec{y}, f(px, \vec{y})) \supset Kf(x, \vec{y})$, these have weight 0 and terminate at the quantified variable in $Kf(x, \vec{y})$. That variable does not appear in Π_h or Π_g, hence these paths contribute zero to the rank of the overall proof.

By the induction hypothesis, $\rho(\Pi_h, \ulcorner x \urcorner) = \rho(h, 1)$ and $\rho(\Pi_h, \ulcorner z \urcorner) = \rho(h, m_h)$. Due to the new paths ending at $\ulcorner u \urcorner$, and the fact that $\ulcorner z \urcorner$ and $\ulcorner x \urcorner$ do not appear in Π_g, it follows that $\rho(\Pi, \ulcorner u \urcorner) = \max\{1+\rho(\Pi_h, \ulcorner z \urcorner), \rho(\Pi_h, \ulcorner x \urcorner)\} = max\{1+\rho(h, m_h), \rho(h, 1)\} = \rho(f, 1)$.

Due to the paths from $\ulcorner z \urcorner$ to $\vec{y}$, one has $\rho(\Pi, \ulcorner y_i \urcorner) = \max\{\rho(\Pi_h, \ulcorner z \urcorner), \rho(\Pi_h, \ulcorner y_{i-1} \urcorner), \rho(\Pi_g, \ulcorner y_{i-1} \urcorner)\}$ for $2 \leq i \leq m_f$. By the induction hypothesis on Π_g and Π_h one has $\rho(\Pi_g, \ulcorner y_{i-1} \urcorner) = \rho(g, i-1)$ and $\rho(\Pi_h, \ulcorner y_{i-1} \urcorner) = \rho(h, i)$ $(2 \leq i \leq m_f)$ and $\rho(\Pi_h, \ulcorner z \urcorner) = \rho(h, m_h)$. Therefore $\rho(\Pi, \ulcorner y_{i-1} \urcorner) = \max\{\rho(g, i-1), \rho(h, i), \rho(h, m_h)\} = \rho(f, i)$ for $2 \leq i \leq m_f$. □

Altogether, we have obtained the following.

THEOREM 7.3. *Let $f \in \mathcal{PR}_2$ and let $r < \omega$. Then $\mathcal{SA}_2^r$ proves the convergence of f, if and only if $f \in \mathcal{PR}_2^r$.*

COROLLARY 7.4. *A function is polynomial-time computable if and only if it is $\mathcal{SA}_2^1$-provably convergent. A function is Kalmar elementary if and only if it is $\mathcal{SA}_2^2$-provably convergent. For $r \geq 2$, a function is in $\mathcal{E}^{r+1}$ if and only if it is $\mathcal{SA}_2^r$-provably convergent.*

8. Ordinary Induction and Recursion

There is nothing special about the base 2 (induction-on-notation) formulation of the above results.

One can just as well define $\mathcal{PR}_1$ instead of $\mathcal{PR}_2$, by replacing s_0, s_1, p, and c with the initial functions $S(x) = x + 1$ and $P(x) = x - 1$ and $C(x, y, z) =$ "if $x = 0$ then y else z". The rule of "full recursion on notation" now becomes "full primitive recursion", replacing p with P. The definition of $\rho(f)$ is carried over unchanged, except that the set of initial functions is different.

In [4] it is shown that $\mathcal{PR}_1^1$ derivations define exactly the linear space computable functions, and for $r \geq 2$, $\mathcal{PR}_1^2$ derivations define exactly the functions in $\mathcal{E}^{r+1}$, the Grzegorczyk functions of level $r + 1$.

For a corresponding logic, one can define $\mathcal{SQ}_1$ and $\mathcal{SA}_1$ exactly as $\mathcal{SQ}_2$ and $\mathcal{SA}_2$ were defined, but using a signature based on $\mathcal{PR}_1$. One changes p to P in (ind) and (rec.1-2), and replaces the initial axioms for s_0, s_1, p, and c with new axioms corresponding to the initial functions of $\mathcal{PR}_1$. The proofs are carried over literally by reading p as P — except for the proof that M satisfies the relativized axioms of the initial functions, which can be directly adapted to suit the new initial functions.

THEOREM 8.1. *Let $f \in \mathcal{PR}_1$, and let $r < \omega$. Then $\mathcal{SA}_1^r$ proves the convergence of f, if and only if $f \in \mathcal{PR}_1^r$.*

COROLLARY 8.2. *A function is linear-space computable if and only if it is $\mathcal{SA}_1^1$-provably convergent. A function is Kalmar elementary if and only if it is $\mathcal{SA}_1^2$-provably convergent. For $r \geq 2$, a function is $\mathcal{SA}_1^r$-provably convergent (using a $\mathcal{PR}_1$ derivation) if and only if it is $\mathcal{SA}_2^r$-provably convergent (using a $\mathcal{PR}_2$ derivation); also if and only if it is in $\mathcal{E}^{r+1}$ (the Grzegorczyk class of level $r + 1$).*

9. Conclusion

The current work provides a contrast to earlier weak subsystems of arithmetic such as those of Buss [6] (for an overview see Krajicek [10]). The system $\mathcal{SA}_2^1$ is equivalent to Buss's S_2^1 in the sense that they both prove convergence of exactly the polytime computable functions. It seems quite remarkable that this definition of polynomial time provability admits unrestricted quantifier complexity in the induction formula. The correspondence of $\mathcal{PR}_j^i$ and $\mathcal{SA}_j^i$ $(j \in \{1, 2\}, 0 \leq i < \omega)$ suggests that a very strong relationship holds between recursion and induction despite the presence of complicating logical elements such as implication and quantification.

It is interesting that ramification levels, as identified in this work by the relativizations $\mathbf{U}^j$, can be inferred from the syntactic structure of a proof by induction. They do not need to be treated explicitly, either in the logical syntax via special unary predicate symbols or in the semantics via successive worlds with nested domains. Instead, they emerge naturally from an examination of the ramification that seems to be implicit in ordinary induction.

Only symbols for the primitive recursive functions were included in $\mathcal{SA}_2$. If one were to admit general recursion or higher-type primitive recursion, then one could define e.g. a function with exponential output length, using only a single recursion on notation. Correspondingly, such a function could be proved convergent using a single induction (see Leivant [13] lemma 3.7). It would be interesting to see whether a polytime logic is obtained when one restricts Leivant's "intrinsic theory" over W_0 and W_1 by requiring that the recursive equations be primitive recursive, rather than by requiring only existential formulas in induction rules [13].

A distinction has been made between 'zero' used as the base case of an induction and unrelated references to 'zero' in the induction formula. It is not clear whether this distinction is fundamental or can be eliminated.

A drawback to the definition of rank is that $\mathcal{SA}_2^r$ is not closed under modus ponens. This contrasts with the fact that $\mathcal{PR}_2^r$ is closed under composition. Although $\mathcal{SA}_2^r$ forms a proof "system" in a loose sense of the word, the approach has really been to view $\mathcal{SA}_2$ as a powerful given system and then to measure the complexity of all $\mathcal{SA}_2$ proofs. In practice, a feeling for the definition of rank should allow one to write down proofs with confidence that they will fall into a particular system such as $\mathcal{SA}_2^1$. An alternative viewpoint is that a system formalizing the idea of reasoning with limited resources *shouldn't* be closed under modus ponens [8].

Connections remain to be drawn between $\mathcal{SA}_2^r$, $\mathcal{SA}_1^r$, and earlier arithmetics. It should happen that the full systems $\mathcal{SA}_2$ and $\mathcal{SA}_1$ are both equivalent to Peano arithmetic, under a suitable translation of signatures and relativization to K. The issue is complicated by the presence of 0_i in the induction rule. Also, it would be nice to know more about $\mathcal{SA}_2^\omega$ and $\mathcal{SA}_1^\omega$. A method of relabeling of variables in Π such that $G(\Pi)$ is guaranteed to be acyclic, would show that each $\mathcal{SA}_2^\omega$ proof can be carried out in $\mathcal{SA}_2^r$ for some finite r. Finally, it would be interesting to carry out an analysis similar to the present one, but on a system having, like S_2^1, only a finite number of predefined function symbols.

There seems to be a conceptual relationship between large non-standard integers and shadows of $\mathbf{N}^\omega$ at low depth. For a recursively defined function $\mathbf{f}$, the countable descending chain $\ldots, \mathbf{f}(\mathbf{z}+2), \mathbf{f}(\mathbf{z}+1), \mathbf{f}(\mathbf{z}), \ldots$ is unfounded in a nonstandard model, where $\mathbf{z}$ is a nonstandard integer that corresponds to 0 in the natural way. In a ramified model where $\mathbf{z}$ is a shadow of 0, this chain has a last element $\mathbf{f}(\mathbf{z})$; but the sequence of values is also unfounded in the sense that $\mathbf{f}(\mathbf{z})$ is unconstrained by the axiom defining $\mathbf{f}$. In both cases, it is only induction that forces $\mathbf{f}$ to take on a 'correct' value at points in these chains. Under this analogy, the subuniverses $\mathbf{U}^0, \mathbf{U}^1, \mathbf{U}^2, \ldots$ are like smaller and smaller initial segments of a nonstandard model, with $\mathbf{U}^\omega$ being the standard integers. It would be quite interesting to find an exact relationship between nonstandard integers in nonstandard unramified models, and shadow integers in standard ramified models.

References

[1] M. Beeson, *Foundations of Constructive Mathematics*, Metamathematical Studies, Springer-Verlag (1985).

[2] S. Bellantoni, "Predicative Recursion and Computational Complexity", T.R. 264/92, Department of Computer Science, University of Toronto, September 1992.

[3] S. Bellantoni, "Predicative Recursion and the Polytime Hierarchy", in *Feasible Mathematics II*, P. Clote and J. B. Remmel eds.; Birkhauser, Boston, (1995) p. 15-29.

[4] S. Bellantoni and K.H. Niggl, "Ranking Primitive Recursions: The Low Grzegorczyck Classes Revisited", to appear.

[5] S. Bellantoni and S. Cook, "A New Recursion-Theoretic Characterization of the Polytime Functions", in *Computational Complexity* v. 2, p. 97-110, 1992.

[6] S. Buss, *Bounded Arithmetic*, Ph.D. Dissertation, Princeton University, 1985; reprinted Bibliopolis, Naples, 1986.

[7] P. Clote, "Computation Models and Function Algebras", in *Handbook of Recursion Theory*, Ed Griffor, ed., Elsevier, 1996.

[8] R. Fagin, J. Halpern, Y. Moses, M. Vardi, *Reasoning About Knowledge*, The MIT Press, Cambridge, MA. 1995.

[9] N. Goodman, "Epistemic Arithmetic is a Conservative Extension of Intuitionistic Arithmetic", in *The Journal of Symbolic Logic*, v. 49, n. 1, March 1984.

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

[11] K. Lambert, ed., *Philosophical Applications of Free Logic*, Oxford University Press, 1991.

[12] D. Leivant, ed., *Logic and Computational Complexity: International Workshop LCC '94, Indianapolis, IN, USA, October 1994, Selected Papers*, D. Leivant, ed.; Lecture Notes in Computer Science 960, Springer (1995).

[13] D. Leivant, "Intrinsic Theories and Computational Complexity", in [12], p 177-194.

[14] D. Leivant, "Ramified Recurrence and Computational Complexity I: Word recurrence and poly-time", in *Feasible Mathematics II*, P. Clote and J. B. Remmel eds.; Birkhauser, Boston, (1995) p. 321-343.

[15] D. Leivant, "Stratified Functional Programs and Computational Complexity" in *Conference Record of the Twentieth Annual ACM Symposium on Principles of Programming Languages, New York, 1993*, Association for Computing Machinery.

[16] A. Margaris, *First Order Mathematical Logic*, Blaisdell Publishing Co., 1967.

[17] E. Mendelson, *Introduction To Mathematical Logic*, Van Nostrand, New York, 1979.

[18] D. Prawitz, "Ideas and Results In Proof Theory", in *Proceedings of the Second Scandinavian Logic Symposium*, J. E. Fenstad, ed., Studies in Logic and the Foundations of Mathematics, v. 63, North Holland, 1971.

[19] S. Shapiro, ed., *Intensional Mathematics*, Studies in Logic V113, North-Holland, 1985.

[20] S. Shapiro, "Epistemic and Intuitionistic Arithmetic", in [19].

Department of Computer Science, University of Toronto, Toronto, Ontario, M5S 3G4
E-mail address: sjb@cs.toronto.edu

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

Lower Bounds on Nullstellensatz Proofs via Designs

Samuel R. Buss

ABSTRACT. The Nullstellensatz proof system is a proof system for propositional logic based on algebraic identities in fields. Prior work has proved lower bounds of the degree Nullstellensatz refutations using combinatorial constructions called designs. This paper surveys the use of designs for Nullstellensatz lower bounds. We give a new, more general definition of designs. We present an explicit construction of designs which give a linear lower bound on the degree of Nullstellensatz proofs of the housesitting principle. Our designs for the housesitting principle work over any ring.

1. Introduction

The Nullstellensatz proof system is a propositional proof system which establishes the truth of tautologies using reasoning about polynomials over a field, based on the Hilbert Nullstellensatz. The original definition of the Nullstellensatz proof system was by [2]. and many of the basic properties of Nullstellensatz proofs can be found in [3] and in the survey [6]. We begin with a review of the the Nullstellensatz.

In Nullstellensatz refutations, the Boolean values *True* and *False* are identified with the algebraic values 1 and 0 respectively. Boolean operations can be expressed as ring operations; e.g., $\neg x$ is the same as $1 - x$, a conjunction $x \wedge y$ is the same as the product xy, and a disjunction $x \vee y$ is the same as $x + y - xy$. In this way, one converts an arbitrary propositional formula $\varphi(\vec{x})$ into an algebraic term $t(\vec{x})$ such that $\varphi(\vec{x})$ and $t(\vec{x})$ have the same value for all assignments of 0/1 values to the variables $\vec{x}$. We shall always fix some underlying ring R and assume that all algebraic operations occur in that ring. Note that the translation from propositional logic to algebraic

1991 *Mathematics Subject Classification.* Primary 03F20, 03B05, 03G99, 68R05.
Key words and phrases. propositional proofs, nullstellensatz, design, housesitting principle.
Supported in part by NSF grant DMS-9503247 and US-Czech Science and Technology grant 93-025.

expressions works over an arbitrary ring. However, in most interesting cases, the ring is actually a field.

The central difficult computational problem of propositional logic is, given a propositional formula $\varphi(\vec{x})$ with $\vec{x} = x_1, \ldots, x_n$ the variables in φ, to determine if $\varphi(\vec{x})$ is satisfiable. If $t(\vec{x})$ is an algebraic expression which computes the same function as $\varphi(\vec{x})$ for all Boolean inputs, then φ is satisfiable if and only if $t(\vec{a}) = 1$ for some 0/1 assignment of values $\vec{a}$ to the variables $\vec{x}$. If our algebraic structure R is a field, then the formula $x_i^2 - x_i = 0$ is satisfied exactly when x_i is assigned a Boolean value 0 or 1. Therefore φ is unsatisfiable if and only if it is not possible to assign field elements to the variables $\vec{x} = x_1, \ldots, x_n$ which simultaneously satisfy the $n+1$ equations

$$\begin{aligned} t(\vec{x}) - 1 &= 0 \\ x_i^2 - x_i &= 0 \qquad 1 \leq i \leq n. \end{aligned}$$

If the algebraic structure R is only a ring, it is still true that φ is unsatisfiable if and only if the $n+1$ equations cannot be simultaneously satisfied in R. However, the statement is harder to prove for rings, since an equation $x^2 - x = 0$ may have many non-0/1 solutions in a ring. However it still holds for rings, since an assignment which satisfies the equations will also satisfy them in the field R/M for M a maximal ideal of R.

It is often convenient to work with terms t that are not obtained from a propositional formula φ by the above canonical method. Further it is often convenient to work with multiple terms instead of a single term t. This leads to the following generalization of the propositional satisfiability problem: given polynomials $t_1, \ldots, t_m$ over a ring R, is there a assignment of 0/1 values to the variables in the terms that makes all the terms simultaneously equal to zero. If R is a field this is equivalent to asking whether the $m+n$ equations

$$\begin{aligned} t_i(\vec{x}) &= 0 \qquad \text{for } i = 1, 2, \ldots, m \\ x_j^2 - x_j &= 0 \qquad \text{for } j = 1, 2, \ldots, n \end{aligned}$$

can be simultaneously satisfied. It is an easy consequence of the Hilbert Nullstellensatz that this question is equivalent to asking whether there are polynomials $F_i(\vec{x})$ and $G_j(\vec{x})$ such that the polynomial identity

$$\begin{aligned} 1 = {} & F_1(\vec{x})t_1(\vec{x}) + F_2(\vec{x})t_2(\vec{x}) + \cdots + F_m(\vec{x})t_m(\vec{x}) + \\ & G_1(\vec{x})(x_1^2 - x_1) + G_2(\vec{x})(x_2^2 - x_2) + \cdots + G_m(\vec{x})(x_n^2 - x_n) \end{aligned} \tag{1}$$

holds in $R[\vec{x}]$.

Definition A *Nullstellensatz refutation* of $t_1, \ldots, t_m$ is a set of polynomials $F_1, \ldots, F_m$, $G_1, \ldots, G_n$ such that the polynomial identity (1) holds. The *degree* of the Nullstellensatz refutation is

$$\max\{deg(F_i) + deg(t_i) \,:\, 1 \leq i \leq m\}.$$

It can be shown that $\max_j\{deg(G_j)+2\} \leq \max_i\{deg(F_i t_i)\}$, and this justifies the fact that the definition of 'degree' does not depend on the degrees of the G_j's.

The purpose of a Nullstellensatz refutation is to prove that the terms $t_1, \ldots, t_m$ cannot be simultaneously satisfied by 0/1 values.

THEOREM 1. *(Soundness of Nullstellensatz) If $t_1, \ldots, t_n$ have a Nullstellensatz refutation, then there is no assignment of 0/1 values to variables that makes $t_1, \ldots, t_n$ simultaneously equal to zero.*

The soundness theorem is readily proven by noting that any assignment of 0/1 values that made each t_i equal to zero would also make each polynomial $x_i^2 - x_i$ equal to zero; but this would contradict the equality (1).

Conversely, the Nullstellensatz proof system is adequate (complete), provided the algebraic structure is a field:

THEOREM 2. *(Completeness of Nullstellensatz refutations). Let the algebraic structure R be a field. Suppose there is no assignment of 0/1 values to variables that makes $t_1, \ldots, t_m$ simultaneously zero. Then $t_1, \ldots, t_n$ have a Nullstellensatz refutation.*

PROOF. Simple proofs of the completeness theorem have been given by [3, 6]. Alternatively, the completeness theorem is a simple corollary of the Hilbert Nullstellensatz; namely, if $t_1 = 0, \ldots, t_m = 0$ have no 0/1 satisfying assignment, then there is no solution to these equations plus the n equations $x_i^2 - x_i = 0$, even in the algebraic closure of R. Therefore the Hilbert Nullstellensatz implies the existence of polynomials F_i and G_j over R that satisfy (1). □

When the algebraic structure R is not a field, then the completeness theorem does not always hold in general. However, there are some notable cases in which is does holds; for instance, when the polynomials $\vec{t}$ take only 0/1 values on 0/1 inputs. This fact and some counterexamples are discussed in the appendix to this paper.

2. Designs

2.1. Designs and the degrees of refutations. We are interested in obtaining exact upper and lower bounds on the degrees of Nullstellensatz refutations of particular sets of polynomials. Upper bounds are generally best obtained by explicit construction of Nullstellensatz refutations. For lower bounds, the most useful tool so far has been the use of combinatorial constructions called designs. A general definition of a design, that applies to Nullstellensatz refutations of arbitrary sets of polynomials, is as follows.

Definition Fix an algebraic structure R and a set of polynomials $\mathcal{F} = \{F_i\}_i$. Let $d \geq 0$. A d-design for $\mathcal{F}$ is a mapping D from R-polynomials of degree $\leq d$ to R such that the following conditions hold:

(1): $D(1) = 1$. The polynomial 1 is mapped to the constant 1.

(2): D is linear. So $D(\alpha P) = \alpha D(P)$ and $D(P+Q) = D(P) + D(Q)$, for all $\alpha \in R$ and all R-polynomials P and Q.
(3): $D(Q \cdot F_i) = 0$ for all polynomials $F_i \in \mathcal{F}$ and all polynomials Q such that $deg(F_i) + deg(Q) \leq d$.
(4): $D(x^2 Q) = D(xQ)$ for all variables x and all polynomials Q of degree less than $d-1$.

A d-design is also called a design of degree d.

A *monomial* (sometimes referred to as a *power product*) is an expression of the form $x_1^{a_1} \cdots x_n^{a_n}$. A monomial is *multilinear* if the exponents a_i are all in $\{0,1\}$. A couple of easy observations about designs are:

(a): Since D is linear, D is completely determined by its values $D(Q)$ for all monomials Q.
(b): Likewise by linearity, it suffices that (3) and (4) hold for all *monomials* Q.
(c): The property of D being a d-design is completely independent of D's values on monomials of degree $> d$.
(d): By virtue of (4), the values of D are completely determined by its values on multilinear monomials. Alternatively, we could have included the polynomials $x_j^2 - x_j$ in the set $\mathcal{F}$ and then condition (4) would have been a consequence of (3).

We therefore will frequently find it convenient to define designs by specifying their values only on the multilinear monomials of degree $\leq d$. This viewpoint leads to the construction of designs as combinatorial objects; namely, the multilinear monomials can be viewed as conjunctions of the atomic statements represented by the (propositional) variables in the monomial.

The next theorem provides the basis for using designs to obtain lower bounds on the degrees of Nullstellensatz refutations.

THEOREM 3. *If $\mathcal{F}$ has a d-design, then $\mathcal{F}$ does not have a Nullstellensatz refutation of degree $\leq d$.*

PROOF. Suppose there exists a Nullstellensatz refutation

$$1 = \sum_i P_i \cdot F_i + \sum_j Q_j \cdot (x_j^2 - x_j)$$

of degree $\leq d$. Applying D to both sides, we have

$$\begin{aligned} 1 = D(1) &= D(\sum_i P_i \cdot F_i + \sum_j Q_j \cdot (x_j^2 - x_j)) \\ &= \sum_i D(P_i \cdot F_i) + \sum_j (D(x_j^2 Q_j) - D(x_j Q_j)) \\ &= 0 \end{aligned}$$

which is a contradiction. □

A converse to the above theorem also holds:

THEOREM 4. *Suppose the algebraic structure R is a field. If $\mathcal{F}$ does not have a Nullstellensatz refutation of degree d, then there is a d-design for $\mathcal{F}$.*

PROOF. The essential idea of the proof is to restate the question of whether a degree d refutation exists to a problem of finding a solution to a linear programming problem. The question of whether a design exists turns out to be the dual problem. Since it involves only elementary concepts from linear algebra, we give only a sketch of the proof.

Let δ be the number of monomials of degree $\leq d$; i.e., $\delta = \sum_{i=0}^{d} \binom{n+i-1}{i}$. Any polynomial H of degree $\leq d$ can be viewed as a column vector $\mathbf{v}_H$ of dimension δ by letting the entries in the vector equal the coefficients of the the monomials in H. We order the entries so that the last entry of $\mathbf{v}_H$ is the constant term of H.

Let $\mathcal{G}$ be the set of polynomials of the forms $Q \cdot F$ for Q a monomial, for F in $\mathcal{F}$ or of the form $x^2 - x$, and where $deg(QF) \leq d$. Let $\mathbf{v}_1$ be the vector with last entry equal to 1 and all other entries zero. Clearly, there is a Nullstellensatz refutation of degree $\leq d$ if only if the vector $\mathbf{v}_1$ can be expressed as an R-linear combination of the vectors $\mathbf{v}_{G_i}$ for $G_i \in \mathcal{G}$. Letting $\mathbf{M}$ be the $\delta \times |\mathcal{G}|$ matrix which has columns $\mathbf{v}_{G_i}$, this is equivalent to having solution $\mathbf{w}$ to the linear equation

$$\mathbf{M}\mathbf{w} = \mathbf{v}_1, \tag{2}$$

where $\mathbf{w}$ is a vector of dimension $|G|$. Let $\mathbf{M}^-$ be the matrix obtained by deleting the last row from $\mathbf{M}$. Since the first $\delta - 1$ entries of $\mathbf{v}_1$ are zero, any solution $\mathbf{w}$ to (2) must be in the nullspace (the kernel) of the mapping $\mathbf{M}^-$. If the last row of $\mathbf{M}$ is in the span of the row vectors of $\mathbf{M}^-$, $\mathbf{M}^-\mathbf{w} = \mathbf{0}$ implies $\mathbf{M}\mathbf{w} = 0$. On the other hand, if the final row is not in their span, then is it possible to find $\mathbf{w}$ satisfying (2). Therefore, there is a Nullstellensatz refutation of degree $\leq d$ if and only if the last row of $\mathbf{M}$ is linearly independent of the rest of the row vectors.

So suppose that the last row of $\mathbf{M}$ is a linear combination of the first $\delta - 1$ rows of $\mathbf{M}$. We must prove that there exists a d-design. Each row in $\mathbf{M}$ corresponds to a monomial Q of degree $\leq d$; we denote that row $\mathbf{u}_Q$. So we have

$$\sum_{deg(Q) \leq d} \alpha_Q \mathbf{u}_Q = 0,$$

where $\alpha_Q \in R$ and where $\alpha_1 = 1$. Define the design D by letting $D(Q) = \alpha_Q$ for all monomials Q. It is not difficult to check that D is a valid d-design. □

Remark: It is not entirely clear who was the original discoverer of designs. This author first heard about designs for the special case of tautologies expressing mod m tautologies in a communication from P. Pudlák, who certainly knew the corresponding special cases of Theorems 3 and 4. In any event, the next section discusses all the published work known to us.

2.2. Prior constructions of designs. Designs have been used to obtain several degree lower bounds for Nullstellensatz refutations.

First, [1] gave designs of degree $\sqrt{n}$ for polynomials which express a version of the pigeonhole principle. This thereby established a $\sqrt{n}$ lower bound on the degree of Nullstellensatz refutations of the pigeonhole principle. It is still open whether this lower bound can be improved substantially, although we conjecture that the correct lower bound is $\Omega(n)$.

Second, [5] gave linear degree designs for the housesitting principle over the field $\mathbb{Z}_2$. The housesitting principle is a form of strong induction. We describe the housesitting principle below and give a new construction of designs for the housesitting principle: our construction shows that the designs work over *any* ring R.

Third, [4] proved logarithmic upper and lower bounds on the degree of Nullstellensatz proofs of the induction principle. Their proof uses a complicated, explicit construction of designs for the induction principle. A simpler derivation of the degree bounds has very recently been obtained by Clegg and Impagliazzo based on the generation of Gröbner basis for the induction principle polynomials; their method does not explicitly construct the design however.

Recently, [3] has given explicit constructions of designs for the mod m matching principles of size n^ϵ. This establishes corresponding lower bounds on the degrees of Nullstellensatz refutations and thereby, using a result from [2], gives exponential lower bounds on the size of constant depth Frege proofs of the mod m matching principle in the presence of mod q counting axioms (with q and m relatively prime).

2.3. Designs for the Gröbner proof system? Designs have been an important tool for obtaining lower bounds on the degrees of Nullstellensatz refutations. A generalization of Nullstellensatz proofs, called "Gröbner proof systems," has been recently proposed by Clegg, Edmonds and Impagliazzo [5]; so far, no non-trivial degree lower bounds for Gröbner proofs have been obtained. (See also [3] for Gröbner proof systems.)

This raises the question of whether designs can be generalized to give degree lower bounds on Gröbner proofs. For this, we define a *Gröbner-design* just as we defined design, but we add an extra condition:

(5): If $D(P) = 0$ and if $\deg(P) + \deg(Q) \leq d$, then $D(Q \cdot P) = 0$.

Of course, with this extra condition present, then conditions (3) (4) can be weakened to state that $D(F_i) = 0$ for all $F_i \in \mathcal{F}$ and that $D(x^2 - x) = 0$ for all variables x.

The following theorem follows easily from the definition of Gröbner designs and the Gröbner proof system:

THEOREM 5. *If there is a Gröbner design of degree d, then there is no Gröbner proof of degree $\leq d$.*

We do not know if the converse to this theorem holds.

3. The Housesitting Principle

The rest of this paper discusses designs for a propositional tautology known as the housesitting principle. The construction of these designs provides a simple, instructive example of the construction of designs.

For the sequel, we let $I = \{0, \ldots, n\}$ and $J = \{1, \ldots, n\}$. For an intuitive picture, we think of J as a linearly ordered set of houses and of I as a set of people who occupy houses. Each person $i \in I$ either stays at home in house i, or housesits for some house $j > i$ for which person j is not at home. Since $0 \notin J$, person 0 must housesit. It is allowed that two people housesit for the same house. The housesitting principle states that these properties cannot be satisfied for all $i \in I$ simultaneously.

The housesitting principle can be viewed as a form of the complete induction principle. That is, let $A(k)$ be the assertion that every person $i \geq n - k$ is at home (i.e., in house i). Then trivially $A(0)$ holds, and it is easy to note that $A(k)$ implies $A(k+1)$. But $A(n)$ is a contradiction since person 0 must housesit some house in J.

Alternatively, the housesitting principle is a form of the infinite descent principle, except that our ordering is reversed so it becomes an infinite *ascent* principle. Namely, let $j_0 = 0$ and let j_{i+1} be the house occupied by person j_i. Then if the housesitting conditions were all met, $j_0, j_1, j_2, \ldots$ would be infinite, strictly increasing sequence of integers less than n.

3.1. The Nullstellensatz formulation. The set of polynomials used to express (the negation of) the housesitting principle in the Nullstellensatz proof setting are constructed as follows.

There are variables $x_{i,j}$, for all $0 \leq i \leq j \leq n$, $1 \leq j$, which intuitively express the condition that person i is in house j (a value of 1 denotes *True* and a value of 0 denotes *False*). There are linear polynomials F_i which state that person i is in some house numbered at least i:

$$F_i \;=\; x_{i,i} + x_{i,i+1} + \cdots + x_{i,n} - 1.$$

There are degree 2 polynomials $F'_{i,j}$, for $i < j$, which state that persons i and j are not both in house j:

$$F'_{i,j} \;=\; x_{i,j} x_{j,j}.$$

Finally there are the usual propositional polynomials, $F''_{i,j} = x_{i,j}^2 - x_{i,j}$.

Let $\mathcal{F}$ be the set of polynomials $\{F_i, F'_{i,j}\}$. Since the polynomials cannot be simultaneously equal to zero under propositional assignments to the variables (since otherwise the housesitting principle would be falsified), there must be a Nullstellensatz refutation of $\mathcal{F}$. We shall construct below an n-design which proves that any Nullstellensatz refutation must have degree at least $n + 1$, over an arbitrary ring R.

Actually, we will do a little better: let $F^{(3)}_{i,k,\ell}$ be the polynomials

$$F^{(3)}_{i,k,\ell} \;=\; x_{i,k} x_{i,\ell}$$

that $D(\emptyset) = 1$. Now, $r(\emptyset) = 0$, so $D(\emptyset) = 1$ as desired. Property (3) holds for the polynomials $F'_{i,j} \in \mathcal{F}$ because D maps non-partial mappings to 0; i.e., for any monomial T, $D(T \cdot F'_{i,j}) = 0$ since condition (β) is violated when $\pi(i) = \pi(j) = j$. Likewise, $D(T \cdot F^{(3)}_{i,k,\ell})$ must equal zero. In addition, condition (γ) implies that $D(T \cdot F^{(4)}_{i,j,k,\ell}) = 0$.

So it remains to establish property (3) for the polynomials $F_i \in \mathcal{F}$. For this, we must show that $D(T \cdot F_s) = 0$ for all s and all monomials T of degree $< n$. Since D is non-zero only for monomials which represent partial mappings, we may assume w.l.o.g. that T is a partial mapping, T_π, identified with a partial mapping π. By the linearity of D and the definitions of D and F_s, it will suffice to prove that

$$D(T_\pi) = \sum_{t \geq s} D(T_\pi x_{s,t}),$$

i.e., that $D(\pi) = \sum_{t \geq s} D(\pi \cup \{(s,t)\})$, where $\pi \cup \{(s,t)\}$ means the extension of π that sends s to t. For s in the domain of π, this is easy to show, so we assume henceforth that $s \notin dom(\pi)$.

We write $\pi' \supset_s \pi$ to denote the condition that π' is an extension of π and that $\{s\} = dom(\pi') \setminus dom(\pi)$. Therefore we can reexpress the previous equation as:

$$D(\pi) \;=\; \sum_{\pi' \supset_s \pi} D(\pi'). \tag{4}$$

Fix an an arbitrary matching π of degree $< n$. Let $s \notin dom(\pi)$. Let $r = r(\pi)$. We need to establish that equation (4) holds.

Case (1): $D(\pi) \neq 0$.

Case (1.a): If $s > r$, then, by condition (δ), the only $\pi' \supset_s \pi$ with $D(\pi') \neq 0$, is the one with $\pi'(s) = s$. Also, for this π', $D(\pi') = D(\pi)$ by the definition of D.

Case (1.b): If $s = r$, let r' be the least value greater than r not in the domain of π. By the lemma, if $\pi' \supset_s \pi$ and $D(\pi') \neq 0$, then $\pi'(s) = r'$. Also, since $D(\pi) \neq 0$ and by condition (δ), we have $\pi(i) = i$ for all $r < i < r'$. Therefore, for the π' such that $\pi'(s) = r'$, we have $D(\pi') = D(\pi)$, by the definition of D.

Case (2): $D(\pi) = 0$. This case splits into subcases based on the reason that $D(\pi) = 0$. In each case we show that $\sum_{\pi' \supset_s \pi} D(\pi') = 0$. We start with the cases that imply that $D(\pi') = 0$ for all $\pi' \supset \pi$.

Case (2.a): Suppose condition (α) or (β) is violated by π. In this case, it is clear that the same condition is violated by any $\pi' \supset_s \pi$, so $D(\pi') = 0$ for all $\pi' \supset_s \pi$.

Case (2.b): Suppose π satisfies conditions (α) and (β) and that $s > r$. In this case, $r(\pi') = r(\pi)$. First, if condition (δ) fails for π, then any $\pi' \supset_s \pi$ also fails to satisfy (δ). Second, suppose (γ) fails for π for the values $i < j < \pi(i)$. If $j \neq s$, then the same condition fails for any $\pi' \supset_s \pi$. If $j = s$, then $\pi' \supset_s \pi$ could satisfy (γ) only if $\pi'(s) = \pi(i) > s$, which would cause condition (δ) to be violated for $\pi\prime$ since $s > r(\pi')$. In any event, $D(\pi') = 0$ for all $\pi' \supset_s \pi$.

Case (2.c): Suppose $s = r$ and there is some $i < j < \pi(i)$ which cause condition (γ) to be violated for π. Clearly the same condition is violated for any $\pi' \supset_s \pi$.

Case (2.d): Suppose $s = r$ and there an $i > r(\pi')$ such that $\pi(i) \neq i$ which causes condition (δ) to fail for π. Obviously this also causes (δ) to fail for any $\pi' \supset_s \pi$.

Now we treat the cases where there is a $\pi' \supset_s \pi$ with $D(\pi') \neq 0$. Therefore none of the above cases hold and $s = r$. From Lemma 8, there must be an $i_0 < s \leq \pi(i_0)$.

Case (2.e): Suppose $\pi(i_0) > s$. Condition (δ) implies that any other $i < s$ with $\pi(i) \geq s$, must satisfy $\pi(i) = \pi(i_0)$. Also, condition (γ) implies that if $s \leq i < \pi(i_0)$, then $\pi'(i) \in \{i, \pi(i_0)\}$. In particular, $\pi'(s) \in \{s, \pi(i_0)\}$. Thus there are two possible $\pi' \supset_s \pi$ such that $S(\pi') \neq 0$:

$$\pi'_1 = \pi \cup \{(s,s)\}$$

and

$$\pi'_2 \;=\; \pi \cup \{(s, \pi(i_0))\}.$$

It is easily checked that they both satisfy all the conditions (α)-(δ) and that $D(\pi'_1) = -D(\pi'_2)$ since s is in the range of π'_1, but not in the range of π'_2. Thus

$$D(\pi) \;= 0 \;=\; D(\pi'_1) + D(\pi'_2) \;=\; \sum_{\pi' \supset_s \pi} D(\pi').$$

Case (2.f): Suppose $\pi(i_0) = s$. Since $D(\pi) = 0$ and since the conditions of cases (2.a)-(2.d) have been ruled out, there must be a least $i_1 > s$ such that $i_1 < r(\pi')$ and $i_1 < \pi(i_1)$, which causes condition (δ) to fail for π. In this case, for $D(\pi')$ to be non-zero, condition (γ) implies that $\pi'(s) \in \{i_1, \pi(i_1)\}$. Thus there are two possible $\pi' \supset_s \pi$ with $D(\pi') \neq 0$, namely,

$$\pi'_1 = \pi \cup \{(s, i_1)\}$$

and

$$\pi'_2 \;=\; \pi \cup \{(s, \pi(i_1))\}.$$

By similar reasoning to the previous case, $D(\pi'_1) = -D(\pi'_2)$, and the desired result follows as before.

□

4. Appendix: (non)completeness over rings

In section 1, we stated the completeness theorem for Nullstellensatz refutation for fields. When working over rings, the completeness theorem

may not always hold. However, the following completeness theorems do apply to rings.

THEOREM 9. *Let R be a commutative ring. Suppose there is no assignment of 0/1 values to variables that make $t_1, \ldots, t_m$ simultaneously equal to zero.*

(a): *Further suppose that R does not have any zero divisors. Then there is a nonzero $a \in R$ and polynomials F_i and G_j so that*

$$\sum_i F_i \cdot t_i + \sum_j G_j \cdot (x_j^2 - x_j) = a. \tag{5}$$

(b): *Now suppose instead that $t_1, \ldots, t_n$ assume only 0/1 values for all assignments of 0/1 values to the variables. That is to say, suppose each t_i is equivalent to a propositional formula. Then $t_1, \ldots, t_m$ have a Nullstellensatz refutation.*

We leave the proof of the above theorem to the reader. Note that the situation of part (a) does not quite give a Nullstellensatz refutation since a may not equal 1. However, since $a \neq 0$, (5) can be viewed as a refutation since it implies that there is no propositional assignment that makes the t_i's simultaneously zero.

We conclude with a couple of examples of situations where the Nullstellensatz completeness theorem fails. Both examples use the ring $R = \mathbb{Z}_6$. First consider solutions of

$$\begin{aligned} y + 1 &= 0 \\ y^2 - y &= 0 \end{aligned}$$

From these, one can use a Nullstellensatz refutation to derive 0, 2 or 4; e.g.,

$$(2 - y)(y + 1) + (y^2 - y) = 2.$$

However, there are no polynomials such that $F(y)(y+1) + G(y)(y^2 - y) = 1$ over $\mathbb{Z}_6$. Second, consider solutions of

$$\begin{aligned} y + 2 &= 0 \\ y^2 - y &= 0 \end{aligned}$$

Here the situation is worse, the only value $a \in \mathbb{Z}_6$ for which

$$F(y)(y + 1) + G(y)(y^2 - y) = a$$

is possible is $a = 0$. To prove this last fact, note that $y = 4$ is a solution to the two equations.

References

[1] P. BEAME, S. COOK, J. EDMONDS, R. IMPAGLIAZZO, AND T. PITASSI, *The relative complexity of NP search problems*, in Proceedings of the 27th ACM Symposium on Theory of Computing, 1995, pp. 303–314.

[2] P. BEAME, R. IMPAGLIAZZO, J. KRAJÍČEK, T. PITASSI, AND P. PUDLÁK, *Lower bounds on Hilbert's Nullstellensatz and propositional proofs*, in Thirty-fifth Annual Symposium on Foundations of Computer Science, IEEE Press, 1994, pp. 794–806. Revised version to appear in Proceedings of the London Mathematical Society.

[3] S. R. BUSS, R. IMPAGLIAZZO, J. KRAJÍČEK, P. PUDLÁK, A. A. RAZBOROV, AND J. SGALL, *Proof complexity in algebraic systems and constant depth Frege systems with modular counting*. To appear in *Computational Complexity*, 199?

[4] S. R. BUSS AND T. PITASSI, *Good degree lower bounds on nullstellensatz refutations of the induction principle*, in Proceedings of the Eleventh Annual Conference on Structure in Complexity Theory, IEEE Computer Society, 1996, pp. 233–242. Revised version to appear in *J. Computer and System Sciences*.

[5] M. CLEGG, J. EDMONDS, AND R. IMPAGLIAZZO, *Gröbner proofs*, in Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing, Association for Computing Machinery, 1996, pp. 174–183.

[6] T. PITASSI, *Algebraic propositional proof systems*. To appear in the proceedings volume of the DIMACS workshop on *Finite Models and Descriptive Complexity, January 14-17, 1996*, 1995.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF CALIFORNIA, SAN DIEGO
E-mail address: sbuss@ucsd.edu

may not always hold. However, the following completeness theorems do apply to rings.

THEOREM 9. *Let R be a commutative ring. Suppose there is no assignment of 0/1 values to variables that make $t_1, \ldots, t_m$ simultaneously equal to zero.*

(a): *Further suppose that R does not have any zero divisors. Then there is a nonzero $a \in R$ and polynomials F_i and G_j so that*

$$\sum_i F_i \cdot t_i + \sum_j G_j \cdot (x_j^2 - x_j) \ = a. \tag{5}$$

(b): *Now suppose instead that $t_1, \ldots, t_n$ assume only 0/1 values for all assignments of 0/1 values to the variables. That is to say, suppose each t_i is equivalent to a propositional formula. Then $t_1, \ldots, t_m$ have a Nullstellensatz refutation.*

We leave the proof of the above theorem to the reader. Note that the situation of part (a) does not quite give a Nullstellensatz refutation since a may not equal 1. However, since $a \neq 0$, (5) can be viewed as a refutation since it implies that there is no propositional assignment that makes the t_i's simultaneously zero.

We conclude with a couple of examples of situations where the Nullstellensatz completeness theorem fails. Both examples use the ring $R = \mathbb{Z}_6$. First consider solutions of

$$\begin{aligned} y + 1 &= 0 \\ y^2 - y &= 0 \end{aligned}$$

From these, one can use a Nullstellensatz refutation to derive 0, 2 or 4; e.g.,

$$(2 - y)(y + 1) + (y^2 - y) = 2.$$

However, there are no polynomials such that $F(y)(y+1) + G(y)(y^2 - y) = 1$ over $\mathbb{Z}_6$. Second, consider solutions of

$$\begin{aligned} y + 2 &= 0 \\ y^2 - y &= 0 \end{aligned}$$

Here the situation is worse, the only value $a \in \mathbb{Z}_6$ for which

$$F(y)(y + 1) + G(y)(y^2 - y) \ = \ a$$

is possible is $a = 0$. To prove this last fact, note that $y = 4$ is a solution to the two equations.

References

[1] P. BEAME, S. COOK, J. EDMONDS, R. IMPAGLIAZZO, AND T. PITASSI, *The relative complexity of NP search problems*, in Proceedings of the 27th ACM Symposium on Theory of Computing, 1995, pp. 303–314.

[2] P. BEAME, R. IMPAGLIAZZO, J. KRAJÍČEK, T. PITASSI, AND P. PUDLÁK, *Lower bounds on Hilbert's Nullstellensatz and propositional proofs*, in Thirty-fifth Annual Symposium on Foundations of Computer Science, IEEE Press, 1994, pp. 794–806. Revised version to appear in Proceedings of the London Mathematical Society.

[3] S. R. BUSS, R. IMPAGLIAZZO, J. KRAJÍČEK, P. PUDLÁK, A. A. RAZBOROV, AND J. SGALL, *Proof complexity in algebraic systems and constant depth Frege systems with modular counting*. To appear in *Computational Complexity*, 199?

[4] S. R. BUSS AND T. PITASSI, *Good degree lower bounds on nullstellensatz refutations of the induction principle*, in Proceedings of the Eleventh Annual Conference on Structure in Complexity Theory, IEEE Computer Society, 1996, pp. 233–242. Revised version to appear in *J. Computer and System Sciences*.

[5] M. CLEGG, J. EDMONDS, AND R. IMPAGLIAZZO, *Gröbner proofs*, in Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing, Association for Computing Machinery, 1996, pp. 174–183.

[6] T. PITASSI, *Algebraic propositional proof systems*. To appear in the proceedings volume of the DIMACS workshop on *Finite Models and Descriptive Complexity, January 14-17, 1996*, 1995.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF CALIFORNIA, SAN DIEGO
E-mail address: sbuss@ucsd.edu

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

Relating the Provable Collapse of P to NC^1 and the Power of Logical Theories

Stephen Cook

ABSTRACT. We show that the following three statements are equivalent: **QPV** is conservative over **QALV**, **QALV** proves its open induction formulas, and **QALV** proves $\mathbf{P} = \mathbf{NC}^1$. Here **QPV** and **QALV** are first order theories whose function symbols range over polynomial-time and $\mathbf{NC}^1$ functions, respectively.

1. Introduction

The motivation for this paper comes from two sources. First, building on a result of Krajíček , Pudlák , and Takeuti [**KPT91**], Buss [**Bus95**] and Zambella [**Zam96**] independently proved that the bounded arithmetic hierarchy of theories $\mathbf{S}^i_2$ collapses if and only if the polynomial time hierarchy provably collapses. We prove that the arithmetic theory **QPV** of polynomial time functions is conservative over the arithmetic theory **QALV** of $\mathbf{NC}^1$ functions if and only if **P** provably collapses to $\mathbf{NC}^1$. (**QPV** is called $\mathbf{PV}_1$ in [**Bus95**].) The second motivation is an attempt to relate the possible collapse of **P** to $\mathbf{NC}^1$ with the possible p-simulation of extended Frege propositional proof systems by Frege systems.

The theory **QPV** considered here is a quantified version of the author's equational theory **PV** [**Coo75**], in which the function symbols represent the polynomial time functions. **PV** is well-studied [**CU93, Kra95a**], and in particular Buss [**Bus86**] showed that the theorems of **PV** are inter-translatable with the Σ^b_1 theorems of $\mathbf{S}^1_2$.

The theory **QALV** is some analog of **QPV** for the $\mathbf{NC}^1$ functions; i.e. the functions computable by uniform log depth polynomial size circuit families. By a theorem of Ruzzo [**Ruz81**], the $\mathbf{NC}^1$ functions are the same as the **ALOGTIME** functions; the functions whose i-th bit relations are computable by alternating Turing machines in log time. There are several possible choices for **QALV**. The one we select is a quantified version of Clote's equational theory $\mathbf{ALV}'$ [**Clo93**]. Other possibilities could be based on Pitt's quantifier-free theory $\mathbf{T}_1$ [**Pit96**], or Arai's first-order theory **AID** [**Ara91**]. It may turn out that all these theories are essentially equivalent, but so far this is an unproven conjecture.

The main reason for selecting $\mathbf{ALV}'$ is that it closely resembles **PV**, and in fact **PV** can be regarded naturally as an extension of $\mathbf{ALV}'$. From our point of

---

1991 *Mathematics Subject Classification.* Primary 03F30; Secondary 68Q15, 03F20.

Research supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Information Technology Research Centre, Ontario (ITRC).

view, the disadvantage of $\mathbf{T}_1$ is that it is a theory of strings rather than numbers, and the disadvantage of **AID** is that, although its predicate symbols range over all $\mathbf{NC}^j$ relations, it has only finitely many function symbols.

A possible disadvantage of **ALV′** is that the proof that its function symbols represent all $\mathbf{NC}^1$ functions rests on Barrington's theorem [**Bar89**], which involves an interesting construction using unsolvable groups. Thus functions as simple as the majority function require relatively complex definitions, and proofs of their properties appear to be complex. However it turns out that the functions we need in this paper are relatively simple and can be represented in **ALV′** without Barrington's construction. Nevertheless it remains an open question whether **QALV** (quantified **ALV′**) is equivalent, for example, to **AID**.

A fundamental open question in propositional proof complexity is whether Frege systems can efficiently simulate extended Frege systems [**CR79, Kra95a**]. Intuitively this question is related to the question of whether $\mathbf{P} = \mathbf{NC}^1$, since formulas in a Frege proof represent $\mathbf{NC}^1$ concepts and formulas in an extended Frege proof represent polynomial time concepts. However, no formal relation between the two questions has been proven.

There are known relationships between the power of logical theories and the power of Frege and extended Frege proof systems. In general, $\forall$-theorems of theories which capture reasoning with polynomial time (respectively, $\mathbf{NC}^1$) concepts correspond to sequences of tautologies with polynomial size extended Frege (respectively, Frege) proofs, and such theories prove the soundness of extended Frege (respectively, Frege) systems.

In the case of polynomial time concepts, the author [**Coo75**] carried out this correspondence for his equational theory **PV**, and argued that **PV** proves the soundness of extended Frege systems. The correspondence has been extended to show that the $\forall\Sigma^b_1$-theorems of Buss's first order theory $\mathbf{S}^1_2$ and the $\forall\Sigma^{1,b}_0$ theorems of the second order theory $\mathbf{V}^1_1$ correspond to tautology sequences with polynomial size extended Frege proofs [**Kra95b, Kra95a**].

In the case of $\mathbf{NC}^1$ concepts, several authors have results. Clote [**Clo90, Clo93**] showed that the theorems of his equational theories **ALV** and **ALV′** correspond to tautology sequences with polynomial size Frege proofs. Arai [**Ara91**] did the same for the Σ^b_0 theorems of his first order theory **AID** and Pitt [**Pit96**] did the same for his quantifier-free theory $\mathbf{T}_1$. Krajicek [**Kra95b**] showed that every $\forall\Sigma^{1,b}_0$ theorem of the second order theory $\mathbf{U}^1_1$ corresponds to a sequence of tautologies with quasi-polynomial size Frege proofs, but apparently no one has yet defined a satisfactory second order theory that nicely corresponds to polynomial size Frege proofs (see [**Kra95b**], page 297).

Buss [**Bus91**] proved that Frege systems have polynomial size partial self-consistency proofs. Arai showed how to formalize this in his system **AID**, to show that **AID** proves the soundness of Frege proof systems. Buss also showed that Frege systems have polynomial size proofs of partial consistency of extended Frege systems iff Frege systems p-simulate extended Frege systems. It seems plausible that this result can also be formalized in **AID**. Assuming this and a suitable interpretation of **AID** in **PV**, we would have

Assertion: The following statements are equivalent:

1. **QPV** is $\forall\Pi^b_1$-conservative over **AID**.
2. **AID** proves the soundness of extended Frege systems.
3. **AID** proves that Frege systems p-simulate extended Frege systems.

Proof sketch: $1 \Longrightarrow 2$: **QPV** proves the soundness of extended Frege systems [**Coo75**], and using the methods of [**Bus91**] and [**Ara91**] this soundness assertion can be formulated as a $\forall\Pi^b_1$-sentence in **AID**.

$2 \Longrightarrow 3$: Formalize Buss's theorem [**Bus91**] in **AID**.

$3 \Longrightarrow 1$: Let A be a $\forall\Pi^b_1$-formula of **AID** which is provable in **QPV**. Then by results in [**Coo75**] (see [**Kra95a**]), A gives rise to a sequence of propositional tautologies which have polynomial size extended Frege proofs. Even though the **QPV** proof involves polynomial time concepts which may not be in $\mathbf{NC}^1$, these extended Frege proofs are uniform and can be described by $\mathbf{NC}^1$ functions, and there existence can be proved in **AID**. By assumption 3, **AID** proves that these tautologies have polynomial size Frege proofs. Since **AID** proves the soundness of Frege systems, **AID** proves A. □

It would be very interesting to show that a fourth statement is equivalent to the above three: **AID** proves that $\mathbf{P} = \mathbf{NC}^1$. However by Theorem 7 below this would be the same (assuming that **AID** and **QALV** are equivalent theories: see above) as showing that Π^b_1-conservativity of **QPV** over **AID** is equivalent to Σ^b_1-conservativity. Although in general the Σ^b_1-conservativity of theories is not the same as Π^b_1 conservativity, it seems a plausible conjecture in this case.

The hard part of the proof of our main theorem (Theorem 7) is showing that if **QALV** proves its open induction formulas, then $\mathbf{P} = \mathbf{NC}^1$, and this equality is provable in **QALV**. At a high level, our proof resembles Buss's [**Bus95**] proof that if $\mathbf{S}^1_2(\mathbf{PV}) = \mathbf{QPV}$, then the polynomial hierarchy collapses. Buss's proof is based on the KPT witnessing theorem [**KPT91**], applied to a $\forall\exists\forall$ formula. Our proof is based on a simpler traditional witnessing theorem, applied to a $\forall\exists$ formula (induction).

2. The theories QPV and QALV

2.1. The equational theories. We start by presenting an outline of the equational theory **PV** following [**CU93**]. The function symbols of **PV** stand for polynomial time computable functions (members of **FP**), and every such function is represented by infinitely many function symbols. The function symbols are introduced in a manner based on Cobham's [**Cob65**] characterization of **FP** as the least class including certain initial functions and closed under composition and bounded recursion on notation.

In more detail, the *initial* function symbols of **PV** are 0, s_0, s_1, *parity*, *tr*, *cond*, *pad*, *trunc*, and $\#$. The intended interpretations of these are $s_0(x) = 2x$, $s_1(x) = 2x+1$, $parity(x) = x \bmod 2$, $tr(x) = \lfloor x/2 \rfloor$, $cond(0,y,z) = y$ and $cond(x,y,z) = z$ for $x > 0$, $pad(x,y) = 2^{|y|} \cdot x$, where $|y| = \lceil \log_2(y+1) \rceil$, $trunc(x,y) = \lfloor x/2^{|y|} \rfloor$, and $x\#y = 2^{|x|\cdot|y|}$.

In general, if t is a **PV** term, and $x_1, ..., x_n$, $n \geq 0$, is a list of variables including all variables in t, then $[\lambda x_1 ... x_n . t]$ is an n-place function symbol. If g, h, and k are n-place, $n+2$-place, and $n+1$-place function symbols respectively, then $R[g,h,k]$ is an $n+1$-place function symbol, representing the result of bounded recursion on notation from g, h, and k.

The axioms of PV consist of $s_0(0) = 0$, together with one, two, or three *defining equations* for each function symbol, except s_0 and s_1 have no defining equation. The defining equations for the remaining initial functions characterize their meaning via recursion on notation. The rules of PV are the usual rules for equality, together with induction on notation.

See Appendix A for a complete list of axioms and rules.

Now we present an outline of the equational theory ALV′, following [Clo93]. ALV′ is similar to PV, except that the function symbols range over $\mathbf{NC}^1$ functions instead of polynomial time functions. The $\mathbf{NC}^1$ function symbols are introduced in a manner based on Barrington's characterization of $\mathbf{NC}^1$ in terms of bounded width branching programs [Bar89, BIS90], and adapted by Clote [Clo93] to show that the $\mathbf{NC}^1$ functions are the least class including certain initial functions and closed under composition, concatenation recursion on notation, and K-bounded recursion on notation (for constant K).

In more detail, the initial functions of ALV′ include all of the initial functions of PV, and also s and $|x|$. (The intended interpretation of s is $s(x) = x + 1$.) In general, if t is a PV term, and $x_1, ..., x_n$, $n \geq 0$, is a list of variables including all variables in t, then $[\lambda x_1 ... x_n.t]$ is an n-place function symbol. If g, h, and k are n-place, $n+1$-place, and $n+1$-place function symbols respectively, then $R^{crn}[g, h, k]$ is an $n+1$-place function symbol, representing the result of concatenation recursion on notation from g, h, and k. If g, h, and k are n-place, $n+2$-place, and $n+2$-place function symbols respectively, and $\ell > 0$, then $R_\ell[g, h, k]$ is an $n+1$-place function symbol, representing the result of K-bounded recursion on notation from g, h, and k, where $K = 2^\ell - 1$.

The axioms of ALV′ again consist of $s_0(0) = 0$ and defining equations for each function symbol except s_0 and s_1. The rules of ALV′ are the same as for PV. See Appendix A for a complete list of axioms and rules.

2.2. Interpreting ALV′ in PV. We now give a simple interpretation of ALV′ in PV. For each ALV′ function symbol f we define an equivalent PV function symbol f^π with the following property: If E is a defining equation for f in ALV′, then PV proves E^π, where E^π is the result of replacing each function symbol g in E by g^π. We define f^π by induction on the structure of f.

If f is an initial function of both ALV′ and PV, then set $f^\pi = f$. If f is an initial function of ALV′ but not of PV, then f is either s or $|\ |$. Both of these functions are easily defined in PV. In fact, referring to [CU93], s is defined in D24 and its ALV′ defining equations follow from T26, and $|x|$ is defined in D133 and its ALV′ defining equations follow from T135. We use these definitions to define s^π and $|\ |^\pi$.

If f is $[\lambda x_1 ... x_n.t]$, then we define f^π to be $[\lambda x_1 ... x_n.t^\pi]$, where t^π is the result of replacing each function symbol g in t by g^π. Then the transformed ALV′ defining equation for f is precisely the PV defining equation for f^π, and hence it is a theorem of PV.

The ALV′ function symbols $R^{crn}[g, h, k]$ and $R_\ell[g, h, k]$ represent forms of recursion on notation, and these functions can be defined using the PV recursor $R[g', h', k']$, for suitable g', h', and k'. In particular the bounding function k' for $R^{crn}[g, h, k]$ is $k'(x, \vec{y}) = g(\vec{y}) \star x$ (where $\star$ represents concatenation), and the bounding function for $R_\ell[g, h, k]$ is the constant $2^\ell - 1$. These bounds are provable in PV, and the transformed defining equations for $R^{crn}[g, h, k]$ and $R_\ell[g, h, k]$ follow.

We will identify each ALV′ function symbol f with its PV counterpart f^π. Thus we have proved

THEOREM 1. *PV is an extension of* ALV′.

2.3. The quantified theories. The systems QPV and QALV are quantified versions of PV and ALV′, respectively. They are first order theories whose non-logical symbols are those of PV (respectively ALV′) and whose axioms are the universal closures of the theorems (all of which are equations) of PV (respectively ALV′), together with two more:

Q1 $0 \neq 1$ (where $1 = s_1(0)$)
Q2 $\forall x(tr(x) = 0 \supset (x = 0 \vee x = 1))$

Thus we have

COROLLARY 1. *QPV is an extension of* QALV.

We emphasize that QALV and QPV are universally axiomatized theories, and we do not include any induction scheme among the axioms. However the axioms include all theorems of ALV′ and PV, respectively, and those theories include induction on notation as a rule. In Theorems 4 and 5 we show that suitable induction schemes are theorems of QALV and QPV.

It is obvious that Q1 is not a logical consequence of the theorems of ALV′ and PV, because the trivial structure with the single element 0 forms a model of both theories. It is less obvious, but true, that Q2 is not a logical consequence of either theory, and yet Q1 and Q2 do not add any new equational consequences of either theory. The next theorem states these results. We postpone the proof until Appendix B.

THEOREM 2. Q1 *is not a logical consequence of* ALV′ *and* Q2, *and* Q2 *is not a logical consequence of* ALV′ *and* Q1. *Similarly for* PV. QALV *is a conservative extension of* ALV′, *and* QPV *is a conservative extension of* PV.

In view of this result, we should ask whether there are simple truths like Q1 and Q2 in the language of QPV which are not theorems of QPV. We argue below that QPV and QALV prove the notational analogs of the Peano postulates, and prove the fact that the ranges of s_0 and s_1 are disjoint and together cover $\mathbb{N}$. Later in this section we introduce $\leq$, and argue that QPV proves all of the Σ_1^b theorems of $\mathbf{S}_2^1$.

The tr defining equations $tr(s_0(x)) = x$ and $tr(s_1(x)) = x$, and the theorem $tr(0) = 0$ immediately imply that QALV (and hence QPV) can prove that s_0 and s_1 are one-one functions, and $x \neq 0 \supset s_0(x) \neq 0$. To prove that $s_1(x) \neq 0$ and the ranges of s_0 and s_1 are disjoint we use Q1 and defining equations for s_0 and *parity*: $s_0(0) = 0$, $parity(s_0(x)) = 0$, and $parity(s_1(x)) = 1$.

By notational induction, ALV′ proves the following equations:

$$cond(x, 0, x) = x \tag{1}$$

$$cond(x, y, z) = cond(cond(x, 0, 1), y, z) \tag{2}$$

$$tr(cond(x, 0, 1)) = 0 \tag{3}$$

$$x = cond(parity(x), s_0(tr(x)), s_1(tr(x))). \tag{4}$$

By setting $y = 0$ and $z = x$ in (2) and using (1), we see that if $cond(x, 0, 1) = 0$ then $x = 0$, so by (3) and **Q2**, **QALV** proves $x \neq 0 \supset cond(x, 0, 1) = 1$. Hence by (2) and the defining equations for *cond*,

(5) $$\mathbf{QALV} \vdash x \neq 0 \supset cond(x, y, z) = z.$$

Finally, from this, (4), and the defining equations for *cond*, we have

(6) $$\mathbf{QALV} \vdash x = s_0(tr(x)) \vee x = s_1(tr(x)).$$

2.4. Induction formulas.

DEFINITION 1. *An* **ALV'** *function symbol* f_A represents *(in* **QALV***) a* **QALV** *formula* A *with free variables* $x_1, ..., x_n$ *if* **QALV** *proves* $A \leftrightarrow f_A(\vec{x}) = 0$.

We wish to show that a large class of **QALV** formulas are representable in this sense. Obviously any formula of the form $t = 0$ is representable, where t is a term, and if formulas A and B are representable, then so are $\neg A$, $A \wedge B$, $A \vee B$, and $A \supset B$, by suitable use of the function *cond*.

Let us say that $x \sqsubseteq y$ (x is an initial segment of y) iff $x = trunc(y, z)$ for some z. We can define *segment quantification* by

$$\forall x \sqsubseteq y A(x) \equiv \forall z A(trunc(y, z))$$

and

$$\exists x \sqsubseteq y A(x) \equiv \exists z A(trunc(y, z)).$$

If f_A represents A, and B is $\forall x \sqsubseteq y A(x)$ or $\exists x \sqsubseteq y A(x)$, then f_B is easily defined from f_A using the recursor R_1 (1-bounded recursion on notation), and the necessary equivalences can be proved in **QALV** using induction on notation. Therefore the representable formulas are closed under segment quantification.

Define $Bit(x, y) \equiv parity(trunc(x, y))$, so $Bit(x, y)$ is the $(|y| + 1)$-st bit of x from the right. Then define

$$x < y \equiv trunc(x, y) = 0 \wedge \exists z \sqsubseteq y[Bit(y, z) \neq 0 \wedge Bit(x, z) = 0 \wedge$$
$$\forall w \sqsubseteq y((trunc(z, w) = 0 \wedge w \neq z) \supset Bit(x, w) = Bit(y, w))].$$

Thus $x < y$ is representable. In fact the usual order properties can be proved in **QALV**, and in particular **QALV** proves $t = u \leftrightarrow \neg t < u \wedge \neg u < t$ (trichotomy). Thus every **ALV'** formula $t = u$ is representable in **QALV**.

Recall [**Bus86**] that a *sharply bounded* quantifier is one of the form $\forall x \leq |t|$ or $\exists x \leq |t|$, where t is a term. Since **QALV** proves $(\forall x \leq |y| A(x) \leftrightarrow \forall x \sqsubseteq y A(|x|))$ and $(\exists x \leq |y| A(x) \leftrightarrow \exists x \sqsubseteq y A(|x|))$ it follows that the representable formulas are closed under sharply bounded quantification. Recall that a Σ_0^b formula is one all of whose quantifiers are sharply bounded. All of the preceding paragraphs about representability clearly apply (and are well-known [**Bus86**])to **QPV** as well as **QALV**, so we have proved

THEOREM 3. *Every* Σ_0^b **QALV** *formula is representable in* **QALV**. *Every* Σ_0^b **QPV** *formula is representable in* **QPV**.

COROLLARY 2. *Every* Σ_0^b *formula of* **QALV** *represents an* $\mathbf{NC}^1$ *relation.*

Recall that **QALV** is universally axiomatized and has no induction scheme among its axioms. Nevertheless some induction formulas are theorems.

THEOREM 4 (Σ_0^b PIND Scheme). **QALV** *proves the P-induction formulas*

$$(A(0) \wedge \forall x(A(tr(x)) \supset A(x))) \supset \forall x A(x),$$

where $A(x)$ *is a* Σ_0^b *formula.*

Proof: Let $\vec{y}$ be a list of the variables in $A(x)$, other than x. Using concatenation recursion on notation, **QALV** can define the function F such that $F(x, \vec{y})$ is the least initial segment z of x such that $(A(tr(z)) \wedge \neg A(z)))$, and $F(x, \vec{y}) = x$ if no such z exists. Then by suitable use of induction on notation, **QALV** proves

$$(A(0) \wedge \neg A(x)) \supset (A(tr(F(x, \vec{y})) \wedge \neg A(F(x, \vec{y}))),$$

from which the P-induction formula follows. □

THEOREM 5 (Σ_0^b IND Scheme). **QPV** *proves the induction formulas*

$$(A(0) \wedge \forall x(A(x) \supset A(x + 1))) \supset \forall x A(x),$$

where $A(x)$ *is a* Σ_0^b *formula.*

Proof: Using the Dowd/Statman trick, **QPV** can use binary search to define a function G such that **QPV** proves

$$(A(0) \wedge \neg A(x)) \supset ((A(G(x, \vec{y})) \wedge \neg A(G(x, \vec{y}) + 1)),$$

from which the induction formula follows. □

We will show in the next section that **QALV** cannot prove the Σ_0^b induction formulas unless $\mathbf{P} = \mathbf{NC}^1$.

It follows from the proof of Theorem 4.7 of [**CU93**] that all of Buss's [**Bus86**] BASIC axioms for $\mathbf{S}_2^1$ are provable in **QPV**. Thus in view of Theorem 5 the arguments in chapter 6 of [**Bus86**] apply to show that **QPV** proves all of the Σ_1^b theorems of $\mathbf{S}_2^1$.

2.5. Witnessing in QALV. We now prove simple versions for **QALV** and **QPV** of Buss's witnessing theorem [**Bus86**].

THEOREM 6. *If* **QALV** *proves* $\forall \vec{x} \exists y A(\vec{x}, y)$, *where* $A(\vec{x}, y)$ *is a* Σ_0^b *formula, then there is a* **QALV** *function symbol* f *such that* **QALV** *proves* $\forall \vec{x} A(\vec{x}, f(\vec{x}))$. *The same holds when* **QPV** *replaces* **QALV** *throughout.*

Proof: By theorem 3, we need only consider the case $A(\vec{x}, y)$ is of the form $g(\vec{x}, y) = 0$. Since **QALV** is axiomatized by $\forall$-sentences, and **QALV** proves $\forall \vec{x} \exists y g(\vec{x}, y) = 0$, it follows from the Herbrand theorem that there are finitely many terms $t_1(\vec{x}),...,t_n(\vec{x})$ such that

$$\mathbf{QALV} \vdash \forall \vec{x}[g(\vec{x}, t_1(\vec{x})) = 0 \vee g(\vec{x}, t_2(\vec{x})) = 0 \vee ... \vee g(\vec{x}, t_n(\vec{x}) = 0].$$

We introduce a **QALV** function f such that $f(\vec{x}) = t_i(\vec{x})$, where i is the smallest index such that $g(\vec{x}, t_i(\vec{x})) = 0$. Thus the defining equation for f is

$$f(\vec{x}) = cond(g(\vec{x}, t_1(\vec{x})), t_1(\vec{x}), cond(g(\vec{x}, t_2(\vec{x})), t_2(\vec{x}), ...,$$
$$cond(g(\vec{x}, t_{n-1}(\vec{x})), t_{n-1}(\vec{x}), t_n(\vec{x})))...).$$

It follows easily that **QALV** $\vdash \forall \vec{x}\ g(\vec{x}, f(\vec{x})) = 0$. □

3. The main theorem

DEFINITION 2. *We say that* ALV′ proves $\mathbf{P} = \mathbf{NC}^1$ *iff for every* PV *function symbol* f *there is an equivalent* ALV′ *function symbol* f^ν *such that for each defining equation* A_f *for* f *in* PV, ALV′ *proves* A^ν_f, *where* A^ν_f *is the result of replacing every function symbol* g *in* A_f *by* g^ν. *We require that the transformation* $f \to f^\nu$ *fixes all* ALV′ *function symbols.*

THEOREM 7. *The following statements are equivalent:*

1. QPV *is conservative over* QALV.
2. QPV *is* Σ^b_1*-conservative over* QALV.
3. QALV *proves its open induction formulas.*
4. ALV′ *proves* $\mathbf{P} = \mathbf{NC}^1$.

Proof that $1 \Longrightarrow 2$: Immediate. □

Proof that $2 \Longrightarrow 3$: Since QALV contains $\#$, all function symbols of QALV are provably bounded by terms in QALV. In particular, QPV proves that the function G introduced in the proof of Theorem 5 is bounded by some QALV term t. Therefore QPV proves the Σ^b_1-formula

$$(A(0) \wedge \neg A(x)) \supset \exists z \leq t(A(z) \wedge \neg A(z+1)),$$

from which the induction formula for A follows. □

Proof that $4 \Longrightarrow 1$: Note that a sufficient set of axioms for QPV consists of the defining axioms for the PV function symbols, Axioms **Q1** and **Q2**, the open P-induction formulas, and the QALV theorem (6). Thus under the assumption 4 we can interpret QPV into QALV by replacing each function symbol f by f^ν. This translates the defining axioms for each PV function symbol into QALV theorems by assumption. Axioms **Q1** and **Q2** and the theorem (6) are fixed under the translation, and are theorems of QALV. The open P-induction formulas remain open P-induction formulas under the translation, and are therefore theorems of QALV by Theorem 4. Thus theorems of QPV are translated into theorems of QALV. Since QALV formulas are fixed under the interpretation, those that are theorems of QPV must also be theorems of QALV. □

Proof that $3 \Longrightarrow 4$: We use the fact that a certain version of the circuit value problem is complete for P. We consider boolean circuits C with k nodes, including m input nodes and n output nodes. We identify the nodes with the numbers $\{0, 1, ..., k-1\}$. Each node is either an *input node* or a *gate*. Each gate has one of the types AND, OR (each of fan-in two), NOT (of fan-in one), or constants 0,1 (of fan-in 0). Some gates are designated *output gates*. We assume that the nodes are reverse topologically numbered, and that nodes $0, 1, ..., n-1$ are output gates, nodes $k-m, k-m+1, ..., k-1$ are input nodes, and each gate i is less than its input node(s).

An *assignment* for such a circuit C is a map t which assigns a value $t(i)$ in {0,1} to each node i. Similarly an *input assignment* to C is a map u which assigns 0 or 1 to each input node of C, and an *output assignment* is a map v which assigns 0 or 1 to each output gate of C. For each circuit C and input assignment u there is a unique correct assignment t and correct output assignment v, defined in the usual way.

We say that assignment t is *locally correct* at node i for circuit C and input assignment u if either i is an input node and $t(i) = u(i)$, or i is a gate and $t(i)$ is correct relative to the value(s) given by t to the input node(s) of i. Thus t is correct for C and u iff t is locally correct at all nodes of C.

We assume a suitable coding of integers for assignments and circuits via binary notation. Explicitly, we code an assignment t by the integer

$$\#(t) = \sum_{i=0}^{k-1} t(i)2^i.$$

We let t^z denote the assignment coded by z, so $t^z(i) = bit(z, i)$. Similarly u^y and v^w denote the input assignment and output assignment coded by y and w, respectively, where y is normalized to the right, so

$$y = \#(u^y) = \sum_{i=0}^{m-1} u(i+k-m)2^i.$$

We denote by C^x the circuit coded by x, where any reasonable encoding will do. Also $k(x)$, $m(x)$, and $n(x)$ are the numbers of nodes, input nodes, and output gates, respectively, of circuit C^x.

We let $t^{x,y}$ denote the correct assignment for the circuit C^x with input assignment u^y. Notice that the function $ASSIGN(x, y) = t^{x,y}(0)$ is complete for FP, and therefore is not an $\mathbf{NC}^1$ function, unless $\mathbf{P} = \mathbf{NC}^1$.

We now give Σ^b_0 formulas for certain relations about circuits. By Corollary 2, each relation is in $\mathbf{NC}^1$.

- $lc(x, y, z, i)$ holds iff assignment t^z is locally correct at node i for circuit C^x and input assignment u^y. (The explicit formula depends on the choice of encoding x for C^x.)
- $corr(x, y, z)$ holds iff $z = \#(t^{x,y})$; i.e. z codes the correct assignment for circuit C^x and input assignment u^y.

$$corr(x, y, z) \equiv |z| \leq k(x) \wedge \forall i < k(x) lc(x, y, z, i)$$

- $lesscorr(x, y, z)$ holds iff $z \leq \#(t^{x,y})$.

$$lesscorr(x, y, z) \equiv corr(x, y, z) \vee [|z| \leq k(x) \wedge$$
$$\exists i < k(x)[bit(z, i) = 0 \wedge \neg lc(x, y, z, i) \wedge \forall j < k(x)(i < j \supset lc(x, y, z, j))]]$$

By the assumption 3, QALV proves its open induction formulas, and thus by Theorem 3, QALV proves the induction formula for $lesscorr(x, y, z)$, with induction on z. Rearranging this formula we have QALV proves

$$(lesscorr(x, y, 0) \wedge \neg lesscorr(x, y, z)) \supset$$
$$\exists w(lesscorr(x, y, w) \wedge \neg lesscorr(x, y, w+1)).$$

Let $g(x) = 2^{k(x)}$, so $g(x)$ is one more than the number of the assignment which assigns 1's to all nodes in the circuit C^x. Thus $\neg lesscorr(x, y, g(x))$ holds, and for a suitable choice of QALV symbol g, QALV proves $\neg lesscorr(x, y, g(x))$. Since also QALV proves $lesscorr(x, y, 0)$, it follows that QALV proves $\exists w(lesscorr(x, y, w) \wedge \neg lesscorr(x, y, w+1))$. Thus by Theorem 6 there is a QALV function symbol F such that

$$\text{QALV} \vdash lesscorr(x, y, F(x, y)) \wedge \neg lesscorr(x, y, F(x, y)+1). \tag{7}$$

If z codes the correct assignment $t^{x,y}$ then by (7) we have $F(x,y) \leq z$ and not $F(x,y)+1 \leq z$, so $F(x,y) = z$. We will reach the same conclusion below by an argument which does not presuppose the existence of z and which can be formalized in **QALV**, so that

$$\mathbf{QALV} \vdash corr(x, y, F(x,y)). \tag{8}$$

In fact, setting $w = F(x,y)$, if $lesscorr(x,y,w)$ but $\neg corr(x,y,w)$ then there is a bit position i such that bits in w to the left of i are locally correct but the i-th bit of w is 0 instead of the correct value of 1. If there is a 0 bit to the right of i in w, then position i plays the same role in w and $w+1$, so that $lesscorr(x,y,w+1)$. If on the other hand all bits to the right of i in w are 1, then bit i of $w+1$ is 1 and all bits to the right of i are 0. If all these revised bits are locally correct, then $corr(x,y,w+1)$, and otherwise there is a leftmost incorrect position i', and so $lesscorr(x,y,w+1)$. Thus the fact $\neg lesscorr(x,y,w+1)$ from (7) leads to a contradiction in all cases, so we conclude $corr(x,y,w)$.

Notice that F is complete for **FP**, so it follows that $\mathbf{P} = \mathbf{NC}^1$. Our goal now is to show that **ALV′** proves $\mathbf{P} = \mathbf{NC}^1$ in the sense of definition 2. By Theorem 2 it suffices to show that **QALV** proves $\mathbf{P} = \mathbf{NC}^1$.

Every **QPV** function f is polytime and therefore is computed by some uniform polynomial size family of boolean circuits. We will define a **QALV** function symbol $circ_f$ such that $circ_f(M)$ codes a circuit which computes f when the length of each of its number arguments is at most $|M|$. Also we will define **QALV** functions in_r and out such that $in_r(M,\vec{x})$ codes the input assignment to a circuit computing a function of arity r with input $\vec{x} = (x_1, ..., x_r)$ with each argument at most $|M|$ bits, and $out(N,z)$ codes the output assignment of a circuit with $|N|$ output nodes and assignment t^z. Thus

$$out(N,z) = z \bmod 2^{|N|} \tag{9}$$

and when the arity $r = 1$

$$in_1(M,x) = x \bmod 2^{|M|}. \tag{10}$$

For each arity r we define the **QALV** input size function M_r by

$$M_r(\vec{x}) = tr(2^{\max(|x_1|,...,|x_r|)}). \tag{11}$$

For each **PV** function symbol f we define the three **ALV′** function symbols f^ν, $circ_f$, and the bound N_f, together by induction on the structure of f. Then $circ_f(M)$ represents a circuit with $|N_f(M)|$ output bits. We show that **QALV** proves the transformed defining equations for f, as required by definition 2, and also show that **QALV** proves the length monotonicity of N_f:

$$|M| \leq |M'| \supset |N_f(M)| \leq |N_f(M')|, \tag{12}$$

and the bounding property of N_f:

$$\bigwedge_{i=1}^{r} |x_i| \leq |M| \supset |f^\nu(\vec{x})| \leq |N_f(M)|, \tag{13}$$

and finally **QALV** proves the correctness of the circuit:

$$\bigwedge_{i=1}^{r} |x_i| \leq |M| \supset f^\nu(\vec{x}) = out(N_f(M), F(circ_f(M), in_r(M,\vec{x}))). \tag{14}$$

Since also **QALV** proves $|x_i| \leq |M_r(\vec{x})|$, $1 \leq i \leq r$, it follows that **QALV** proves

$$f^\nu(\vec{x}) = out(N_f(M_r(\vec{x})), F(circ_f(M_r(\vec{x})), in_r(M_r(\vec{x}), \vec{x}))). \tag{15}$$

For the base case of the induction, f is an initial function symbol of **PV**. Then f is also an initial function symbol of **ALV′**, so we take $f^\nu = f$. The defining equations for f are the same in **ALV′** and **PV**, and the transformation ν fixes these equations, so the transformed defining equations are theorems of **QALV**. The bound N_f is straightforward to define, and the circuit function $circ_f$ is either trivial, or defined using the recursive construction described below.

For the first case of the induction step, $f = [\lambda\vec{x}.t]$, where ν has been defined on the function symbols in the term t. Then we set $f^\nu = [\lambda\vec{x}.t^\nu]$, where t^ν results from t by replacing each function symbol g in t by g^ν. (If f is already an **ALV′** function symbol, then t^ν is t and this identity already holds.) The transformed **PV** defining equation for f is $f^\nu(\vec{x}) = t^\nu$, which is the defining equation for f^ν in **ALV′**. Hence the condition for definition 2 is satisfied.

We describe the functions $circ_f$ and N_f for the special case in which t is $g(h(x))$, so $f^\nu = [\lambda x.g^\nu(h^\nu(x))]$. The general case is shown by induction on the structure of t. When t is $g(h(x))$, we set

$$N_f(M) = N_g(N_h(M)). \tag{16}$$

Then (12) and (13) follow from (12) and (13) with f replaced first by h and then by g, which hold by the induction hypothesis.

To define $circ_f$, we use C^f, C^g, and C^h to denote the circuits represented by $circ_f(M)$, $circ_g(N_h(M))$, and $circ_h(M)$, respectively (see Figure 1). Then C^f is obtained by putting together C^g and C_1^h and identifying the output nodes of C_1^h with the input nodes of C^g, where C_1^h is obtained from C^h by adding $|K^g| - |N_h(M)|$ to each node number, where $|K^g|$ is the number of nodes in C^g.

To establish (14) we use the defining equation $f^\nu(\vec{x}) = g^\nu(h^\nu(\vec{x}))$ for f^ν, and also (14) with f replaced first by h and then by g, and (13) with f replaced by h. These hold by the induction hypothesis. Since the nodes of C_g have been uniformly renumbered, we also need the fact that F behaves similarly on a circuit and its renumbered version. This is shown by the correctness (8) of F.

For the remaining case, $f = R[g,h,k]$, and f is defined from g, h by recursion on notation, with bounding function k. The defining equation for f is

$$f(x,\vec{y}) = cond(x, g(\vec{y}), cond(trunc(t, k(x,\vec{y})), t, k(x,\vec{y}))), \tag{17}$$

where $t \equiv h(x,\vec{y}, f(tr(x),\vec{y}))$.

We introduce three auxiliary functions FF, H and K by the defining equations

$$H(x,\vec{y},z) = cond(x, g(\vec{y}), h(x,\vec{y},z)), \tag{18}$$

$$K(x,\vec{y}) = cond(x, g(\vec{y}), k(x,\vec{y})), \tag{19}$$

and

$$FF(x,\vec{y},z) = cond(trunc(H(x,\vec{y},z), K(x,\vec{y})), H(x,\vec{y},z), K(x,\vec{y}))). \tag{20}$$

If follows immediately from (17) and these definitions that

$$f(x,\vec{y}) = FF(x,\vec{y}, f(tr(x),\vec{y})). \tag{21}$$

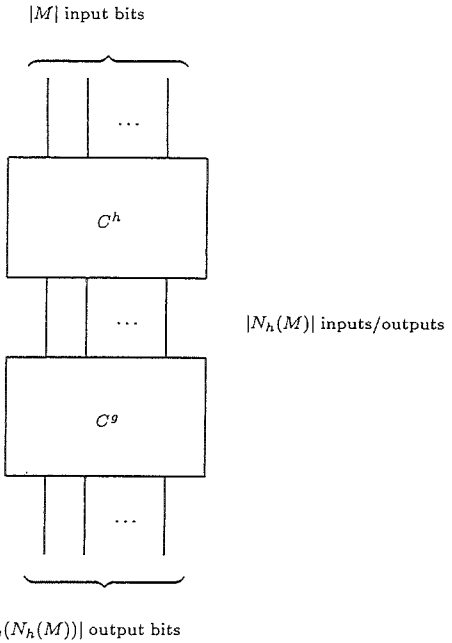


FIGURE 1. Circuit C^f for $f(x) = g(h(x))$.

We may assume that the induction hypothesis applies to FF, H, and K by the previous case in our induction. Thus let

$$N_f(M) = N_K(M). \tag{22}$$

We base the definition of the circuit $C^f = circ_f(M)$ on the recursion (21). Thus let

$$M' = \max(M, N_K(M)) \tag{23}$$

and

$$C^{FF} = circ_{FF}(M') \tag{24}$$

Then C_f consists of a stack of $|M| + 1$ copies of C^{FF}, with inputs and outputs suitably directed and nodes suitably renumbered (see Figure 2). We designate these copies $C^{FF}_{|M|}, C^{FF}_{|M|-1}, ..., C^{FF}_0$. The inputs to C^f are $(x, \vec{y})$ and the inputs to C^{FF} are $(x', \vec{y}, z')$. for suitable x' and y'. The inputs $(x', \vec{y}, z')$ of the top copy $C^{FF}_{|M|}$ are set to $(0, \vec{y}, 0)$. For $0 \leq j < |M| - 1$ the inputs to copy C^{FF}_j are $(\lfloor x/2^j \rfloor, \vec{y}, w)$ where w is the output from the copy above. The output to C^f is the output to copy C^{FF}_0, padded with zeroes or chopped to exactly $|N_K(M)|$ bits.

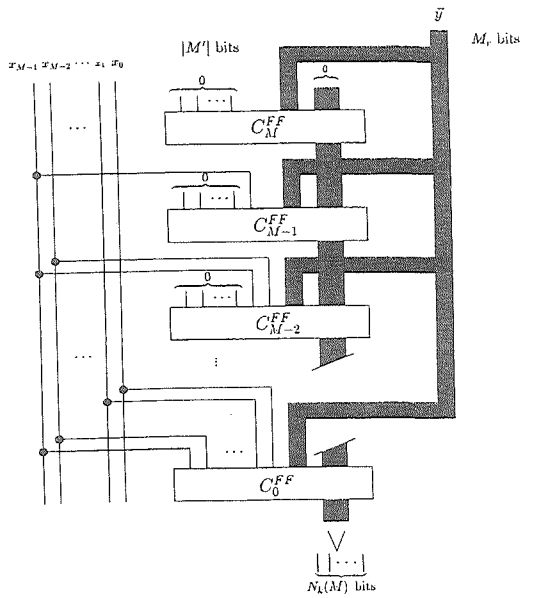


FIGURE 2. C^f for $f(x, \vec{y}) = FF(x, \vec{y}, f(t_r(x), \vec{y}))$.

Each input number for C^f has $|M|$ bits, but each input number for C^{FF} has $|M'|$ bits. Because $|M'|$ may exceed $|M|$, the high order bits of some input arguments of the C^{FF} copies may have to be set to 0. (In Figure 2 we assume $M = M'$.) Also, each copy of C^{FF} has $|N_{FF}(M')|$ output bits. If $|N_{FF}(M')| > |M'|$, then only the $|M'|$ low-order output bits of C^{FF} are connected to he z-input of the copy below. (The discarded output bits are always 0.)

We now calculate the total number k_f of nodes in C^f and the amount added to each node number of each copy of C^{FF}. Assuming that $\vec{y}$ has arity r, the total number of input bits to C^f to code the input $(x, \vec{y})$ is $m_f = |M|(r + 1)$. By convention, these bits have node numbers $k_f - m_f, k_f - m_f + 1, ..., k_f - 1$. For simplicity, we mandate that for each copy of C^{FF}, each input bit set to 0 is identified with a single 0 gate, numbered $k_f - m_f - 1$. Thus each of the $m_{FF} = |M'|(r + 2)$ input bits to each copy of C^{FF} is identified with this 0 gate, or with an input to C^f, or with an output to the copy of C^{FF} above. Also the bottom copy C^{FF}_0 must have $B = \max(|N_{FF}(M)| - |N_K(M)|, 0)$ constant 0 output gates inserted to pad its output to $|N_K(M)|$ bits. Therefore $k_f = B + (|M| + 1)(k_{FF} - m_{FF}) + m_f + 1$, where k_{FF} is the number of nodes in C^{FF}. For $0 \leq j \leq |M|$, copy C^{FF}_j has each of its non-input node numbers incremented by $B + j(k_{FF} - m_{FF})$, except that the output gates of the bottom copy C^{FF}_0 are unchanged.

This completes our description of the circuit function $circ_f(M)$. We point out that the length multiplications needed to renumber gates are readily implemented in **QALV** using the # function.

If f is not already an $\mathbf{ALV}'$ function symbol, we define

$$f^\nu = [\lambda x \vec{y}.T], \tag{25}$$

where T is the right hand side of (15) (with $(\vec{x})$ replaced by $(x, \vec{y})$).

If f is already an $\mathbf{ALV}'$ function symbol, then we must take $f^\nu = f$. In this case $f = f^\pi$ in the interpretation defined before Theorem 1, and so f is defined by concatenation recursion on notation or K-bounded recursion on notation, and its defining equations in $\mathbf{ALV}'$ and **PV** are similar and inter-provable in **QALV**. It will follow from the argument below that **QALV** proves $f(x, \vec{y}) = T$, where T is as in (25). Therefore we will not further consider this case separately.

Monotonicity (12) of N_f follows immediately from (22) and the induction hypothesis (12) for N_K. To prove (13), we can show in **QALV**

$$\bigwedge_{i=1}^{r} |x_i| \leq |M| \supset |M_r(\vec{x})| \leq |M|.$$

Then (13) follows easily from (15), (12), and the simple fact $|out(N, z)| \leq |N|$ from (9).

To show that **QALV** proves the circuit correctness (14) we must compare the results of two circuits for f: $C^f = circ_f(M)$ and the circuit $\hat{C}^f = circ_f(M_{r+1}(x, \vec{y}))$ which computes f^ν according to (15). For this we need to know that the copies of C^{FF} in the two circuits each correctly compute FF^ν, and we need properties of FF^ν. We can use the transformed defining equations (18^ν), (19^ν), and (20^ν) for H^ν, K^ν, and FF^ν, which follow from (18), (19), and (20) by the induction hypothesis. It follows easily from the transformed equations that

$$|FF^\nu(x, \vec{y}, z)| \leq |K^\nu(x, \vec{y})|. \tag{26}$$

The correctness of each copy of C^{FF} follows from (14), with f replaced by FF, which holds by the induction hypothesis. We also need to know that the outputs of these circuits are not too big, since we only use the $|M'|$ low-order bits of the output as inputs to the copy below. Since we are trying to prove (14) for f, we may assume that each of the argument $(x, y_1, ..., y_r)$ is bounded in length by $|M|$, so by (26) we have

$$|FF^\nu(\lfloor x/2^j \rfloor, \vec{y}, z)| \leq |N_K(M)|,$$

where we have used the bound (13) with f replaced by K. This bound suffices, since we are allowing $|M'| \geq |N_K(M)|$ input bits for each number input to C^{FF} in C^f. A similar argument applies to $\hat{C}^f$, with M replaced by $M_r(\vec{x})$.

Now from the correctness of the circuits C^{FF}, we can show that among the bottom $|x|$ copies of C^{FF} in the two circuits C^f and $\hat{C}^f$, corresponding copies compute the same values. From the defining equations, we conclude that $FF^\nu(0, \vec{y}, z) = g^\nu(\vec{y})$, so the output of copy $C^{FF}_{|x|}$ in the two circuits is the same; namely $g^\nu(\vec{y})$ (see Figure 2). Proceeding by P-induction on w for $|w| \leq |x|$, the output of copy $C^{FF}_{|trunc(x,w)|}$ in C^f represents the same number as the output of copy $\hat{C}^{FF}_{|trunc(x,w)|}$ in $\hat{C}^f$. In particular, for $w = x$, we conclude that the outputs of the two circuits C^f and $\hat{C}^f$ are the same. This established (14).

It remains to show that **QALV** proves the transformed defining equation (17^ν) for f^ν. This is easily equivalent to establishing (21^ν) in **QALV**. From (14) we see that the circuit $C^f = circ_f(M)$, with $M = M_{r+1}(x, \vec{y})$, correctly computes both $f^\nu(x, \vec{y})$ and $f^\nu(tr(x), \vec{y})$ (see Figure 2). Arguing as above, we can compare the outputs of the bottom $|x|$ copies of C^{FF} in C^f with input $(x, \vec{y})$ and input $(tr(x), \vec{y})$. We see by P-induction that for $0 \leq j \leq |tr(x)|$, the output of copy C^{FF}_j for C^f-input $(tr(x), \vec{y})$ is the same as the output of copy C^{FF}_{j+1} for C^f-input $(x, \vec{y})$. In particular, for $j = 0$, the output of C^f with input $(tr(x), \vec{y})$ (namely $f^\nu(tr(x), \vec{y})$) is the same as the output of the second-from-bottom copy C^{FF}_1 of C^f with input $(x, \vec{y})$. Application of the bottom copy C^{FF}_0 gives $f^\nu(x, \vec{y})$, and (21^ν) follows. □

4. Discussion

As mentioned in the Introduction, one motivation for this paper is to relate the question $\mathbf{P} = \mathbf{NC}^1$ to the question of whether Frege systems can p-simulate extended Frege systems. It is natural to try to find combinatorial principles which separate Frege and extended Frege systems, based on the perception that their proofs require polynomial time concepts which are not in $\mathbf{NC}^1$. Analogous principles are known for the problem of separating Frege and constant-depth Frege systems. For example, the pigeonhole principle, formulated as a sequence of tautologies, has polynomial size Frege proofs but requires exponential size constant-depth Frege proofs [**Kra95a**].

In [**BBP95**], Bonet, Buss, and Pitassi suggest several combinatorial principles which may separate Frege and extended Frege systems. Some of these, including the Odd-town theorem, seem to require linear algebra for their proofs. Linear algebra proofs usually depend on some form of Gaussian elimination, which is not known to be expressible using $\mathbf{NC}^1$ functions. Thus no one knows how to formalize the proofs of these combinatorial principles in $\mathbf{ALV}'$ (or **AID** or $\mathbf{T}_1$), although they are readily formalized in **PV**. Thus we have no techniques for constructing polynomial size proofs of the corresponding tautologies.

Incidently, a principle whose subject matter is linear algebra is the following: If A and B are $n \times n$ matrices over GF(2) such that $AB = I$, then $BA = I$. The proof seems to require Gaussian elimination and can be formalized in **PV**, but, as far as we know, not in $\mathbf{ALV}'$.

Paraphrasing the Assertion in the Introduction, it seems likely that one can show that **PV** is conservative over $\mathbf{ALV}'$ iff $\mathbf{ALV}'$ proves that Frege systems p-simulate extended Frege systems. It follows from the easy part of Theorem 7 that if $\mathbf{ALV}'$ proves that $\mathbf{P} = \mathbf{NC}^1$ then **PV** is conservative over $\mathbf{ALV}'$. A major problem is to strengthen Theorem 7 and show a converse: If **PV** is conservative over $\mathbf{ALV}'$ (i.e. $\mathbf{ALV}'$ proves that Frege systems p-simulate extended Frege systems) then $\mathbf{P} = \mathbf{NC}^1$.

Appendix A. Axioms and Rules of QALV and QPV

A.1. Axioms for PV.

A1. $s_0(0) = 0$
A2. $parity(s_0(x)) = 0$
A3. $parity(s_1(x)) = 1$ $(= s_1(0))$
A4. $tr(s_0(x)) = x$
A5. $tr(s_1(x)) = x$

A6. $cond(0,y,z) = y$
A7. $cond(s_0(x),y,z) = cond(x,y,z)$
A8. $cond(s_1(x),y,z) = z$
A9. $pad(x,s_0(y)) = cond(y,x,s_0(pad(x,y))$
A10. $pad(x,s_1(y)) = s_0(pad(x,y))$
A11. $trunc(x,s_0(y)) = cond(y,x,tr(trunc(x,y)))$
A12. $trunc(x,s_1(y)) = tr(trunc(x,y))$
A13. $x\#s_0(y) = cond(y,1,pad(x\#y,x))$
A14. $x\#s_1(y) = pad(x\#y,x)$
A15. $[\lambda x_1 ... x_n.t](x_1,...,x_n) = t$

BRN. $R[g,h,k](x,\vec{y}) = cond(x,g(\vec{y}),cond(trunc(t,k(x,\vec{y})),t,k(x,\vec{y})))$, where
$t \equiv h(x,\vec{y},R[g,h,k](tr(x),\vec{y}))$

A.2. Axioms for ALV′.

A1-A15, and also
A16. $s(0) = 1$
A17. $s(s_0(x)) = s_1(x)$
A18. $s(s_1(x)) = s_0(s(x))$
A19. $|0| = 0$
A20. $|s_0(x)| = cond(x,0,s(|x|))$
A21. $|s_1(x)| = s(|x|)$

CRN. $R^{crn}[g,h,k](x,\vec{y}) = cond(x,g(\vec{y}),cond(parity(x),U,V))$, where
$U = cond(h(tr(x),\vec{y}),s_0(t),s_1(t))$ and
$V = cond(k(tr(x),\vec{y}),s_0(t),s_1(t))$ and
$t = R^{crn}[g,h,k](tr(x),\vec{y})$.
KBRN. $R_\ell[g,h,k](x,\vec{y}) = cond(x,g(\vec{y}),cond(parity(x),U_\ell(x,\vec{y}),V_\ell(x,\vec{y})))$, where
$U_\ell(x,\vec{y}) = h(tr(x),\vec{y},R_\ell[g,h,k](tr(x),\vec{y}))$ mod 2^ℓ and
$V_\ell(x,\vec{y}) = k(tr(x),\vec{y},R_\ell[g,h,k](tr(x),\vec{y}))$ mod 2^ℓ.
Here x mod 2^1 is $parity(x)$, x mod 2^2 is
$cond(parity(x),s_0(parity(tr(x))),s_1(parity(tr(x))))$, etc.

A.3. Rules of ALV′ and PV.

R1. $t = u \vdash u = t$
R2. $t = u, u = v, \vdash t = v$
R3. $t_1 = u_1, ..., t_n = u_n \vdash f(t_1,...,t_n) = f(u_1,...,u_n)$
R4. $t = u \vdash t[v/x] = u[v/x]$, x a variable, v any term.
R5. Induction on notation:

$$\frac{\begin{array}{c} t_1[0/x] = t_2[0/x] \\ t_1[s_0(x)/x] = v_0[t_1/a] \quad t_2[s_0(x)/x] = v_0[t_2/a] \\ t_1[s_1(x)/x] = v_1[t_1/a] \quad t_2[s_1(x)/x] = v_1[t_2/a] \end{array}}{t_1 = t_2}$$

Appendix B. Proof of Theorem 2

As mentioned before Theorem 2, the single element universe $\{0\}$ forms a model for **ALV′**, **PV**, and **Q2**, and hence $0 \neq 1$ is not a logical consequence of **Q2** together with the theorems of these theories.

To prove the rest of Theorem 2, we need a simple property of Horn clause sets. Recall that a Horn clause is a disjunction of literals, where each literal is a negated atom, except we allow at most one unnegated atom per clause. The property we need is that every propositionally consistent set of Horn clauses has a unique *minimal model*; that is, a truth assignment τ which satisfies all clauses in the set, and further for each atom P, if P^τ = TRUE, then P^σ = TRUE for every assignment σ which satisfies the set. The minimal model is easily constructed iteratively, starting with the totally FALSE assignment.

THEOREM 8. **Q2** *is not a logical consequence of* **Q1** *and the theorems of* **ALV′**, *and not a logical consequence of* **Q1** *and the theorems of* **PV**.

Proof: We give the argument for **PV**. The argument for **ALV′** is the same.

Let $\mathbf{PV}^0$ denote the set of all theorems of **PV**, together with the set of all substitution instances of the standard equality axioms for the language of **PV**. Then $\mathbf{PV}^0$ is a set of Horn clauses. Suppose that **Q2** is a logical consequence of the theorems of **PV** together with $0 \neq 1$. Then by the Herbrand Theorem, **Q2** without its quantifier, i.e. $tr(x) = 0 \supset (x = 0 \vee x = 1)$, is a truth functional consequence of $\mathbf{PV}^0 \cup \{0 \neq 1\}$. Since $\mathbf{PV}^0 \cup \{0 \neq 1, tr(x) = 0\}$ is a consistent set of Horn clauses, it has has a minimal model τ. Then $(x = 0 \vee x = 1)^\tau$ = TRUE. But $(x = 0)^\tau$ = FALSE and $(x = 1)^\tau$ = FALSE, because τ is minimal and each of the inequations $x \neq 0$ and $x \neq 1$ can be added separately to $\mathbf{PV}^0 \cup \{0 \neq 1, tr(x) = 0\}$ to form a consistent set. This is a contradiction. □

THEOREM 9. **QALV** *is a conservative extension of* **ALV′** *and* **QPV** *is a conservative extension of* **PV**.

Proof: Again we prove this for the case of **PV**. We need certain theorems of **PV**, but the **PV** proofs of these involve only the initial functions 0, s_0, s_1, tr, and $cond$, so these theorems are also theorems of **ALV′**.

Suppose $u = v$ is a logical consequence of the theorems of **PV**, together with **Q1** and **Q2**. Then by the Herbrand Theorem, there is a finite set of substitution instances $A_1, ..., A_n$ of **Q2** such that $u = v$ is a truth functional consequence of $\mathbf{PV}^0 \cup \{0 \neq 1, A_1, ..., A_n\}$, where $\mathbf{PV}^0$ is as defined in the proof of Theorem 8. By Lemma 5 below, $u = v$ is a consequence of $\mathbf{PV}^0$, so $\mathbf{PV} \vdash u = v$. (We use the fact that the equality rules of **PV** are complete for equational first order consequences.) □

For the rest of the proof, **E** denotes an arbitrary set of **PV** equations, and $\models$ denotes propositional logical consequence.

From the defining equations for $cond$, we have that **PV** proves $cond(0,u,v) = u$ and $cond(1,u,v) = v$, and therefore

$$\mathbf{PV}^0, 0 = 1 \models u = v \tag{27}$$

for any **PV** terms u and v. Thus we can prove

LEMMA 1. *If* $\mathbf{PV}^0, \mathbf{E}, 0 \neq 1 \models u = v$, *then* $\mathbf{PV}^0, \mathbf{E} \models u = v$ *for any* **PV** *terms* u *and* v.

Proof: If $\mathbf{PV}^0 \cup \mathbf{E}$ is inconsistent the result is obvious. Otherwise let τ be the minimal model for $\mathbf{PV}^0 \cup \mathbf{E}$. If $(0 = 1)^\tau$ = FALSE, then by hypothesis $(u = v)^\tau$ = TRUE, so $(u = v)^\sigma$ = TRUE for every assignment σ satisfying $\mathbf{PV}^0 \cup \mathbf{E}$. If $(0 = 1)^\tau$ = TRUE then $\mathbf{PV}^0 \cup \mathbf{E} \models 0 = 1$, so the result follows from (27). □

LEMMA 2. *If* $\mathbf{PV}^0, \mathbf{E}, t = 0 \models u = v$ *then* $\mathbf{PV}^0, \mathbf{E} \models cond(t, u, v) = v$, *for any* PV *terms* t, u, *and* v.

Proof: If $\mathbf{PV}^0, \mathbf{E}, t = 0 \models u = v$ then $u = v$ can be derived from theorems of PV E, and $t = 0$ by repeated use of the equality rules **R1**, **R2**, and **R3**. The Lemma is proved by induction on the length of this derivation. The first part of the base case is when $u = v$ is either a theorem of PV or in E. By notational induction on x, PV proves $cond(x, y, y) = y$, and hence PV proves $cond(t, v, v) = v$. By the equality axioms and $u = v$, we conclude $\mathbf{PV}^0, \mathbf{E} \models cond(t, u, v) = v$. The second part of the base case is when $u = v$ is $t = 0$. By notational induction on x, PV proves $cond(x, x, 0) = 0$ so PV proves $cond(t, t, 0) = 0$.

For the first part of the induction step, suppose $u = v$ results from the application of rule **R1** to $v = u$. By the induction hypothesis, $\mathbf{PV}^0, \mathbf{E} \models cond(t, v, u) = u$. Using appropriate substitution instances of the PV theorems $cond(x, y, y) = y$ and

$$cond(x, cond(x, y, z), w) = cond(x, y, w)$$

(see T4, p123 of [**CU93**]), by the equality axioms we conclude

$$\mathbf{PV}^0, \mathbf{E} \models cond(t, u, v) = cond(t, cond(t, v, u), v) \tag{28}$$

$$= cond(t, v, v) = v. \tag{29}$$

The other two rules are handled similarly. (The reasoning is similar to the justification of **DR13** and **DR14** of [**CU93**].) □

LEMMA 3. *If* $\mathbf{PV}^0, \mathbf{E}, t = 1 \models u = v$ *then* $\mathbf{PV}^0, \mathbf{E}, tr(t) = 0 \models cond(t, u, v) = u$, *for any* PV *terms* t, u, *and* v.

Proof: As in the proof of the above lemma, we proceed by induction on the derivation of $u = v$ via **R1**, **R2**, and **R3**. The induction step is the same as the justification of **DR12**, **DR13**, and **DR14** in [**CU93**]. The interesting part of the base case is when $u = v$ is $t = 1$. For this we need the PV theorem

$$cond(tr(x), cond(x, 1, x), 1) = 1. \tag{30}$$

This can be proven by Conditional Induction on x (**DR17** of [**CU93**]), with a secondary induction to take care of the induction step $s_1(x)/x$. From (30) we conclude $\mathbf{PV}^0, tr(t) = 0 \models cond(t, 1, t) = 1$, as required. □

LEMMA 4. *If* $\mathbf{PV}^0, \mathbf{E} \models tr(t) = 0$ *and* $\mathbf{PV}^0, \mathbf{E}, t = 0 \models u = v$ *and* $\mathbf{PV}^0, \mathbf{E}, t = 1 \models u = v$, *then* $\mathbf{PV}^0, \mathbf{E} \models u = v$.

Proof: Immediate from the two preceding lemmas. □

LEMMA 5. *Let* $A_1, ..., A_n$ *be* n *substitution instances of* **Q2**. *If*

$$\mathbf{PV}^0, \mathbf{E}, 0 \neq 1, A_1, ..., A_n \models u = v, \tag{31}$$

then $\mathbf{PV}^0, \mathbf{E} \models u = v$.

Proof: Induction on n. If $\mathbf{PV}^0 \cup \mathbf{E} \cup \{0 \neq 1\}$ is inconsistent, we are done by Lemma 1, so let τ be the minimal model for $\mathbf{PV}^0 \cup \mathbf{E} \cup \{0 \neq 1\}$. Let A_i be $tr_i(t_i) = 0 \supset (t_i = 0 \vee t_i = 1)$, for $i = 1, ..., n$, and assume the hypothesis (31). If $(tr(t_i) = 0)^\tau = \text{FALSE}$ for all i, then $(u = v)^\tau = \text{TRUE}$, and we are done by Lemma 1. Otherwise we may assume $(tr(t_1) = 0)^\tau = \text{TRUE}$, so $\mathbf{PV}^0, \mathbf{E} \models tr(t_1) = 0$ by Lemma 1. Also, by (31),

$$\mathbf{PV}^0, \mathbf{E}, t_1 = 0, 0 \neq 1, A_2, ..., A_n \models u = v$$

so $\mathbf{PV}^0, \mathbf{E}, t_1 = 0 \models u = v$ by the induction hypothesis. Similarly $\mathbf{PV}^0, \mathbf{E}, t_1 = 1 \models u = v$. Therefore $\mathbf{PV}^0, \mathbf{E} \models u = v$ by Lemma 4. □

Acknowledgment: I thank Sam Buss and especially Jan Krajíček for their illuminating comments on the first version of this paper.

References

[Ara91] T. Arai, *Frege systems, ALOGTIME and bounded arithmetic*, 1991, Manuscript.

[Bar89] D. A. Barrington, *Bounded-width polynomial-time branching programs recognize exactly those languages in NC^1*, J. Comp. Systems Sci. **38** (1989), no. 1, 150–164.

[BBP95] M. L. Bonet, S. R. Buss, and T. Pitassi, *Are there hard examples for Frege systems?*, Feasible Mathematics II (P. Clote and J.B. Remmel, eds.), Birkhauser, 1995, pp. 30–56.

[BIS90] D. M. Barrington, N. Immerman, and H. Straubing, *On uniformity in NC^1*, J. Comp. Systems Sci. **41** (1990), no. 3, 274–306.

[Bus86] S. R. Buss, *Bounded Arithmetic*, 1986, Naples, Bibliopolis. (Revision of 1985 Princeton University Ph.D. thesis).

[Bus91] S. R. Buss, *Propositional consistency proofs*, Annals of Pure and Applied Logic **52** (1991), 3–29.

[Bus95] S. R. Buss, *Relating the bounded arithmetic and polynomial-time hierarchies*, Annals of Pure and Applied Logic **75** (1995), 67–77.

[Clo90] P. Clote, *Alogtime and a conjecture of s.a. cook*, Proceedings of IEEE Symposium on Logic in Computer Science, 1990.

[Clo93] P. Clote, *On polynomial size Frege proofs of certain combinatiorial principles*, Arithmetic, Proof Theory, and Computational Complexity (P. Clote and J. Krajiček, eds.), Oxford, 1993, pp. 162–184.

[Cob65] A. Cobham, *The intrinsic computational difficulty of functions*, Proceedings of the International Congress Logic, Methodology, and Philosophy of Science (Y. Bar-Hillel, ed.), North Holland, 1965, pp. 24–30.

[Coo75] S. A. Cook, *Feasibly constructive proofs and the propositional calculus*, Proceedings of the 7th Annual ACM Symposium on the Theory of Computing, 1975, pp. 83–97.

[CR79] S. A. Cook and R. A. Reckhow, *The relative efficiency of propositional proof systems*, J. Symbolic Logic **44** (1979), no. 1, 36–50.

[CU93] S. A. Cook and A. Urquhart, *Functional interpretations of feasibly constructive arithmetic*, Annals of Pure and Applied Logic **63** (1993), 103–200.

[KPT91] J. Krajiček, P. Pudlak, and G. Takeuti, *Bounded arithmetic and the polynomial hierarchy*, Annals of Pure and Applied Logic **52** (1991), 143–153.

[Kra95a] J. Krajiček, *Bounded arithmetic, propositional logic, and complexity theory*, Cambridge University Press, 1995.

[Kra95b] J. Krajíček, *On Frege and extended Frege proof systems*, Feasible Mathematics II (P. Clote and J. Remmel, eds.), Birkhauser, 1995, pp. 284–319.

[Pit96] F. Pitt, *A quantifier-free theory based on a string algebra for NC^1*, Submitted, 1996.

[Ruz81] W. L. Ruzzo, *On uniform circuit complexity*, J. Comp. and Systems Sci. **22** (1981), no. 3, 365–383.

[Zam96] D. Zambella, *Notes on polynomially bounded arithmetic*, J. Symbolic Logic **61** (1996), no. 3, 942–966.

DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF TORONTO, TORONTO, ONTARIO, CANADA M5S 3G4

E-mail address: sacook@cs.toronto.edu

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

On PHP, st-connectivity, and odd charged graphs

Peter Clote and Anton Setzer

1. Introduction

As witnessed by this proceedings, there is currently much interest in the analysis of proof size for tautology families in certain proof systems for propositional logic. The reason is twofold: (1) the analysis of proof systems leads to a better understanding of efficiency issues for theorem provers and (2) the development of new combinatorial methods in establishing proof size lower bounds for propositional proof systems may help in better understanding and solving difficult problems in complexity theory (it is well-known that $NP = co-NP$ if and only if there is a sound, complete propositional proof system which furnishes polynomial size proofs for all tautologies).

In this paper, we consider several proof systems (resolution, cutting planes, and a multiplicative extension of cutting planes), and analyze the proof size of certain combinatorial statements related to the pigeonhole principle and to graph theoretic principles.

The well-known pigeonhole principle PHP_k is given by

$$\bigwedge_{0\le i\le k}\bigvee_{0\le j<k} p_{i,j} \to \bigvee_{0\le i<i'\le k}\bigvee_{0\le j<k}(p_{i,j}\wedge p_{i',j}).$$

In [7], W. Degen gave a natural generalization of the pigeonhole principle, which states that for positive integers m, k if f is a function mapping $\{0,\dots,m\cdot k\}$ into $\{0,\dots,k-1\}$ then there is $j<k$ for which $f^{-1}(j)$ has size greater than m. For ease of notation, non-negative integers will be considered as von Neumann ordinals, so that $n=\{0,\dots,n-1\}$, and $[m]^n$ denotes the set of size n subsets of $\{0,\dots,m-1\}$. Formulated in propositional logic, Degen's generalization is given by a family $\{D_{m,k} : m,k\in \mathrm{N}-\{0\}\}$ where $D_{m,k}$ is

$$\bigwedge_{0\le i\le m\cdot k}\bigvee_{0\le j<k} p_{i,j} \to \bigvee_{0\le j<k}\bigvee_{I\in[m\cdot k+1]^{m+1}}\bigwedge_{i\in I} p_{i,j}.$$

1991 *Mathematics Subject Classification.* Primary 03B05; Secondary 68Q15, 03F03.

Part of this research supported by NSF CCR-9408090, US-Czech Science and Technology Grant 93-025, and the Volkswagen Stiftung.

Part of this research supported by the Volkswagen Stiftung.

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

On PHP, st-connectivity, and odd charged graphs

Peter Clote and Anton Setzer

1. Introduction

As witnessed by this proceedings, there is currently much interest in the analysis of proof size for tautology families in certain proof systems for propositional logic. The reason is twofold: (1) the analysis of proof systems leads to a better understanding of efficiency issues for theorem provers and (2) the development of new combinatorial methods in establishing proof size lower bounds for propositional proof systems may help in better understanding and solving difficult problems in complexity theory (it is well-known that $NP = co-NP$ if and only if there is a sound, complete propositional proof system which furnishes polynomial size proofs for all tautologies).

In this paper, we consider several proof systems (resolution, cutting planes, and a multiplicative extension of cutting planes), and analyze the proof size of certain combinatorial statements related to the pigeonhole principle and to graph theoretic principles.

The well-known pigeonhole principle PHP_k is given by

$$\bigwedge_{0\le i\le k}\bigvee_{0\le j<k} p_{i,j} \to \bigvee_{0\le i<i'\le k}\bigvee_{0\le j<k}(p_{i,j}\wedge p_{i',j}).$$

In [7], W. Degen gave a natural generalization of the pigeonhole principle, which states that for positive integers m, k if f is a function mapping $\{0,\dots,m\cdot k\}$ into $\{0,\dots,k-1\}$ then there is $j<k$ for which $f^{-1}(j)$ has size greater than m. For ease of notation, non-negative integers will be considered as von Neumann ordinals, so that $n=\{0,\dots,n-1\}$, and $[m]^n$ denotes the set of size n subsets of $\{0,\dots,m-1\}$. Formulated in propositional logic, Degen's generalization is given by a family $\{D_{m,k} : m,k\in \mathrm{N}-\{0\}\}$ where $D_{m,k}$ is

$$\bigwedge_{0\le i\le m\cdot k}\bigvee_{0\le j<k} p_{i,j} \to \bigvee_{0\le j<k}\bigvee_{I\in[m\cdot k+1]^{m+1}}\bigwedge_{i\in I} p_{i,j}.$$

1991 *Mathematics Subject Classification.* Primary 03B05; Secondary 68Q15, 03F03.

Part of this research supported by NSF CCR-9408090, US-Czech Science and Technology Grant 93-025, and the Volkswagen Stiftung.

Part of this research supported by the Volkswagen Stiftung.

W. Degen showed that for m fixed, over ZF set theory without the axiom of choice, the set theoretic analogue of $\{D_{m,k} : k \in \mathbf{N}\}$ is properly weaker than the set theoretic analogue of $\{D_{m+1,k} : k \in \mathbf{N}\}$ when $m+1$ is a prime. The pigeonhole principle, Ajtai's parity principle, and various modular counting principles have been investigated in boolean circuit complexity and in propositional proof theory, the idea being that counting is a difficult notion to capture in finite depth circuits or proofs. Motivated by Degen's surprising hierarchy result in set theory, we investigated his principle in propositional logic. This paper establishes that Degen's principle is of the same strength as the pigeonhole principle. Additionally, we consider a propositional logic formulation of *st*-connectivity, and using the Karchmer-Wigderson lower bound for monotonic circuits, furnish an example where tree-like resolution is weaker than resolution (by a different proof, a separation between tree-like resolution and resolution was first given by Tseitin). Finally, we note the similarity of operations allowed in defining *graph minors* and those in restricted resolution proofs. This raises the question of a graph theoretic characterization of those bounded degree graphs whose Tseitin tautologies require large proofs in resolution or restricted resolution. In this context, we prove a relation between the pigeonhole principle and Tseitin's odd-charged graphs.

2. Preliminaries

We refer the reader to Krajíček's book [11] for any undefined terminology. Resolution is a sound, complete refutation system for conjunctive normal form (CNF) formulas – *sound*, in that if CNF formula ϕ has a resolution refutation, then ϕ is unsatisfiable, and *complete*, in that every unsatisfiable CNF formula has a resolution refutation. CNF formulas are represented in resolution by a set of clauses containing literals (a literal is a propositional variable or its negation), where the clause $\{\alpha_1, \ldots, \alpha_n\}$ represents $\alpha_1 \vee \cdots \vee \alpha_n$. The resolution rule allows the derivation of clause $C \cup D$ from the clauses $C \cup \{x\}$ and $D \cup \{\overline{x}\}$. A resolution derivation from $C_1, \ldots, C_n$ is a sequence of clauses $D_1, \ldots, D_m$, such that every D_i is either one of the C's, or obtained from D_j, D_k for $j, k < i$ by the resolution rule. A resolution refutation of clauses $C_1, \ldots, C_n$ is a derivation of the empty clause from $C_1, \ldots, C_n$. By abuse of notation, we say that a disjunctive normal form formula has a resolution proof, if its negation (a CNF formula) has a resolution refutation. A resolution derivation is tree-like if every clause is used at most once in an application of the resolution rule (multiple resolutions on the same clause require multiple derivations of that clause).

The cutting plane proof system, CP, is a sound and complete refutation system for CNF formulas. Propositional variable x_i is represented by itself; $\neg x_i$ is represented by $1 - x_i$; a disjunction $\bigvee_{i \in I} \alpha_i$ of literals is represented by $\sum_{i \in I} R(\alpha_i) \geq 1$, where $R(\alpha_i)$ represents the literal α_i; finally, a CNF formula $\bigwedge_{i \in I} \bigvee_{j \in J_i} \alpha_{i,j}$ is represented by the family

$$\sum_{j \in J_i} R(\alpha_{i,j}) \geq 1$$

of linear inequalities. Without loss of generality, we assume all linear inequalities are of the form $\sum a_i \cdot x_i \geq A$ where $a_i, A \in \mathbf{Z}$. The a_i are the coefficients of the propositional variables, and for lack of a better term, we call A the integer sum. The *axioms* of CP are $x_i \geq 0$, $-x_i \geq -1$. The *rules of inference* of CP are

$$\bullet \text{ addition } \frac{\sum a_i \cdot x_i \geq A \qquad \sum b_i \cdot x_i \geq B}{\sum (a_i + b_i) \cdot x_i \geq A + B}$$

$$\bullet \text{ division } \frac{\sum (c \cdot a_i) \cdot x_i \geq A}{\sum a_i \cdot x_i \geq \lceil \frac{A}{c} \rceil}$$

where integer $c > 1$,

$$\bullet \text{ multiplication } \frac{\sum a_i \cdot x_i \geq A}{\sum (c \cdot a_i) \cdot x_i \geq c \cdot A}$$

where integer $c > 1$.

A derivation D for inequalities I from inequalities $I_1, \ldots, I_m$ is a sequence $D = (D_0, \ldots, D_n)$ such that for all $i \leq n$ either D_i is an axiom, or one of $I_i, \ldots, I_m$ or inferred from D_j, D_k for $j, k < i$ by means of a rule of inference. A *refutation* of $I_1, \ldots, I_m$ is a derivation of $0 \geq 1$ from $I_1, \ldots, I_m$. As in the case of resolution, by abuse of terminology, we say that a disjunctive normal form formula has a CP proof if its negation has a CP refutation. The *size* of a CP refutation is the sum over all inequalities $\sum a_i \cdot x_i \geq A$ occurring in the refutation of $\sum |a_i| + |A|$, where $|A|$ indicates the length of the binary representation of A. It is easy to see [6] that CP is a sound extension of resolution, hence complete.

3. Cutting plane proofs of Degen's principle

By $E_{m,k}$ we denote the CP inequalities corresponding to the CNF formula $\neg D_{m,k}$. Thus $E_{m,k}$ is

$$\sum_{j=0}^{k-1} p_{i,j} \geq 1$$

for $0 \leq i \leq m \cdot k$, together with

$$-p_{i_1,j} - p_{i_2,j} - \cdots - p_{i_{m+1},j} \geq -m$$

for $0 \leq j < k$ and $0 \leq i_1 < i_2 < \cdots < i_{m+1} \leq m \cdot k$.

THEOREM 1. *There are $O(k^5)$ size CP refutations of $E_{2,k}$.*

PROOF. By assumption from $E_{2,k}$, for all $0 \leq i_1 < i_2 < i_3 \leq 2k$ and all $0 \leq r < k$,

$$2 \geq p_{i_1,r} + p_{i_2,r} + p_{i_3,r}.$$

CLAIM 2. *For all $0 \leq i_1 < i_2 < i_3 < i_4 \leq 2k$ and all $0 \leq r < k$,*

$$2 \geq p_{i_1,r} + p_{i_2,r} + p_{i_3,r} + p_{i_4,r}.$$

PROOF OF CLAIM: Fix i_1, i_2, i_3, i_4 and r, and temporarily, set $a = p_{i_1,r}$, $b = p_{i_2,r}$, $c = p_{i_3,r}$, $d = p_{i_4,r}$. By assumption from $E_{2,k}$, we have

$$\begin{aligned} 2 &\geq a+b+c \\ 2 &\geq b+c+d \\ 2 &\geq a+b+d \\ 2 &\geq a+c+d \end{aligned}$$

and so by addition

$$8 \geq 3a + 3b + 3c + 3d$$

and hence by division by 3

$$2 = \lfloor 8/3 \rfloor \geq a + b + c + d.$$

Q.E.D. claim.

For later generalization, note that the pattern of the previous inequalities is of the following form:

$$\begin{array}{cccc} + & + & + & - \\ - & + & + & + \\ + & - & + & + \\ + & + & - & + \end{array}$$

where $+$ [resp. $-$] indicates presence [resp. absence] of the corresponding element (i.e. in the first row, there is a, b, c but no d present). In this manner, with $O(k^5)$ (i.e. order $k \cdot \binom{2k+1}{4}$) many proof lines we can show that

$$2 \geq p_{i_1,r} + \cdots + p_{i_4,r}$$

for all rows $0 \leq r < k$ and all 4-tuples $0 \leq i_1 < i_2 < i_3 < i_4 \leq 2 \cdot k$ from that row. In a similar manner, we could show by a proof of $O(k^{s+1})$ lines, that $2 \geq p_{i_1,r} + \cdots + p_{i_s,r}$, for all rows $0 \leq r < k$ and all distinct s-tuples $i_1, \ldots, i_s$. However, the overall proof would then be of $\sum_{i=5}^{2k+1} O(k^i)$ lines, hence of exponential size. For that reason, in the following claim, we consider sets $i_1, \ldots, i_s$ of a particular form. Define integers $x_1, \ldots, x_m$ to be *consecutive* if for all $1 \leq j < m$, $x_{j+1} = x_j + 1$.

CLAIM 3. *Assume that $3 \leq s \leq 2k$ and for all $0 \leq i_1 < \cdots < i_s \leq 2k$ such that $i_2, \ldots, i_s$ are consecutive, and for all $0 \leq r < k$, it is the case that*

$$2 \geq p_{i_1,r} + \cdots p_{i_s,r}.$$

Then for all $0 \leq i_1 < \cdots < i_{s+1} \leq 2k$ such that $i_2, \ldots, i_{s+1}$ are consecutive, and for all $0 \leq r < k$, it is the case that

$$2 \geq p_{i_1,r} + \cdots + p_{i_{s+1},r}.$$

PROOF OF CLAIM: Fix $0 \leq i_1 < \cdots < i_{s+1}$ and r. By assumption

$$\begin{aligned} 2 &\geq p_{i_1,r} + \cdots + p_{i_s,r} \\ 2 &\geq p_{i_2,r} + \cdots + p_{i_{s+1},r} \\ 2 &\geq p_{i_1,r} + p_{i_3,r} + \cdots + p_{i_{s+1},r} \\ 2 &\geq p_{i_1,r} + p_{i_2,r} + p_{i_{s+1},r} \end{aligned}$$

Note that the pattern in the previous inequalities is of the following form:

$$\begin{array}{cccccc} + & + & + & \cdots & + & - \\ - & + & + & \cdots & + & + \\ + & - & + & \cdots & + & + \\ + & + & - & \cdots & - & + \end{array}$$

The first three inequalities hold by the assumption in the claim, and the fourth (which contains only 3 terms) holds by assumption of $E_{2,k}$. By addition, we have

$$8 \geq 3p_{i_1,r} + \cdots + 3p_{i_{s+1},r}$$

and hence by division by 3

$$2 = \lfloor 8/3 \rfloor \geq p_{i_1,r} + \cdots + p_{i_{s+1},r}.$$

Q.E.D. Claim.

By induction on s, using the base case $2 \geq p_{i_1,r} + p_{i_2,r} + p_{i_2+1,r}$ for all $0 \leq r < k$ and $0 \leq i_1 < i_2 \leq 2 \cdot k$ (given by $E_{2,k}$), and applying Claim 3 in the inductive case, it follows that for all $0 \leq r < k$,

$$2 \geq p_{0,r} + \cdots + p_{2k,r}.$$

Adding all k inequalities (one for each $0 \leq r < k$), we have

$$2k \geq \sum_{i=0}^{2k} \sum_{j=0}^{k-1} p_{i,j}.$$

However, by hypothesis $E_{2,k}$, for each fixed $0 \leq i \leq 2k$, $\sum_{j=0}^{k-1} p_{i,j} \geq 1$, and by addition of these $2k+1$ inequalities (one for each $0 \leq i \leq 2k$), we have

$$\sum_{i=0}^{2k} \sum_{j=0}^{k-1} p_{i,j} \geq 2k + 1.$$

Thus we arrive at the contradiction $2k \geq 2k+1$. Rewriting the above proof in the required normal form $\sum a_{i,j} \cdot p_{i,j} \geq A$ we obtain a derivation of $0 \geq 1$ from $E_{2,k}$.

What is the size of this CP refutation? In Claim 3, for each fixed $s \geq 3$, there are at most $O(2k)$ choices of $0 \leq i_1 \leq 2k$ and (by consecutivity) at most $O(2k)$ choices of the remaining consecutive $0 \leq i_2, \ldots, i_{s+1} \leq 2k$ with $i_1 < i_2$. There are k many values of $0 \leq r < k$, so altogether this makes $O(k^3)$ proof lines for establishing the claim in going from s to $s+1$. As $s \leq 2k$, the entire proof requires $O(k^4)$ lines. The coefficients of the propositional variables have size bounded by 2 (the largest coefficient is 3).

Except in the last two steps, where at most $2k+1$ inequalities are added (producing sums $2k$ and $2k+1$), all sums are bounded by 8. Each inequality (proof line) has size $O(k)$, since coefficients of the variables are bounded by a constant, the integer sum is bounded by $2k+1$, and there are at most $O(k)$ variables per inequality. Thus the proof size is $O(k^4 \cdot k)$ or $O(k^5)$. □

THEOREM 4. *Let $m \geq 2$ and $n = m \cdot k + 1$. Then there are $O(n^{m+3})$ size CP refutations of $E_{m,k}$, where the constant in the O-notation depends on m, and $O(n^{m+4})$ size CP refutations, where the constant is independent of n, m.*

PROOF. We generalize the proof of the previous theorem.

CLAIM 5. *Assume that $3 \leq s \leq mk$ and for all $0 \leq i_1 < \cdots < i_s \leq mk$ such that $i_m, \ldots, i_s$ are consecutive, and for all $0 \leq r < k$, it is the case that*

$$m \geq p_{i_1,r} + \cdots + p_{i_s,r}.$$

Then for all $0 \leq i_1 < \cdots < i_{s+1} \leq mk$ such that $i_m, \ldots, i_{s+1}$ are consecutive, and for all $0 \leq r < k$, it is the case that

$$m \geq p_{i_1,r} + \cdots + p_{i_{s+1},r}.$$

PROOF OF CLAIM: Fix $i_1 < \cdots < i_{s+1}$ and r. We have the following $m+2$ inequalities:

$$\begin{aligned}
m &\geq p_{i_1,r} + \cdots + p_{i_s,r} \\
m &\geq p_{i_2,r} + \cdots + p_{i_{s+1},r} \\
m &\geq p_{i_1,r} + p_{i_3,r} + \cdots + p_{i_{s+1},r} \\
m &\geq p_{i_1,r} + p_{i_2,r} + p_{i_4,r} + \cdots + p_{i_{s+1},r} \\
m &\geq p_{i_1,r} + \cdots + p_{i_3,r} + p_{i_5,r} + \cdots + p_{i_{s+1},r} \\
m &\geq p_{i_1,r} + \cdots + p_{i_4,r} + p_{i_6,r} + \cdots + p_{i_{s+1},r} \\
&\vdots \\
m &\geq p_{i_1,r} + \cdots + p_{i_{m-1},r} + p_{i_{m+1},r} + \cdots + p_{i_{s+1},r} \\
m &\geq p_{i_1,r} + \cdots + p_{i_m,r} + p_{i_{s+1},r}
\end{aligned}$$

The pattern of terms in the $m+2$ inequalities above is of the form:

$$\begin{array}{cccccccccc}
+ & + & + & \cdots & + & + & + & + & - \\
- & + & + & \cdots & + & + & + & + & + \\
+ & - & + & \cdots & + & + & + & + & + \\
+ & + & - & \cdots & + & + & + & + & + \\
\cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot & \cdot & \cdot \\
+ & + & + & \cdots & + & - & + & + & + \\
+ & + & + & \cdots & + & + & - & - & +
\end{array}$$

Removal of any of the first $m-1$ summands in the term $p_{i_1,r} + \cdots + p_{i_{s+1},r}$ produces a term where $p_{i_m,r}, \ldots, p_{i_{s+1},r}$ are consecutive. This observation, with the assumption in the claim, justifies the first $m+1$ inequalities. The last inequality (which contains only $m+1$ terms) holds by assumption of $E_{m,k}$. By addition, we have

$$m \cdot (m+2) \geq (m+1) \cdot (p_{i_1,r} + \cdots + p_{i_{s+1},r})$$

and hence by division by $m+1$

$$m = \lfloor \frac{m(m+2)}{m+1} \rfloor \geq p_{i_1,r} + \cdots + p_{i_{s+1},r}.$$

Q.E.D. Claim.

Adding k inequalities $m \geq p_{0,r} + \cdots + p_{mk,r}$, we have $mk \geq \sum_{i=0}^{mk} \sum_{j=0}^{k-1} p_{i,j}$. Similarly adding the $mk+1$ inequalities $p_{i,0} + \cdots + p_{i,k-1} \geq 1$, we have $\sum_{i=0}^{mk} \sum_{j=0}^{k-1} p_{i,j} \geq mk+1$. Finally, we have the desired contradiction $mk \geq mk+1$.

Let $n = mk$. In Claim 5, for each fixed $3 \leq s \leq mk$, there are at most $O(n^m)$ choices of $0 \leq i_1 < \cdots < i_s \leq mk$ for which $i_m, \ldots, i_s$ are consecutive (such sequences are stipulated by choice of $i_1, \ldots, i_m$). There are k many values of $0 \leq r < k$, so altogether this makes $O(n^{m+1})$ proof lines for establishing the claim in going from s to $s+1$. As $s \leq mk$, the entire proof requires $O(n^{m+2})$ lines. In each proof line (inequality), there are at most $mk+1$ variables (hence $O(k)$, if m is held as a constant), while the coefficients of the variables are bounded by $m+1$, and the integer sum (except in the last steps where k inequalities are added together) are bounded by $m(m+2)$. In the last steps involving addition of k inequalities, the integer sum is bounded by $mk+1$, hence the size of each inequality (proof line) in the proof is bounded by $O(n)$ if m is held as a constant. It follows that the proof size is $O(n^{m+2} \cdot n)$ or $O(n^{m+3})$, where the constant in the O-notation depends only on m. The previous considerations yield that the proof size is $O(n^{m+3} \cdot \log m)$ or $O(n^{m+4})$, where the constant in the O-notation is an absolute constant independent of n, m. □

COROLLARY 6. *There are polynomial size CP refutations of $\{E_{m,k} : m \geq 2, k \geq 1\}$.*

PROOF. Each application of Claim 3 requires $m+1$ additions and one division. Taking this into account, as in the previous analysis, there are then $O(m \cdot n^{m+3})$ proof lines, each of size $O(\log m)$, so the refutation size of $E_{m,k}$ is $O(n^{m+4})$, where the constant in the O-notation is independent of n, m, k. On the other hand, the size of $E_{n,k}$ is $O(n^{m+1})$. Thus the refutation size is polynomial in the size of the formula being refuted. □

Let $[n]^r$ [resp. $[X]^r$] denote the set of r-element subsets of $\{0, \ldots, n-1\}$ [resp. X]. The Erdös-Rado partition calculus notation $n \to (m)^r_k$ means that for any partition $f : [n]^r \to \{0, \ldots, k-1\}$, there is a size m subset X of

$\{0,\ldots,n-1\}$ on which f is constant; i.e. $f([X]^r)$ has cardinality 1. Using the Erdös-Rado partition calculus notation, PHP_n can be written as

$$n+1 \to (2)^1_n$$

and Degen's principle as

$$mk+1 \to (m+1)^1_k.$$

QUESTION 7. *Are there polynomial size CP proofs of versions of Ramsey's theorem*

$$n+1 \to (m)^r_k$$

for appropriate n, m, r, k?

On p. 327 of [3], V. Chvatal gave a cutting planes proof of the instance

$$\begin{array}{lll} 14 & \not\to & (3)^2_2 \\ 15 & \to & (3)^2_2 \end{array}$$

of Ramsey's theorem, and claimed that a general form of Ramsey's theorem could be proved in cutting planes. Since no details were given, it is unclear whether Chvatal's intended proof was indeed polynomial size. In [13], P. Pudlák has shown the existence of polynomial size constant depth Frege proofs of an appropriate formulation of Ramsey's theorem. It appears unlikely that CP (or $CPLE$, an extension of cutting planes with limited extension, see [2]) can polynomially simulate constant depth Frege systems. Hence it would be of interest to extend the counting arguments within CP to prove Ramsey's theorem and stronger combinatorial theorems.

4. Polynomial equivalence over resolution between Degen's principle and the pigeonhole principle

In the following we denote the cardinality of a finite set S by $||S||$.

DEFINITION 8. *The proof system Q* p-simulates *the proof system P if there is a polynomial time computable function f such that for all x, y, if x is a P-proof of y then $f(x,y)$ is a Q-proof of (the translation of) of y. Proof systems P, Q are* p-equivalent *if each p-simulates the other.*

In this section, we show that over resolution, the clausal form of Degen's principle

$$(\forall f : (mk+1) \to k)(\exists i < k)(||f^{-1}(i)|| \geq m+1)$$

is p-equivalent to the clausal form of the pigeonhole principle

$$(\forall g : (mk+1) \to mk)(\exists i < mk)(||g^{-1}(i)|| \geq 2).$$

(Note that we identify n with $\{0,1,\ldots,n-1\}$).
The idea of proof by contrapositives is quite simple. To show $\neg D_{m,k} \Rightarrow \neg PHP_{mk}$, suppose that $f : (mk+1) \to k$ violates Degen's principle; i.e.

$$(\forall j < k)||(f^{-1}(j)|| \leq m) \ .$$

Define $g : (mk+1) \to mk$ as follows: $g(i) = l \cdot k + j$ iff $f(i) = j$ and this is the $l+1$th such i (in case $l = m-1$ at least the mth such i). In other words, $g(i) = lk + j$ iff

$$(l < m-1 \wedge f(i) = j \wedge ||f^{-1}(j) \cap i|| = l) \text{ or}$$
$$(l = m-1 \wedge f(i) = j \wedge ||f^{-1}(j) \cap i|| \geq l) \ .$$

Since f violates Degen's principle, it follows that g is injective, so g violates the pigeonhole principle. To show $\neg PHP_{mk} \Rightarrow \neg D_{m,k}$, suppose that $g : (mk+1) \to mk$ is injective. It follows that $f(i) := g(i) \bmod k$ violates Degen's principle. In this section, the previous argument of equivalence is formalized within resolution.

Throughout this section, fix k, m and let $n = mk$. In the following definitions, the reader should bear in mind that in the sketch proof that $\neg D_{m,k} \Rightarrow \neg PHP_{mk}$, the p's encode f, and the q's encode g; i.e. $p_{i,j} = 1$ iff $f(i) = j$ and $q_{i,j} = 1$ iff $g(i) = j$.

DEFINITION 9. (a) *In the following formulas denote the clauses equivalent to them. For instance $\bigwedge_{i\in I} p'_i \to \bigvee_{i\in J} q'_i$ denotes the clause*

$$\{\overline{p}'_i : i \in I\} \cup \{q'_i : i \in J\} \ .$$

(b) *The set of defining clauses for $p' \equiv \bigwedge_{i\in I} q'_i$ is the set of clauses*

$$\{p' \to q'_i : i \in I\} \cup \{\bigwedge_{i\in I} q'_i \to p'\} \ .$$

The set of defining clauses for $p' \equiv \bigvee_{i\in I} q'_i$ is the set

$$\{p' \to \bigvee_{i\in I} q'_i\} \cup \{q'_i \to p' : i \in I\} \ .$$

DEFINITION 10. *$\neg D_{m,k}$ is the set of clauses expressing the refutable version of Degen's principle*

$$(\exists f : (n+1) \to k)(\forall i < k)(||f^{-1}(i)|| \leq m)$$

so that

$$\neg D_{m,k} := D^{\neg,0}_{m,k} \cup D^{\neg,1}_{m,k}$$

with

$$D^{\neg,0}_{m,k} := \{\bigvee_{j<k} p_{i,j} : i \leq n\}$$

and

$$D^{\neg,1}_{m,k} := \{\bigvee_{i\in S} \overline{p}_{i,j} : j < k, S \in [n+1]^{m+1}\}.$$

DEFINITION 11. (a) $[i]^{<m} := \bigcup_{l<m} [i]^l$.
(b) *If $||S|| < m-1$, then let $\mathrm{Def}(p^S_{i,j})$ be the set of defining clauses for*

$$p^S_{i,j} \equiv p_{i,j} \wedge \bigwedge_{i'\in S} p_{i',j} \wedge \bigwedge_{i'\in(i\setminus S)} \overline{p}_{i',j} \ .$$

If $||S|| = m-1$, *let* $\mathrm{Def}(p^S_{i,j})$ *be the set of defining clauses for*

$$p^S_{i,j} \equiv p_{i,j} \wedge \bigwedge_{i' \in S} p_{i',j}.$$

(c)

$$\mathrm{Def}(\vec{p^S}) := \bigcup_{i \le n} \bigcup_{j<k} \bigcup_{S \in [i]^{<m}} \mathrm{Def}(p^S_{i,j}).$$

(d) *For* $i \le n$, $l < m$, $j < k$ *let* $\mathrm{Def}(q_{i,j})$ *be the set of defining clauses for*

$$q_{i,lk+j} \equiv \bigvee_{S \in [i]^l} p^S_{i,j}$$

(e)

$$\mathrm{Def}(\vec{q}) := \bigcup_{i \le n} \bigcup_{j<n} \mathrm{Def}(q_{i,j}) \ .$$

We have: If $f(i) = j$ and $f^{-1}(j) \cap i \supseteq S$, then $p^{S'}_{i,j}$ holds for some $S' \supseteq S$ (which implies, that $q_{i,lk+j}$ holds for some l):

LEMMA 12. *If* $i \le n$, $j < k$, $S \in [i]^{<m}$, *then*

$$D^{\neg,0}_{m,k}, \mathrm{Def}(\vec{p^S}) \vdash (p_{i,j} \wedge \bigwedge_{i' \in S} p_{i',j}) \to \bigvee_{S \subseteq S' \in [i]^{<m}} p^S_{i,j} \ .$$

For all possible choices of i, j, S *altogether we need* $\le (n+1)^{m+3}$ *resolution steps.*

PROOF. By induction on $(m-1) - ||S||$.
The base case $||S|| = m-1$ is contained in $\mathrm{Def}(p^S_{i,j})$.
For the induction step, assume $||S|| = l < m-1$, and that the theorem is proven for $||S|| = l+1$. Let $A := p_{i,j} \wedge \bigwedge_{i' \in S} p_{i',j}$.
By $\mathrm{Def}(\vec{p^S})$ we have

$$(A \wedge \bigwedge_{i' \in (i \setminus S)} \overline{p}_{i',j}) \to p^S_{i,j} \ .$$

By the induction hypothesis follows for $i' \in (i \setminus S)$

$$(A \wedge p_{i',j}) \to \bigvee_{S \cup \{i'\} \subseteq S' \in [i]^{<m}} p^{S'}_{i,j} \ .$$

Now, applying $i - ||S||$ resolution steps we deduce the assertion.

To compute the number of resolution steps, note that for every $i \le n$, $j < k$, $l < m$, $S \in [i]^l$, $i - ||S|| \le n$ steps are required, and $||[i]^l|| = \binom{i}{l} \le (n+1)^m$, therefore altogether at most $k \cdot (n+1) \cdot m \cdot (n+1)^m \cdot (n+1) = km(n+1)^{m+2} \le (n+1)^{m+3}$ steps are required. □

From lemma 12 follows now immediately, that $\forall i \le n \exists y < n.g(i) = y$ (namely $g(i) = lk + j$ if $p^S_{i,j}$ and $||S|| = l$):

LEMMA 13. *For* $i \le n$

$$\bigvee_{j<k} p_{i,j}, \mathrm{Def}(\vec{p^S}), \mathrm{Def}(\vec{q}) \vdash \bigvee_{y<n} q_{i,y}$$

for all i *altogether in* $\le 3(n+1)^{m+3}$ *steps.*

PROOF. By lemma 12, for each fixed i we have $p_{i,j} \to \bigvee_{S \in [i]^{<m}} p^S_{i,j}$. From this and $\bigvee_{j<k} p_{i,j}$ in k resolution steps we deduce

$$\bigvee_{S \in [i]^{<m}} \bigvee_{j<k} p^S_{i,j} \ .$$

By $\mathrm{Def}(\vec{q})$ $p^S_{i,j} \to q_{i,k \cdot ||S|| + j}$. In $\sum_{i \le n} \sum_{j<k} \sum_{l<m} ||[i]^l||$ resolution steps we conclude

$$\bigvee_{y<n} q_{i,y} \ .$$

Number of steps needed: For lemma 12 at most $(n+1)^{m+3}$ steps, additionally at most $k \cdot (n+1) \cdot (k + (n+1)^{m+1} \cdot m)$, so altogether at most $3(n+1)^{m+3}$ steps. □

We show now that g is injective. First we have that we never get $p^S_{i,j}$ and $p^{S'}_{i',j}$ with $i \neq i'$, $||S|| = ||S'||$:

LEMMA 14. *Assume* $i < i' \le n$, $j < k$, $S \in [i]^l$, $S' \in [i']^l$. *If* $l < m-1$, *then*

$$\mathrm{Def}(\vec{p^S}) \vdash \overline{p}^S_{i,j} \vee \overline{p}^{S'}_{i',j}$$

and if $l = m-1$, *then*

$$\mathrm{Def}(\vec{p^S}), D^{\neg,1}_{m,k} \vdash \overline{p}^S_{i,j} \vee \overline{p}^{S'}_{i',j}.$$

For all possible i, i', j, S *together, at most* $(n+1)^{2m+3}(m+1)$ *steps are needed.*

PROOF. Case $l < m-1$:
If $S = S'$, then, by $\mathrm{Def}(\vec{p^S})$, we have $p^S_{i,j} \to p_{i,j}$, since $S' = S \subseteq i$, $i \notin S'$, and therefore $p^{S'}_{i',j} \to \overline{p}_{i,j}$, by resolution $\overline{p}^S_{i,j} \vee \overline{p}^{S'}_{i',j}$.
If $S \neq S'$, it follows, since $||S|| = ||S'||$, that $S \not\subseteq S'$, therefore there exists some $i_0 \in (S \setminus S')$, we have $p^S_{i,j} \to p_{i_0,j}$, $p^{S'}_{i',j} \to \overline{p}_{i_0,j}$, again by resolution the assertion.
Case $l = m-1$:
By $D^{\neg,1}_{m,k}$

$$\bigvee_{i_0 \in S} \overline{p}_{i_0,j} \vee \overline{p}_{i,j} \vee \overline{p}_{i',j} \ ,$$

by $\mathrm{Def}(\vec{p^S})$, for $i_0 \in S$, $p^S_{i,j} \to p_{i_0,j}$, $p^S_{i,j} \to p_{i,j}$, $p^{S'}_{i',j} \to p_{i',j}$, by $m+1$ resolution steps $\overline{p}^S_{i,j} \vee \overline{p}^{S'}_{i',j}$.
Number of steps needed: there are at most
$\sum_{i<i' \le n} \sum_{j<k} \sum_{l<m} \sum_{S \in [i]^l} \sum_{S \in [i']^l} (m+1) \le km(m+1)(n+1)^{2m+2}$
$\le (n+1)^{2m+3}(m+1)$ steps. □

LEMMA 15. *If $i < i' \leq n$, $y < n$,*

$$\mathrm{Def}(\vec{p^S}), \mathrm{Def}(\vec{q}), D^{\neg,1}_{m,k} \vdash \overline{q}_{i,y} \vee \overline{q}_{i',y} \ .$$

All proofs together need (if $k \geq 2$) at most $(n+1)^{2m+4}$ steps.

PROOF. By lemma 14, if $l < m$, $j < k$, $S \in [i]^l$, $S' \in [i']^l$ it follows that $\overline{p}^S_{i,j} \vee \overline{p}^{S'}_{i',j}$, further by $\mathrm{Def}(\vec{q})$ in $||[i]^{<m}||$ resolution steps

$$q_{i,kl+j} \rightarrow \bigvee_{S \in [i]^l} p^S_{i,j} \ ,$$

therefore

$$q_{i,kl+j} \rightarrow \overline{p}^{S'}_{i',j},$$

by $\mathrm{Def}(\vec{q})$

$$q_{i',kl+j} \rightarrow \bigvee_{S' \in [i']^l} p^{S'}_{i',j} \ ,$$

by $||[i']^l||$ steps $\overline{q}_{i,kl+j} \vee \overline{q}_{i',kl+j}$.

Number of Steps: steps from lemma 14 at most $(m+1)(n+1)^{2m+3}$ steps, additionally

$$2 \sum_{i<i' \leq n} \sum_{l<m} \sum_{j<k} (||[i]^{<m}|| \cdot ||[i']^{<m}||) \leq 2(n+1)^2 \cdot k \cdot m \cdot (n+1)^{2m} \leq 2(n+1)^{2m+3}$$

steps, altogether at most $(n+1)^{2m+3} \cdot (m+3)$ steps. □

This completes the formalization in resolution of $\neg D_{m,k} \Rightarrow \neg PHP_{mk}$. The formalization in resolution of the converse is straightforward and left to the reader. From the analysis of resolution steps and consideration of the number of variables which can appear in any clause, we deduce that $D_{m,k} \equiv PHP_{mk}$ has resolution proofs of size polynomial in the size of $D_{m,k}$ and PHP_{mk}.

Note that the proof of equivalence really uses extended resolution, since we introduced polynomially (in n) many new literals $p^S_{i,j}$ and $q_{i,lk+j}$. The $p^S_{i,j}$ were defined in terms of the original $p_{i,j}$, and the $q_{i,lk+j}$ were defined in terms of the $p^S_{i,j}$; thus the depth of the extension is 2, and all definitions involve polynomially (in n) many literals. By substituting the defining clauses appropriately, from a derivation of the empty clause from $\neg PHP_{mk}$ we can obtain a derivation of the empty clause from $\neg D_{m,k}$. It is in this sense we mean that over resolution PHP_{mk} and $D_{m,k}$ are polynomially equivalent.

Since cutting planes p-simulates resolution, it follows that $D_{m,k} \equiv PHP_{mk}$ has polynomial size cutting planes proofs (more precisely, cutting planes with extension, where the depth of the extension is 2, and all extending inequalities involve polynomially (in n) many literals; see [5] for information on cutting planes with extension). Since PHP_{mk} is well-known to have cutting planes proofs of size polynomial in mk, we have an alternative proof of the main result of the previous section.

5. *st*-Connectivity

Graph connectivity has been studied under various guises by many authors (Floyd-Warshall's $O(n^2 \log n)$ algorithm for transitive closure, the well-known observation that the so-called directed *graph accessibility problem* GAP is complete for nondeterministic logarithmic space, Borodin's observation that LOGSPACE is contained in AC^1, etc.). In [10], M. Karchmer and A. Wigderson gave a significant size lower bound for boolean circuits computing *st-connectivity*, the problem whether there is a path from designated nodes s to t in an undirected graph.

THEOREM 16 (Karchmer-Wigderson [10]). *Monotonic fan-in 2 depth of st-connectivity is $\Theta(\log^2 n)$.*

In [18], I. Wegener proved the monotonic analogue of Spira's theorem which relates depth and size of monotonic formulas: a problem P has depth $O(d)$ monotonic formulas if and only if P has size $2^{O(d)}$ monotonic circuits (see Boppana-Sipser [1] for an overview of boolean circuit complexity). Wegener's result, with the previous theorem, implies the following.

COROLLARY 17 (Karchmer-Wigderson [10]). *Monotonic formula size of st-connectivity is $\Theta(n^{\log n})$.*

Another application of the previous theorem, not used in this paper, is the following result.

THEOREM 18 (Clote [4]). *The monotonic depth for fan-in 2 circuits recognizing whether a 2-CNF formula is refutable is $\Theta(\log^2 n)$.*

There are various possible formulations of *st*-connectivity in propositional logic.

DEFINITION 19 (*st*-connectivity (Form 1)). *Assume that G is a finite undirected graph, with two designated vertices s, t of degree 1, while all other vertices have degree 2. Then there is a path from s to t.*

This notion of *st*-connectivity, proposed by P. Pudlák, was investigated by S. Buss and P. Clote in [2] (this version of *st*-connectivity for *directed graphs* was there shown to be equivalent (over constant depth, polynomial size Frege systems) to the onto version of the pigeonhole principle, and polynomial size CP proofs were given for *st*-connectivity for undirected graphs). An alternate, weaker form of *st*-connectivity is now defined.

DEFINITION 20 (*st*-connectivity (Form 2)). *Assume that G is a finite undirected graph with two distinct, designated vertices s, t. Then either there is a path from s to t, or there is a partition of the vertices of G into two classes, where s and t lie in different classes and no edge goes between vertices lying in different classes.*

We will show that there are polynomial size resolution proofs for *st*-connectivity (Form 2), though of course, since *st*-connectivity (Form 1)

implies the pigeonhole principle, there is an exponential lower bound for resolution proof size for *st*-connectivity (Form 1). The formalization in resolution of Definition 20 is now given. The formula $A(\vec{p},\vec{q})$ is the conjunction of the following clauses:

1. $q_{0,0}$
2. $q_{n+1,n+1}$
3. $\overline{q}_{i,j},\overline{q}_{i,k}$, for all $j \neq k$ in $\{0,\ldots,n+1\}$.
4. $q_{i,0},\ldots,q_{i,n+1}$, for all $i \in \{1,\ldots,n\}$.
5. $\overline{q}_{i,j},\overline{q}_{i+1,k},p_{j,k}$, for all $j \neq k$ in $\{0,\ldots,n+1\}$.
6. $\overline{p}_{i,j},p_{j,i}$, for all $i \neq j$ in $\{0,\ldots,n+1\}$.

The idea is that the p's express the edge relation of G ($p_{i,j}=1$ iff there is a directed edge from i to j), and that the q's define a path from $s=0$ to $t=n+1$. We allow multiple occurrences of the same vertex along a path.] Thus $A(\vec{p},\vec{q})$ expresses that G is a directed graph and there is a path from s to t.

The formula $B(\vec{p},\vec{r})$ is the conjunction of the following clauses:

1. $\overline{r}_0$
2. r_{n+1}
3. $\overline{r}_i,\overline{p}_{i,j},r_j$, for all $i \neq j$ in $\{0,\ldots,n+1\}$.

The idea is that the p's express the edge relation of G, and the r's express the partition: those vertices i in the same partition class as s (we identify s with 0) satisfy $\overline{r}_i$, while those in the same class as t (we identify t with $n+1$) satisfy r_i. Thus $B(\vec{p},\vec{r})$ expresses that there is a *cut* (or partition into 2 classes), for which s [resp. t] belongs to [resp. does not belong to] the cut, and such that no edge goes from a vertex belonging to the cut to a vertex not belonging to the cut.

The resolution formulation of *st*-connectivity (Form 2) is the conjunction of both $A(\vec{p},\vec{q})$, which expresses that either G is not an undirected graph, or there is a path from s to t, and $B(\vec{p},\vec{r})$, which states that there is a partition of G's vertices, with s,t in different classes, and for which no edge of G goes between vertices in different classes. Note that all occurrences of $\vec{p}$ in the clauses B are negative.

J. Krajíček [12] proved an interesting interpolation result, which relates monotonic circuit lower bounds with lower bounds resolution proofs (P. Pudlák [14] extended this to an interpolation result relating monotonic real circuit lower bounds with lower bounds for cutting plane proofs).

THEOREM 21 (Krajíček [12]). *Suppose that propositional variables $\vec{p}$ are positive in $A(\vec{p},\vec{q})$, or that $\vec{p}$ are negative in $B(\vec{p},\vec{r})$, and that there is a resolution refutation P of $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$ of depth d and size s. Then there is a monotonic boolean circuit C of depth d and size $O(s)$ for which*

$$C(\vec{p}) = \begin{cases} 0 & \text{if } A(\vec{p},\vec{q}) \text{ is refutable} \\ 1 & \text{else if } B(\vec{p},\vec{r}) \text{ is refutable} \end{cases}$$

Moreover, if P is a proof tree, then the circuit C has fan-out 1.

By Theorem 21 and Corollary 17 we have the following.

THEOREM 22. *All tree-like resolution proofs of st-connectivity (Form 2) have size $\Omega(n^{\log n})$.*

PROOF. Let $s(n)$ be the size of a refutation of $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$, where the latter expresses the statement that a directed graph G on n nodes has a path from s to t and at the same time a cut between s and t. From Theorem 21 there are monotone boolean circuits C of size $O(s(n))$ such that $C(\vec{a})=0$ implies that $A(\vec{a},\vec{q})$ is refutable and $C(\vec{a})=1$ implies that $B(\vec{a},\vec{r})$ is refutable. Restricting C to those inputs $\vec{a}$ which correctly encode a directed graph G, it follows that $C(\vec{a})=1$ if and only if there is a path from s to t. By Corollary 17, it follows that $s(n)$ must be $\Omega(n^{\log n})$. □

The separation between tree-like resolution and resolution is a corollary of the following.

THEOREM 23. *There are polynomial size resolution proofs of st-connectivity (Form 2).*

PROOF. We begin by the following claim.

Claim: For $1 \leq i \leq n+1$, there is a resolution proof of $\overline{q}_{i,j},\overline{r}_j$

The proof of the claim is by induction on i. For the base case of $i=1$, note that

$$\dfrac{\dfrac{\dfrac{\overline{q}_{0,0},\overline{q}_{1,k},p_{0,k} \qquad \dfrac{\overline{r}_k,\overline{p}_{k,0},r_0 \qquad \overline{p}_{0,k},p_{k,0}}{\overline{r}_k,\overline{p}_{0,k},r_0}}{\overline{q}_{0,0},\overline{q}_{1,k},\overline{r}_k,r_0} \qquad q_{0,0}}{\overline{q}_{1,k},\overline{r}_k,r_0} \qquad \overline{r}_0}{\overline{q}_{1,k},\overline{r}_k}$$

The resolution proof for the base case is $O(n)$ size. Now, the induction hypothesis is

$$\overline{q}_{i,j},\overline{r}_j.$$

We have the following auxiliary result.

$$\dfrac{\overline{q}_{i,j},\overline{q}_{i,k},p_{j,k} \qquad \dfrac{\overline{p}_{j,k},p_{k,j} \qquad \overline{r}_k,\overline{p}_{k,j},r_j}{\overline{r}_k,\overline{p}_{j,k},r_j}}{\overline{q}_{i,j},\overline{q}_{i+1,k},\overline{r}_k,r_j} \qquad \overline{q}_{i,j},\overline{r}_j}{\overline{q}_{i,j},\overline{q}_{i+1,k},\overline{r}_k}$$

Now

$$\dfrac{\dfrac{q_{i,0},q_{i,1},\ldots,q_{i,n+1} \qquad \overline{q}_{i,0},\overline{q}_{i+1,k},\overline{r}_k}{q_{i,1},q_{i,2},\ldots,q_{i,n+1},\overline{q}_{i+1,k},\overline{r}_k} \qquad \overline{q}_{i,1},\overline{q}_{i+1,k},\overline{r}_k}{q_{i,2},\ldots,q_{i,n+1},\overline{q}_{i+1,k},\overline{r}_k}$$

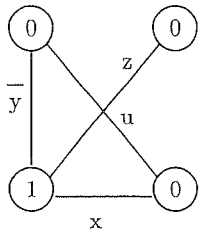


FIGURE 1. Odd-charged graph with edges labeled by literals

Inductively continuing in this manner, we obtain

$$\overline{q}_{i+1,k}, \overline{r}_k.$$

This completes the inductive case. For i, k fixed, there are $O(n)$ additional resolution steps, with overall size $O(n^2)$.

Taking $i = n + 1$, it follows that

$$\overline{q}_{n+1,k}, \overline{r}_k$$

for all k, so that

$$\dfrac{\dfrac{\overline{q}_{n+1,n+1}, \overline{r}_{n+1} \qquad q_{n+1,n+1}}{\overline{r}_{n+1}} \qquad r_{n+1}}{\emptyset}$$

We have thus derived the empty clause by a proof of size $O(n^4)$ from the assum ptions. □

COROLLARY 24. *Tree-like resolution does not polynomially simulate resolution.*

See A. Urquhart's survey article [17] for discussion of Tseitin's original argument.

6. Tseitin's odd-charged graphs

In [15], Tseitin associated propositional formulas with labeled undirected graphs, and developed a technique for obtaining lower bounds for regular resolution refutations (regular resolution allows, on any branch of the refutation tree, at most one resolution on any particular variable).

Suppose that G is a finite, undirected graph, whose vertices are labeled by 0, 1 and whose edges are labeled by distinct propositional literals (if literal α labels edge e, then neither α nor $\overline{\alpha}$ can label another edge). The label on a vertex is said to be its charge. The graph G is said to be odd-charged, if the sum modulo 2 of the vertex labels is 1. Figure 1 is an example.

Associate with G its charge equations, i.e. for vertex v the equation $EQ(v)$ states that the sum modulo 2 of the literals incident to v equals the charge on v. For the example in Figure 1, here are the charge equations:

1. $\overline{y} \oplus u = 0$
2. $\overline{y} \oplus x \oplus z = 1$
3. $z = 0$
4. $x \oplus u = 0$

The Tseitin clauses $F(G)$ associated with graph G are the clauses corresponding to the CNF formulation of the charge equations. With this example, we have:

1. $\{\overline{u}, \overline{y}\}, \{u, y\}$
2. $\{x, y, \overline{z}\}, \{x, \overline{y}, z\}, \{\overline{x}, y, z\}, \{\overline{x}, \overline{y}, \overline{z}\}$
3. $\{\overline{z}\}$
4. $\{x, \overline{u}\}, \{\overline{x}, u\}$

The rule for producing clauses from a charge equation is to place an odd [resp. even] number of negations on the associated literals, if the charge is 0 [resp. 1]. Clearly, there are 2^{d-1} clauses associated with the charge equation for vertex v if the degree of v is d (note that half of the 2^d truth assignments satisfy the charge equation). When considering proof size, we are thus only interested in graph families of bounded degree. The key property of odd-charged graphs is given by the following.

FACT 25 (Tseitin [15, 17]). *The connected graph G is odd-charged if and only if the clauses $F(G)$ are unsatisfiable.*

In [15], Tseitin developed recurrence relations for regular resolution refutation size, depending on a connectivity parameter for the graphs, and thus proved an exponential lower bound for regular resolution refutations for $F(L_n)$, where L_n is the odd-charged $n \times n$ lattice. In [16], A. Urquhart employed A. Haken's lower bound technique [8] to prove an exponential lower bound for Tseitin tautologies associated with particular expander graphs. Two interesting questions remain in this area:

1. Are there polynomial size CP refutations for Urquhart's formulas [16]?
2. For which families of graphs of bounded degree are there superpolynomial lower bounds for resolution [resp. regular resolution] refutations of Tseitin formulas?

Despite Tseitin's recurrence relations, it seems to be an interesting open problem to determine which graph theoretic properties lead to polynomial size regular resolution proofs.

LEMMA 26. *Let u, v be two nodes of a charged, labeled, undirected graph G, which are joined by an edge in G labeled by x. Let G' be obtained from G by contracting the edge $\{u, v\}$. In other words, define $V(G') = V(G) - \{u, v\} \cup \{w\}$ where $w \notin V(G)$ is a new node, and*

$$E(G') = E(G) - \{e : u \in e \text{ or } v \in e\} \cup \{\{r, w\} : \{r, u\} \in E(G) \text{ or } \{r, v\} \in E(G)\}.$$

The charge on every node in $V(G') - \{w\}$ is the same as the charge of that node in G, while the charge on w is defined to be

$$charge(u) \oplus charge(v).$$

Then there are $2^{dg(u)+dg(v)-3}$ resolution steps to derive the clauses associated with the charge equation $EQ(w)$ in G' from the charge equations $EQ(u)$, $EQ(v)$ in G.

The proof of this lemma follows from a simple computation, where positive [resp. negative] occurrences of the edge literal x in $EQ(u)$ are resolved against negative [resp. positive] occurrences of x in $EQ(v)$. The formal proof is left to the reader.

DEFINITION 27. *The graph H is a minor of the graph G, if H can be obtained from G by the operations of deleting an isolated vertex, removing an edge, contracting an edge.*

Edge deletions and their relation to regular resolution were first discussed by Tseitin [**15, 17**], and were the basis of his recurrence relations. Isolated vertices play no role in Tseitin's formulas, as there is no charge equation for such vertices. Finally, edge contractions can be handled by the previous lemma. From this discussion, it is clear that a precise relation between $F(G)$ and $F(H)$ can be worked out, if H is a minor of G. Moreover, a simple computation, contracting edges beginning with those adjacent to the leaves, shows that the Tseitin formulas $F(G_n)$ have linear size regular resolution refutations, for families $\{G_n : n \in \mathbf{N}\}$ of bounded degree odd-charged trees. This follows from the next proposition.

PROPOSITION 28. *Let T be a rooted, odd-charged binary tree with n nodes and degree bound e. Then there are regular resolution refutations of $F(T)$ consisting of at most $n(2^e - 1)$ steps.*

PROOF. Contract edges, beginning with the leaves of T. By Lemma 26, the number of resolution steps required to contract the edge connecting a leaf (of degree 1) with its parent is 2^{e-1}. For that parent, there are respectively 2^{e-2}, 2^{e-3}, etc. many steps in contracting sibling leaves with the same parent. Thus an upper bound for the number of resolution steps required to derive the empty clause is the number of internal nodes of T times $\sum_{i=0}^{e-1} 2^i$, so bounded by $n(2^e - 1)$. □

Hence, it seems possible that one could classify via graph theoretic properties those families of bounded degree connected graphs whose associated Tseitin formulas have polynomial size regular resolution refutations. Noting the correspondence in regular resolution with operations in the definition of graph minor, certain techniques may be applicable from the Robertson-Seymour theorem that the collection of all finite graphs is well quasi-ordered under graph minors.

As a small step in this direction, we describe degree 3 graphs whose Tseitin formulas correspond to the pigeonhole principle, and for which there

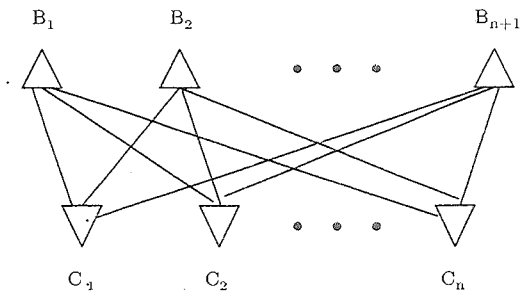


FIGURE 2. Global view of Tseitin graph representing PHP_n

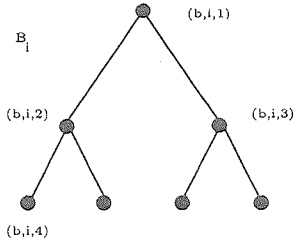


FIGURE 3. Local view of Tseitin graphs for the B's

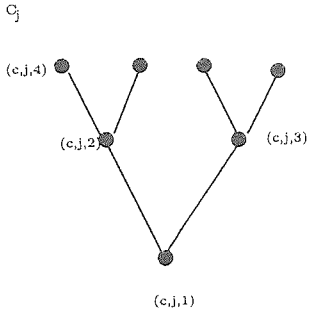


FIGURE 4. Local view of Tseitin graphs for C's

are no polynomial size resolution (or even constant depth Frege) proofs. As shown in Figure 2, the graph G_n consists of $n+1$ top trees B_i, and n bottom trees C_j, both indicated by triangles. B_i is responsible for mapping the i-th pigeon, and C_j is responsible for the j-th hole. The jth leaf of B_i is connected with the ith leaf of C_j. The B's themselves are shown in Figure 3 and the C's are shown in Figure 4. The trees B_i have n leaves each, therefore $2n-1$ nodes, whereas the trees C_i have $n+1$ leaves and $2n+1$

nodes. The jth leaf of B_i has number $\mathrm{leaf}_B(j) := n-1+j$ and the ith leaf of C_j has number $\mathrm{leaf}_C(i) := n+i$. Therefore G_n has vertices

$$\{\langle b,i,j\rangle : 1 \le i \le n+1, 1 \le j \le 2n-1\} \cup \{\langle c,i,j\rangle : 1 \le i \le n, 1 \le j \le 2n+1\}$$

and edges from the following three sets:

$$\{\{\langle b,i,\lfloor j/2\rfloor\rangle, \langle b,i,j\rangle\} : 1 \le i \le n+1, 2 \le j \le 2n-1\}$$

and

$$\{\{\langle c,i,\lfloor j/2\rfloor\rangle, \langle c,i,j\rangle\} : 1 \le i \le n, 2 \le j \le 2n+1\}$$

and

$$\{\{\langle b,i,\mathrm{leaf}_B(j)\rangle, \langle c,j,\mathrm{leaf}_C(i)\rangle\} : 1 \le i \le n+1, 1 \le j \le n\}.$$

We assign propositional variables to the edges of G_n as follows: $p_{\langle i,\lfloor j/2\rfloor\rangle,\langle i,j\rangle}$ labels the edge from $\lfloor j/2\rfloor$ to j in the tree B_i; $q_{\langle i,\lfloor j/2\rfloor\rangle,\langle i,j\rangle}$ labels the edge from $\lfloor j/2\rfloor$ to j in the tree C_i; and $r_{\langle i,\mathrm{leaf}_B(j)\rangle,\langle j,\mathrm{leaf}_C(i)\rangle}$ is responsible for connecting leaf j of B_i to leaf i of C_j. Here, the propositional variable sets are given as follows:

$$\{p_{\langle i,\lfloor j/2\rfloor\rangle,\langle i,j\rangle} : 1 \le i \le n+1, 2 \le j \le 2n-1\}$$

and

$$\{q_{\langle i,\lfloor j/2\rfloor\rangle,\langle i,j\rangle} : 1 \le i \le n, 2 \le j \le 2n+1\}$$

and

$$\{r_{\langle i,\mathrm{leaf}_C(j)\rangle,\langle j,\mathrm{leaf}_B(i)\rangle} : 1 \le i \le n+1, 1 \le j \le n\}.$$

Let each of $\langle b,i,1\rangle$ for $1 \le i \le n+1$, and $\langle c,j,1\rangle$ for $1 \le j \le n$ have charge 1, and all other nodes have charge 0. The edges of G_n are labeled in the obvious manner (eg. $\{\langle b,i,\lfloor j/2\rfloor\rangle, \langle b,i,j\rangle\}$ is labeled by $p_{\langle i,\lfloor j/2\rfloor\rangle,\langle i,j\rangle}$ etc.). Therefore G has odd charge $2n+1$ and we have the following charge equations:

$$p_{\langle i,1\rangle,\langle i,2\rangle} \oplus p_{\langle i,1\rangle,\langle i,3\rangle} = 1 \tag{1}$$

for all $1 \le i \le n+1$,

$$q_{\langle i,1\rangle,\langle i,2\rangle} \oplus q_{\langle i,1\rangle,\langle i,3\rangle} = 1 \tag{2}$$

for all $1 \le i \le n$,

$$p_{\langle i,\lfloor j/2\rfloor\rangle,\langle i,j\rangle} \oplus p_{\langle i,j\rangle,\langle i,2j\rangle} \oplus p_{\langle i,j\rangle,\langle i,2j+1\rangle} = 0 \tag{3}$$

for all $1 \le i \le n+1, 2 \le j \le n-1$,

$$q_{\langle i,\lfloor j/2\rfloor\rangle,\langle i,j\rangle} \oplus q_{\langle i,j\rangle,\langle i,2j\rangle} \oplus q_{\langle i,j\rangle,\langle i,2j+1\rangle} = 0 \tag{4}$$

for all $1 \le i \le n, 2 \le j \le n$,

$$p_{\langle i,\lfloor \frac{\mathrm{leaf}_B(j)}{2}\rfloor\rangle,\langle i,\mathrm{leaf}_B(j)\rangle} \oplus r_{\langle i,\mathrm{leaf}_B(j)\rangle,\langle j,\mathrm{leaf}_C(i)\rangle} = 0 \tag{5}$$

for all $1 \le i \le n+1$, and $1 \le j \le n$,

$$r_{\langle i,\mathrm{leaf}_B(j)\rangle,\langle j,\mathrm{leaf}_C(i)\rangle} \oplus q_{\langle j,\lfloor \frac{\mathrm{leaf}_C(i)}{2}\rfloor\rangle,\langle j,\mathrm{leaf}_C(i)\rangle} = 0 \tag{6}$$

for all $1 \le i \le n+1$, and $1 \le j \le n$.

The set of G_n's vertices has cardinality $(n+1)(2n-1)+n(2n+1) = 0(n^2)$ and each node has degree at most 3.

We claim that there are polynomial size resolution derivations of $F(G_n)$ from $\neg PHP_n$ – recall that the Tseitin formula $F(G_n)$ is refutable, whereas PHP_n is a tautology.

To this end, for $1 \le i \le n+1, 2 \le j \le 2n-1$ let

$$p_{\langle i,\lfloor j/2\rfloor\rangle,\langle i,j\rangle} \equiv \bigvee_{\{1\le k\le n : j \sqsubseteq \mathrm{leaf}_B(k)\}} P_{i,k} \tag{7}$$

where $u \sqsubseteq v$ means that u is a prefix of v (integer u is a prefix of integer v if the binary representation of u is a prefix of the binary representation of v). The idea is that the leaves of the tree B_{i_0} are labeled by $P_{i_0,1}, P_{i_0,2}, \ldots, P_{i_0,n}$ and that an edge $p_{\langle i,\lfloor j/2\rfloor\rangle,\langle i,j\rangle}$ between node $\langle b,i,\lfloor j/2\rfloor\rangle$ and node $\langle b,i,j\rangle$ of B_{i_0} has the value 1 iff $\langle b,i,\lfloor j/2\rfloor\rangle$ is the ancestor of a leaf bearing the value 1. For $1 \le j \le n, 2 \le i \le 2n+1$ let

$$q_{\langle j,\lfloor i/2\rfloor\rangle,\langle j,i\rangle} \equiv \bigvee_{\{1\le k\le n+1 : i \sqsubseteq \mathrm{leaf}_C(k)\}} P_{k,j} \tag{8}$$

Similarly, the idea is that the leaves of the tree C_{j_0} are labeled by $P_{1,j_0}, P_{2,j_0}, \ldots, P_{n+1,j_0}$ and that an edge $q_{\langle j,\lfloor i/2\rfloor\rangle,\langle j,i\rangle}$ between internal nodes of C_{j_0} has the value 1 iff $\langle c,j,\lfloor i/2\rfloor\rangle$ is the ancestor of a leaf bearing the value 1. For $1 \le i \le n+1, 1 \le j \le n$ let

$$r_{\langle i,\mathrm{leaf}_B(j)\rangle,\langle j,\mathrm{leaf}_C(i)\rangle} \equiv P_{i,j}. \tag{9}$$

The function of the r's is to connect up leaves of the B's with those of the C's, where leaf the j-th leaf of B_i (labeled by $P_{i,j}$) is connected to the i-th leaf of C_j. Finally, let $DEF(G_n)$ be the resolution clauses corresponding (as in Section 4) to the above definitions.

DEFINITION 29. *The negation of the onto-version of the pigeonhole principle, denoted $\neg PHP_n^{onto}$, states that there is a bijection from $\{1,\ldots,n+1\}$ onto $\{1,\ldots,n\}$, and is given by the following clauses:*

$$\{\{P_{i,1},\cdots,P_{i,n}\} : 1 \le i \le n+1\} \cup \{\{\overline{P}_{i,j}, \overline{P}_{i',j}\} : 1 \le i < i' \le n+1, 1 \le j \le n\}$$

together with

$$\{\{P_{1,j},\cdots,P_{n+1,j}\} : 1 \le j \le n\} \cup \{\{\overline{P}_{i,j}, \overline{P}_{i,j'}\} : 1 \le i \le n+1, 1 \le j < j' \le n\}.$$

Note that the onto version of the pigeonhole principle requires a bijection (not simply injection) from the domain of size $n+1$ to range of size n. In contrast to the formalization of PHP_n, we have $1 \leq i \leq n+1$ and $1 \leq j \leq n$ to allow a simple correspondence with the trees B_i and C_j.

THEOREM 30. *There are polynomial size resolution derivations of $F(G_n)$ from $DEF(G_n)$ and $\neg PHP_n^{onto}$.*

PROOF. For $1 \leq j \leq n$, by Equation (7)

$$p_{\langle i,\lfloor \frac{\mathrm{leaf}_B(j)}{2} \rfloor\rangle,\langle i,\mathrm{leaf}_B(j)\rangle} \equiv P_{i,j}$$

and by Equation (9)

$$r_{\langle i,\mathrm{leaf}_B(j)\rangle,\langle j,\mathrm{leaf}_C(i)\rangle} \equiv p_{\langle i,\lfloor \frac{\mathrm{leaf}_B(j)}{2} \rfloor\rangle,\langle i,\mathrm{leaf}_B(j)\rangle}.$$

This proves Equation (5). Equation (6) is similarly derived. Thus we've established the charge equations for the connection between appropriate trees B_i to C_j.

Fix $1 \leq i \leq n+1$. Equations (1) and (3) respectively correspond to the charge equation at the root and the charge equations at non-root internal nodes of the tree B_i. Consider first Equation (1). To simplify notation, write d resp. e in place of $p_{\langle i,1\rangle,\langle i,2\rangle}$ resp. $p_{\langle i,1\rangle,\langle i,3\rangle}$, and let $P_a, P_{a+1}, \ldots, P_b$ resp. $P_{b+1}, \ldots, P_c$ denote the leaf labels in B_i below d resp. e (hence the P's correspond to appropriate $P_{i,j}$'s). With this notation, Equation (7) states that $d \equiv P_a \vee \cdots \vee P_b$ and $e \equiv P_{b+1} \vee \cdots \vee P_c$, so from its clausal form in $DEF(G_n)$,

$$\{\overline{d}, P_a, \ldots, P_b\}.$$

Repeatedly resolve this clause against clauses

$$\{\overline{P}_a, \overline{P}_{b+1}\}, \{\overline{P}_{a+1}, \overline{P}_{b+1}\}, \{\overline{P}_{a+2}, \overline{P}_{b+1}\}, \ldots, \{\overline{P}_b, \overline{P}_{b+1}\}$$

(these clauses come from $\neg PHP_n^{onto}$) to obtain $\{\overline{d}, \overline{P}_{b+1}\}$. In a similar fashion, obtain $\{\overline{d}, \overline{P}_{b+2}\}$, $\{\overline{d}, \overline{P}_{b+3}\}, \ldots$, $\{\overline{d}, \overline{P}_c\}$. From $DEF(G_n)$, we have $\{\overline{e}, P_{b+1}, \ldots, P_c\}$, so by repeated resolution against the previous clauses, we obtain $\{\overline{d}, \overline{e}\}$.

From $\neg PHP_n^{onto}$, we have $\{P_a, \ldots, P_c\}$, and from $DEF(G_n)$ we have $\{\overline{P}_a, d\}, \ldots$, $\{\overline{P}_b, d\}$ and $\{\overline{P}_{b+1}, e\}, \ldots$, $\{\overline{P}_c, e\}$. Repeatedly resolving the former against the latter, we obtain $\{d, e\}$. Now $\{d, e\}$ and $\{\overline{d}, \overline{e}\}$ form the clausal representation of the charge equation (1) $d \oplus e = 1$. In a similar fashion (using the onto part of the formulation of the pigeonhole principle) one can prove Equation (2). This concludes the derivation of charge equations for roots of the B_i and C_j.

Consider now Equation (3). To simplify notation, write d, e, f resp. for $p_{\langle i,\lfloor j/2 \rfloor\rangle,\langle i,j\rangle}$, $p_{\langle i,j\rangle,\langle i,2j\rangle}$, $p_{\langle i,j\rangle,\langle i,2j+1\rangle}$, and let $P_a, \ldots, P_b$ resp. $P_{b+1}, \ldots, P_c$ denote the leaves of tree B_i respectively below e, f. Thus the leaves below d are $P_a, \ldots, P_c$. The clausal representation of Equation (3) has the following clauses:

1. $\{\overline{d}, e, f\}$
2. $\{d, \overline{e}, f\}$
3. $\{d, e, \overline{f}\}$
4. $\{\overline{d}, \overline{e}, \overline{f}\}$.

The first clause is simple to obtain. From $DEF(G_n)$ we have $\{\overline{d}, P_a, \ldots, P_c\}$ and $\{\overline{P}_a, e\}$, $\{\overline{P}_{a+1}, e\}, \ldots$, $\{\overline{P}_b, e\}$ and $\{\overline{P}_{b+1}, f\}$, $\{\overline{P}_{b+2}, f\}, \ldots$, $\{\overline{P}_c, f\}$. By resolution of the former against the latter, we have $\{\overline{d}, e, f\}$. The second and third clauses are similarly derived. To obtain the fourth clause, proceed as follows. By $DEF(G_n)$, we have $\{\overline{e}, P_a, \ldots, P_b\}$ and from $\neg PHP_n^{onto}$ we have $\{\overline{P}_a, \overline{P}_{b+1}\}$, $\{\overline{P}_{a+1}, \overline{P}_{b+1}\}, \ldots$, $\{\overline{P}_b, \overline{P}_{b+1}\}$ and so by resolution we obtain $\{\overline{e}, \overline{P}_{b+1}\}$. Similarly, we obtain $\{\overline{e}, \overline{P}_{b+2}\}, \ldots$, $\{\overline{e}, \overline{P}_c\}$, and in an analogous fashion $\{\overline{f}, \overline{P}_a\}, \ldots$, $\{\overline{f}, \overline{P}_b\}$. By $DEF(G_n)$, we have $\{\overline{d}, P_a, \ldots, P_c\}$ and so by repeated resolution against the preceding clauses, we derive $\{\overline{d}, \overline{e}, \overline{f}\}$, as required. This establishes Equation (3). The derivation of Equation (4) is analogous. This completes our treatment of the charge equations for non-root internal nodes of trees B_i and C_j.

Thus we have a resolution derivation of $F(G_n)$ from $DEF(G_n)$ and $\neg PHP_n^{onto}$. Straightforward estimation shows that the sketched resolution proof is of polynomial size. □

COROLLARY 31. *There is an exponential lower bound for resolution (even constant depth Frege) refutations of $F(G_n)$.*

The corollary is immediate, since it is well-known that $\neg PHP_n^{onto}$ has an exponential size lower bound for resolution (and constant depth Frege) refutations. See [11] for details.

From this construction, one might think that for every unsatisfiable propositional formula H there is a related odd charged graph G, for which $H \to F(G)$ has a polynomial size resolution (or constant depth Frege) derivation. This however is false, unless $NP = co-NP$.

PROPOSITION 32. *For any polynomials p, q there exists an unsatisfiable propositional formula H such that for all odd charged graphs G of size at most $q(|H|)$, there is a resolution (or constant depth Frege, or Frege, etc.) derivation of $H \to F(G)$, where $F(G)$ is the Tseitin formula related to G.*

PROOF. If not, then we have an NP-procedure to test whether a formula ϕ is a tautology: $\phi \in$ TAUT iff $\neg\phi \notin$ SAT iff there is odd charged G of size $q(|\phi|)$ and a resolution (or constant depth Frege, or Frege, etc.) derivation of $\neg\phi \to F(G)$. □

7. Acknowledgements

We would like to thank A. Haken, J. Krajíček, P. Pudlák and A. A. Razborov for comments, and especially W. Degen, for relating his hierarchy results concerning the Degen principle, and for a preliminary copy of [7]. Thanks to J. Krajíček and P. Pudlák for a preliminary copy of their unpublished typescripts. Special thanks to J. Johannsen for sending his unpublished

manuscript, which raised the question of how Degen's principle relates to the pigeonhole principle, over constant depth Frege systems. Thanks as well to the referee for suggestions.

In his manuscript, Johanssen made some observations concerning $I\Delta_0$ provability that $D_{m+1,k} \to D_{m,k}$; since it follows from this paper, that $I\Delta_0$ proves the equivalence of the pigeonhole principle and Degen's principle, we have answered Johannsen's principal question. The resolution proofs were typeset using LaTeXmacros developed by S. Buss.

References

[1] R. Boppana and M. Sipser. The complexity of finite functions. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, pages 759–804. Elsevier, MIT Press, 1990. Elsevier (Amsterdam), MIT Press (Cambridge).

[2] S.R. Buss and P. Clote. Cutting planes, connectivity, and threshold logic. *Archive for Mathematical Logic*, 35:33–62, 1966.

[3] V. Chvátal. Edmonds polytopes and hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.

[4] P. Clote. Note on monotonic complexity of $2 - REF$. *Information Processing Letters*, 57:117–123, 1996.

[5] P. Clote. Cutting plane and Frege proofs. *Information and Computation*, 121(1):103–122, 1995.

[6] W. Cook, C.R. Coullard, and G. Turan. On the complexity of cutting plane proofs. *Discrete Applied Mathematics*, 18:25–38, 1987.

[7] J.W. Degen. Pigeonhole principles and choice principles. Typescript, Universität Erlangen-Nürnberg, 1995.

[8] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297 – 305, 1985.

[9] A. Haken and S.A. Cook. An exponential lower bound for the size of monotonic real circuits. Typeset manuscript, December 23, 1995.

[10] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *J. Discrete Mathematics*, 3, 1990. 255–265.

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

[12] J. Krajíček. Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. Typeset manuscript, submitted, 1995.

[13] P. Pudlák. Ramsey's theorem in bounded arithmetic. In E. Börger, editor, *Proceedings of* Computer Science Logic 1990. Springer Lecture Notes in Computer Science 553, 1992.

[14] P. Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. Typeset manuscript, 1995.

[15] G. S. Tseitin. On the complexity of derivation in propositional calculus. In J. Siekmann and G. Wrigtson, editors, *Automation of Reasoning*, volume 2, pages 466 – 483. Springer Verlag, 1983.

[16] A. Urquhart. Hard examples for resolution. *Journal of the Association of Computing Machinery*, 34(1):209 – 219, 1987.

[17] A. Urquhart. The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, 1:425–467, 1995.

[18] I. Wegener. *Complexity of Boolean Functions*. Teubner-Wiley, 1987.

INSTITUT FÜR INFORMATIK, LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN, OETTINGENSTRASSE 67, D-80538 MÜNCHEN, GERMANY
E-mail address: clote@informatik.uni-muenchen.de

DEPARTMENT OF MATHEMATICS, UPPSALA UNIVERSITY, P.O. BOX 480, S-751 06 UPPSALA, SWEDEN
E-mail address: setzer@math.uu.se

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

Descriptive Complexity and the W Hierarchy

Rodney G. Downey, Michael R. Fellows, and Kenneth W. Regan

ABSTRACT. The classes $W[t]$ of the Downey-Fellows *W hierarchy* are defined, for each t, by fixed-parameter reductions to the weighted-assignment satisfiability problem for *weft-t* circuits. This paper proves that for each $t \geq 1$, $W[t]$ equals the closure under fixed-parameter reductions of the class of languages L definable by formulas of the form $\phi = (\exists U)\psi$, where U is a set variable and ψ is a first-order formula in $\prod_t$ prenex form. This is a fixed-parameter analogue of Fagin's well-known characterization of NP by second-order existential formulas. An equivalent form of this result states that the fixed-parameter "slices" L_k of L are definable by a family $\{\phi_k\}$ of first-order formulas in $\sum_t$ prenex form, subject to the restriction that the quantifier blocks in ϕ_k after the leading existential block are independent of k. Whether this restriction can be removed is connected to open problems in other recent papers on the W hierarchy.

1. Parameterized Problems and the W Hierarchy

Many important and familiar problems have the general form

INSTANCE: An object x, a natural number k.

QUESTION: Does x have some property Π_k that depends on k?

For example, the NP-complete CLIQUE problem asks: given an undirected graph $G = (V, E)$ and natural number k, is there a subset $U \subseteq V$ of size k that forms a clique in G? The VERTEX COVER problem asks whether G has a vertex subset U of size k that covers every edge, while the DOMINATING SET problem asks whether there is a U of size k such that every vertex in $V \setminus U$ is adjacent to a member of U. Here k is called the *parameter*.

Formally, a *parameterized language* L is a subset of $\Sigma^* \times \mathbf{N}$. Via a simple bijective encoding from $\Sigma^* \times \mathbf{N}$ to Σ^*, one can identify a parameterized language with an ordinary language over Σ^*, but we prefer to emphasize parameterized languages as entities in their own right.

DEFINITION 1.1 ([**DF95**]). A parameterized language L is *fixed-parameter tractable* if there is a polynomial p, a function $f : \mathbf{N} \to \mathbf{N}$, and a Turing machine

1991 *Mathematics Subject Classification.* Primary: 03D15, 68Q15, 68Q25. Secondary: 03D10, 03D20, 03D30, 03D80, 68R10.

Research supported by the New Zealand Marsden Fund for Basic Science.

Research supported by the National Science and Engineering Research Council of Canada.

Research supported by the U.S. National Science Foundation under grant CCR-9409104.

M such that on any input (x, k), M decides whether $(x, k) \in L$ within $f(k) \cdot p(|x|)$ steps. FPT stands for the class of fixed-parameter tractable languages.

This definition does not require f to be computable, and does not impose any limit on the growth rate of $f(k)$. When f is computable, L is said to belong to *strongly uniform* FPT. Languages L in FPT for which no f is computable are constructed in [**DF93**], while [**DF95**] gives *natural* problems in FPT for which no computable f is known. However, if (the ordinary language corresponding to) L belongs to P, then we can arrange that f is computable in time polynomial in the length (i.e., logarithm) of k, and further that $f(k)$ itself is polynomial in $\log k$. Thus P is contained in strongly-uniform FPT. In this paper we try to skirt the technicalities involving these and other notions of "uniformity" for FPT from [**DF95, DF93**], except to note that the issues are roughly similar to those of uniform versus non-uniform circuit classes.

If NP = P, then CLIQUE and the other two NP-complete languages belong to FPT. The converse, however, need not hold. Indeed, VERTEX COVER *does* belong to (strongly uniform) FPT, via an algorithm that runs in time $2^k \cdot O(n)$: Given a graph $G = (V, E)$, number V and E separately. To any string w in $\{0,1\}^k$, there corresponds a size-k subset V_w of V defined as follows: To choose the ith element of V_w, let e be the least-numbered edge not yet covered. If $w_i = 0$ then add the lower-numbered vertex on e to V_w, while if $w_i = 1$, then add the higher-numbered one. Then G has a vertex cover of size k iff one of the V_w forms such a cover, and the algorithm follows. (This has been credited to several people; see [**DF94**].) Note that f is exponential in k. Quite a few other NP-complete problems, with natural parameter k, are in FPT via algorithms of time $f(k) \cdot O(n)$ through $f(k) \cdot O(n^3)$—see the compendia in [**DF95, HW96**].

The best known method for solving the parameterized CLIQUE problem is an algorithm due to Nesetril and Poljak [**NP85**] that runs in time $O(n^{(\frac{2+\epsilon}{3})k})$, where $2 + \epsilon$ represents the exponent on the time for multiplying two $n \times n$ matrices (best known is $2.376\ldots$, see [**CW90**]). For DOMINATING SET we know of nothing better than the trivial $O(n^{1+k})$-time algorithm that tries all vertex subsets of size k, using $O(n)$ time per try. Many other problems listed in the appendix of [**DF95**] seem to be hard in the manner of CLIQUE and DOMINATING SET. In order to compare the difficulty of problems for fixed-parameter complexity, Downey and Fellows [**DF95**] took the time-honored route of defining appropriate notions of *reducibility* and *completeness*. For the reducibility relation, we select the "many-one" rather than the "Turing" kind of FPT-reductions to talk about in this paper.

DEFINITION 1.2 ([**DF95**]). A parameterized language A *FPT-many-one reduces* to a parameterized language B, written $A \leq_m^{fpt} B$, if there are a polynomial q, functions $f, g : \mathrm{N} \to \mathrm{N}$, and a Turing machine T such that on any input (x, k), T runs for $f(k) \cdot q(|x|)$ steps and outputs $(x', g(k))$ such that $(x, k) \in A \iff (x', g(k)) \in B$.

The reduction is *strongly uniform* if f is computable. Then (strongly uniform) FPT is closed downward under (strongly uniform) FPT reductions. Note that g is computable, and the parameter $k' = g(k)$ in the reduction does not depend on x.

DEFINITION 1.3. For any class $\mathcal{C}$ of languages, the downward closure of $\mathcal{C}$ under strongly uniform FPT many-one reductions is denoted by $(\mathcal{C})_{fpt}$.

For the completeness notion, Downey and Fellows [**DF95**] defined a hierarchy of classes of parameterized languages

$$\mathrm{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq W[poly], \tag{1.1}$$

and showed that the parameterized version of CLIQUE is complete for $W[1]$ under FPT reductions, while that of DOMINATING SET is complete for $W[2]$. This gives a sense in which DOMINATING SET is apparently harder than CLIQUE.

This paper provides a third support for parameterized complexity theory by giving a *descriptive logical characterization* of the $W[t]$ classes, one that is analogous to the important theorems of Fagin and Stockmeyer for NP and the polynomial-time hierarchy [**Fag74, Sto76**]. There are several twists that make our results interesting and not routine. First, they relate second-order formulas to sequences of first-order formulas in a way that shows a subtle effect of the parameter k. Second, our results open new ways to understand the W classes and solve problems about them, as we show with some preliminary applications in Section 5. Third, they hint at the possibility that the W hierarchy "collapses" to $W[2]$, by analogy with a "collapse" that is implicit in Fagin's theorem itself.

Although we have not yet defined the W hierarchy, we can convey the spirit of the results by expanding the CLIQUE and DOMINATING SET examples. For each k, the language L_k of graphs with a clique of size k is defined by the first-order existential formula

$$\phi_k := (\exists u_1 \ldots u_k) : \bigwedge_{i,j \leq k} E(u_i, u_j),$$

where $u_1 \ldots u_k$ range over vertices in V. That is, we have a sequence $[\phi_k]_{k=1}^{\infty}$ of $\sum_1^{\mathrm{FO}}$ formulas such that each ϕ_k defines the kth "slice" L_k of the parameterized language CLIQUE. Now we can also represent CLIQUE itself by the single second-order formula

$$\Phi = (\exists U)(\forall v, w)[(U(v) \ \wedge \ U(w) \ \wedge \ w \neq v) \to E(v, w)],$$

where U ranges over monadic relations on V—equivalently, over subsets of V. For standard languages this formula is trivial, since Φ holds for all graphs with $U = \emptyset$, but it faithfully represents the *parameterized* language CLIQUE under the following stipulation:

> For all k, the slice L_k equals the set of strings x (encoding graphs G_x) such that Φ holds for G_x with a set U of size k.

Note that Φ has the form $(\exists U)\varphi$ where φ is a $\prod_1^{\mathrm{FO}}$ formula, so we call Φ a $\mathrm{SO}\exists \cdot \prod_1^{\mathrm{FO}}$ formula.

DOMINATING SET is similarly represented by the sequence $[\psi_k]$ of $\sum_2^{\mathrm{FO}}$ formulas with

$$\psi_k := (\exists u_1 \ldots u_k)\ (\forall v) : \bigvee_{i \leq k} (v = u_i \ \vee \ E(v, u_i)),$$

and by the $\mathrm{SO}\exists \cdot \prod_2^{\mathrm{FO}}$ formula

$$\Psi = (\exists U)(\forall v)(\exists w)[U(v) \ \vee \ (U(w) \ \wedge \ E(v, w))].$$

Thus we can write that the $W[1]$-complete language CLIQUE belongs to $\mathrm{SO}\exists \cdot \prod_1^{\mathrm{FO}}$, and also to $\sum_1^{FO}$-*seq*. Similarly the $W[2]$-complete language DOMINATING SET

belongs to $\mathrm{SO}\exists \cdot \prod_2^{\mathrm{FO}}$, and also to $\sum_2^{FO}$-*seq*. The pattern is suggestive, and indeed the first main theorem of our paper is:

THEOREM 1.4. *For all* $t \geq 1$, $W[t] = \langle \mathrm{SO}\exists \cdot \prod_t^{\mathrm{FO}} \rangle_{fpt}$.

One is tempted to go on to conjecture a "yes" answer to the following question:

Problem. *For all* $t \geq 1$, *does* $W[t] = \langle \sum_t^{FO}\text{-}seq \rangle_{fpt}$?

However, this is not what we obtain. Rather, we obtain a first-order characterization of $W[t]$ by sequences of $\sum_t^{\mathrm{FO}}$ formulas ϕ_k that obey the following "niceness condition": for each of the $(t-1)$ quantifier blocks after the leading existential block, the number of variables quantified is independent of k. All $\sum_1$ families are trivially nice, so the $t = 1$ case of Problem 1 holds. The $\sum_2$ formulas ψ_k defining DOMINATING SET above are nice, since the "$(\forall v)$" block is the same for all of them. The second main theorem of this paper is:

THEOREM 1.5. *For all* $t \geq 1$, $W[t]$ *equals the* FPT*-closure of the class of parameterized languages defined by nice sequences of* $\sum_t^{\mathrm{FO}}$ *formulas.*

The "niceness" condition seems to arise by virtue of how the sequences of first-order formulas are related to the single second-order formulas in Theorem 1.4: The first existential block of a $\sum_t^{FO}$ formula ψ_k depends on k since it corresponds to the size-k set U in the $\mathrm{SO}\exists \cdot \prod_t^{\mathrm{FO}}$ formula $\Psi = (\exists U)\varphi$. The remaining $(t-1)$ quantifier blocks of ψ_k then correspond to blocks in the first-order part φ of Ψ, which are the same for all k. However, the correspondence is not so straightforward because φ has t-many quantifier blocks, so there is one left over, and even in the case of DOMINATING SET it takes some thought. In Section 5 we show how Theorem 1.5 classifies two problems related to DOMINATING SET that were previously not known to belong to $W[2]$.

In any event, the main open question of this paper becomes: What kinds of (parameterized) languages are represented by sequences of $\sum_t^{\mathrm{FO}}$ formulas with the niceness condition removed? A second important matter is raised by the link to Fagin's Theorem, which can be stated for standard languages as $\mathrm{NP} = \mathrm{SO}\exists \cdot \prod_2^{\mathrm{FO}} = \mathrm{SO}\exists \cdot \prod_t^{\mathrm{FO}}$ for all $t \geq 2$. By our Theorem 1.4, if a similar "$\mathrm{SO}\exists \cdot \prod_2^{\mathrm{FO}} = \mathrm{SO}\exists \cdot \prod_t^{\mathrm{FO}}$," collapse happens under the above stipulation for representations of parameterized languages, then $W[t] = W[2]$ for all $t \geq 2$. We note that the compendium [**HW96**] still lists no problems as $W[t]$-complete for $t \geq 3$, and indeed we have moved some problems from the "$W[2]$-hard" section into $W[2]$. For $t \leq 2$, connections similar to our Theorem 1.4 were observed by Cai and Chen [**CC93**], via the link between second-order formulas and optimization versions of NP problems that was first observed by Papadimitriou and Yannakakis [**PY91**]. We take all this to mean that the role of the parameter k deserves much further scrutiny, both at the level of NP-complete problems *and also* at the level of very-low-complexity languages defined by FO formulas.

Section 2 defines the W hierarchy, and Section 3 gives details on descriptive complexity and logic. Section 4 proves the above two theorems, and Section 5 gives applications and connections to open problems in other recent papers on the $W[t]$ classes.

2. Parameterized Circuit Complexity and the $W[t]$ Classes

Boolean circuits are said to be of *mixed type* if they may contain both *small* gates of fan-in ≤ 2 and *large* AND and/or OR gates of unbounded fan-in. We consider only *decision circuits*; i.e., those with a single output gate. The *weft* of such a circuit is the maximum number of large gates on a path from an input to the output. The n inputs are labeled by variables $x_1, \ldots, x_n$, and the *Hamming weight* $wt(x)$ of an assignment $x \in \{0,1\}^n$ equals the number of bits that are set to 1. The circuit is *monotone* if it has no NOT gates, and *anti-monotone* if all wires from an input go to a NOT gate, and these are the only NOT gates in the circuit. A pure $\sum_t$ circuit as defined by Sipser [**Sip83**] consists of t *levels* of large gates that alternate $\wedge$ and $\vee$ with a single $\vee$ gate at the top (i.e., the output), and with the bottom-level gates connected to the input gates $x_1, \ldots, x_n$ and their negations $\bar{x}_1, \ldots, \bar{x}_n$. A pure $\prod_t$ circuit is similarly defined with a large $\wedge$ gate at the output. In both cases, "pure" means that the circuit has no small gates. A Boolean expression is the same as a circuit in which each gate has fan-out 1. We call a Boolean expression *t-normalized* if it forms a pure $\prod_t$ circuit. For $t = 2$ this is the same as an expression in conjunctive normal form. For $t = 3$ this is product-of-sums-of-products (P-o-S-o-P) form; for $t = 4$ this is P-o-S-o-P-o-S form, and so on.

For all constants $h, t > 0$, the parameterized WEIGHTED CIRCUIT SATISFIABILITY problem is defined by:

$WCS(t,h)$
INSTANCE: A circuit C of weft t and overall depth $t + h$.
PARAMETER: k.
QUESTION: Does C accept some input of Hamming weight exactly k?

Then for all $t \geq 1$, $W[t]$ may be defined to be the class of parameterized languages A such that for some h, $A \leq_m^{fpt} WCS(t,h)$ (see [**BFH94**]). Also $W[poly]$ equals the class of problems that FPT many-one reduce to the problem WCS with no restriction on depth or weft. WCS is the parameterized version of the standard NP-complete CIRCUIT SATISFIABILITY problem, of which SAT is the specialization to the case where the circuit is a Boolean formula (in conjunctive normal form). An interesting aspect of the $W[\cdot]$ theory is that more-extreme special cases of the parameterized WCS problems remain complete. For all $t \geq 2$ define:

WEIGHTED t-NORMALIZED BOOLEAN EXPRESSION SATISFIABILITY ($WBES(t)$)
INSTANCE: A t-normalized Boolean expression E.
PARAMETER: k.
QUESTION: Is there a satisfying assignment to E of Hamming weight exactly k?

MONOTONE $WBES(t)$ ($MWBES(t)$)
Restriction of $WBES(t)$ to instances E that are *monotone*.

ANTI-MONOTONE $WBES(t)$ ($AWBES(t)$)
Restriction of $WBES(t)$ to instances E that are *anti-monotone*.

For $t = 1$, also define $AWBES(1,1)$ to be the restriction of $WCS(t,1)$ to instances consist of a single large AND gate, with input from a layer of binary OR gates, with the OR gates connected to negated inputs only.

THEOREM 2.1 ([**DF95, ADF93**], see also [**ADF95**]). (a) *For all* even $t \geq 2$, *MWBES*(t) *is complete for* $W[t]$ *under* $\leq_m^{fpt}$. *Hence so is WBES*(t).
(b) *For all* odd $t \geq 3$, *the problem AWBES*(t) *is complete for* $W[t]$ *under* $\leq_m^{fpt}$. *Hence so is WBES*(t).
(c) *The problem AWBES*$(1,1)$ *is complete for* $W[1]$ *under* $\leq_m^{fpt}$.

For $t = 1$, the extra level of small OR gates is necessary (unless $W[1] = \text{FPT}$) [**ADF93**]. The methods there and in Section 4 in [**ADF95**] remove this layer of small gates from earlier completeness proofs for odd $t \geq 3$.

We point out one important aspect of FPT reductions that strongly governs the *size* of the objects one can produce. Suppose $A \leq_m^{fpt} WCS(t,h)$, and take the polynomial q and functions $f, g : \text{N} \to \text{N}$ from Definition 1.2. Since T on input (x,k) must run in time $f(k)q(n)$ $(n = |x|)$, the circuits $C_{x,k}$ it produces have size polynomial in n for fixed k, and most important, the exponent of the polynomial is independent of k. Moreover, setting $k' = g(k)$, k' becomes the Hamming weight parameter for $C_{x,k}$ and is independent of x.

DEFINITION 2.2. A *parameterized family* of circuits is a bi-indexed family of circuits $\mathcal{F} = \{ C_{n,k} \}$ such that for some functions $f, g : N \to N$ and a polynomial q, each $C_{n,k}$ has n inputs and has size at most $f(k)q(n)$. We say that such a family is *FPT-uniform* if there is an algorithm to produce the circuit $C_{n,k}$ in time $O(q(n))$.

The idea of bounded Hamming weight in the weighted circuit satisfiability problems has been very successful in classifying many problems to belong to, and be complete for, the $W[t]$ classes [**DF95, ADF93, DF94**]. Our results support the assertion that Hamming weight is a "universal parameter."

3. $W[t]$ and Descriptive Complexity

The system of first-order logic for strings used by Immerman et al. [**Imm83, Imm87, BIS90**] to characterize the *log-time hierarchy*, and to provide a robust definition for "uniform AC^0," consists of the following:

- Constants 0,1, and n; n stands for the length of the string.
- A supply of first-order variables $u, v, w, \ldots$, which range over elements of the universe $V = \{ 1, \ldots, n \}$.
- In addition to the usual quantifiers and Boolean connectives, a special unary predicate symbol $X(\cdot)$, binary predicate symbols $=, \leq$, and ternary predicate symbols $PLUS(u,v,w)$ and $TIMES(u,v,w)$.

Given a sentence ϕ with these constituents, whether a binary string x belongs to the language L_ϕ is determined as follows. Instantiate the constant symbol n to be the length of x. Let us number the bits of x beginning with 1. Then the variables $u, v, w, \ldots$ run over the domain $\{ 1, \ldots, n \}$. An atom $X(u)$ is made true by an assignment that sets u to the value i iff the ith bit of x is 1. The truth values of the other atoms depend only on the assignment to the variables, not on the input x; e.g. $TIMES(u,v,w)$ is made true by any assignment of values to u, v that makes their product equal the value assigned to w. Hence each x induces a truth value on the sentence ϕ, and $x \in L_\phi$ iff that value equals *true*.

A formula in this system that has no quantifiers is called a "matrix," and is also called both a $\sum_0^{\text{FO}}$ formula and a $\prod_0^{\text{FO}}$ formula. For $t \geq 1$, a $\sum_t^{\text{FO}}$ formula has the form $(\exists u)\psi$, where ψ is either a $\prod_{t-1}$ formula or another $\sum_t^{\text{FO}}$ formula.

$\prod_t^{\text{FO}}$ formulas are similarly inductively defined in terms of $\sum_{t-1}^{\text{FO}}$ formulas, and are equivalent to the negations of $\sum_t^{\text{FO}}$ formulas. We may combine two or more adjacent quantifiers of the same kind into one *block*, e.g. writing $(\exists u, v)$ for $(\exists u)(\exists v)$. Then the "logical complexity" of the formula equals the minimum number of blocks needed to write (something equivalent to) the formula. The only *second-order* formulas we consider will have the form $\Phi = (\exists R)\phi$, where the second-order variable R ranges over relations on the universe V, and ϕ is a $\sum_t^{\text{FO}}$ or $\prod_t^{\text{FO}}$ formula for some t. The formula Φ is *monadic* if R is unary, so that Φ essentially quantifies over subsets of V.

The presence of the *PLUS* and *TIMES* predicates has two simplifying effects. First, it enables us to translate between formulas over *strings* and formulas over *graphs*, which have a binary relation $E(\cdot,\cdot)$ in place of $X(\cdot)$, without affecting the above notion of "logical complexity." Namely, encode an n-vertex graph G by its adjacency matrix in "row-major" order, yielding a binary string of length n^2. For each pair of vertex variables u, v, introduce a string variable w and maintain the property $w = (v-1) * n + u$. Then $E(u,v) \leftrightarrow X(w)$. This functional use of $+$ and $*$ can be represented via the *PLUS* and *TIMES* predicates with the help of some quantifiers that can be chosen to be either existential or universal. For all $t \geq 1$, a $\sum_t$ formula in the graph system is converted into a $\sum_t$ formula over strings, with extra existential quantifiers for the conversion. Thus to prove that a language encoding of a graph property belongs to $\sum_t^{\text{FO}(+,*)}$ for strings, it suffices to write a $\sum_t$ formula in the system for graphs themselves. Interestingly, it will suffice in our main results to use just E and $=$ as predicates for the graphs—*PLUS* and *TIMES* are only needed to relate the results to Immerman's systems for strings. Thus we are using "pure FO" for graphs, and this lends extra force to our writing "FO" in place of "FO$(+,*)$" in what follows.

Second, it is now known that FO$(+,*)$ is equivalent to the system FO$(BIT, \leq)$ originally used by Immerman, where $BIT(u,v)$ expresses that the vth bit of u in binary notation is a '1.' This follows from known tricks about encoding the graph of the exponential function via *PLUS* and *TIMES*, summarized by Lindell [**Lin94**] (see also [**Smo91, HP93**]), and from the result by Lindell [**Lin92**] that exponentiation suffices to simulate *BIT*.

Some of the features given above are redundant. For example, $=$ can be simulated via $\leq$, or alternatively $\leq$ via $=$ and *PLUS*. The constants 1 and n can be dispensed with by introducing a fresh variable v and adding the assertions '$(\forall w)[v \leq w]$,' respectively '$(\forall w)[w \leq v]$.' However, the above description is most convenient for our purposes.

These remarks apply also to second-order formulas, since the graph-to-string conversion can be bundled into the first-order parts of such formulas. For a second-order formula $\Phi = (\exists R)\phi$ in the language of graphs, R refers to an m-ary relation on the vertex set V, for some V. Our proofs going *from* $W[t]$ classes *to* formulas will produce formulas in the language of graphs. However, our proofs going *from* formulas ϕ_k or Φ *to* $W[t]$ classes will apply to formulas over any class of finite structures, including graphs and strings. This element of generality is best explained at the appropriate junctures of the proofs themselves.

DEFINITION 3.1. For all $t \geq 1$:

(a) A parameterized language L belongs to $\sum_t^{\mathrm{FO}}$-*seq* if there is a sequence of $\sum_t^{\mathrm{FO}}$ formulas ϕ_k such that for each k, ϕ_k defines the language $L_k = \{ x : (x,k) \in L \}$.

(b) The language L belongs to $\sum_t^{\mathrm{FO}}$-*nice* if in addition, all quantifier blocks in all ϕ_k after the leading existential block have size independent of k.

(c) The language L belongs to $\mathrm{SO}\exists \cdot \prod_t^{\mathrm{FO}}$ if there is an existential second-order formula Φ of the form $\Phi = (\exists R)\phi$, where R is an m-ary relation symbol (for some $m \geq 1$), ϕ is a $\prod_t$ first-order formula (naturally including the symbol R), and for all $k \geq 1$,

$$L_k = \{ x : \text{for some } \bar{R} \subseteq \{1,\ldots,n\}^m \text{ with } |\bar{R}| = k,\ x \text{ satisfies } \phi(\bar{R}) \}.$$

If Φ has the more-general form $(\exists R_1)(\exists R_2)\cdots(\exists R_\ell)\phi(R_1,\ldots,R_\ell)$, then we stipulate that L_k comprises those structures x (strings or graphs or etc.) that can satisfy $\phi(\bar{R}_1,\ldots,\bar{R}_\ell)$ for some choice of relations $\bar{R}_1,\ldots,\bar{R}_\ell$ such that $\prod_{i=1}^{\ell} |\bar{R}_i| = k$. However, *PLUS* and *TIMES* supply enough arithmetic to encode tuples of relations into a single relation, so we stay with the simpler form above. Indeed, the proofs of the two directions in our main theorems convert everything down to the case of a single monadic relation (which happens likewise for standard languages when *PLUS* and/or *TIMES* are present—see e.g. Lynch [**Lyn82**]). One interpretation of our results is that both the cardinality of a set and the Hamming weight of a string serve as a "universal parameter" for the W classes.

4. Main Results

Going from the $W[t]$ classes to the FPT-closures of the logic classes is relatively easy, since we need only find an appropriate logical specification of one $W[t]$-complete problem. The monotone and anti-monotone forms of the *WBES* problems are used crucially in this direction. We combine both the second-order and the first-order statements of our main results into one theorem and proof for each direction.

THEOREM 4.1. *For all $t \geq 1$, $W[t] \subseteq (\sum_t^{\mathrm{FO}}\text{-}nice)_{fpt}$, and also $W[t] \subseteq \langle \mathrm{SO}\exists \cdot \prod_t^{\mathrm{FO}} \rangle_{fpt}$.*

PROOF. First suppose t is even. We show that the $W[t]$-complete problem MONOTONE *WBES*(t) is definable by a family of $\sum_t$ formulas in FO. An instance C of this problem is a tree with edges directed away from the root r and n sink nodes. The root represents an AND gate at level 1, and is connected to a layer of OR gates at level 2, alternating down to a layer of OR gates at level t (since t is even), which in turn is connected to the positive-only inputs. The layering enables us to avoid having to refer to the label of a gate node at all, and it is not even necessary to quantify that the nodes $u_1,\ldots,u_k$ are sinks in the graph. More importantly, because the instance C is *monotone*, it is not necessary to add to the quantifier-free matrix of the formula a term of the form $\wedge_{i\neq j}(u_i \neq u_j)$ saying that the u_i are all distinct. The formula ϕ_k expressing that C has a satisfying assignment of Hamming weight k is:

$$
\begin{array}{l}
(\exists u_1,\ldots,u_k) \\
\quad (\forall v_2) : E(r,v_2) \rightarrow \\
\qquad (\exists v_3) : E(v_2,v_3) \wedge \\
\qquad\quad (\forall v_4) : E(v_3,v_4) \rightarrow \\
\qquad\qquad \ldots \\
\qquad\qquad\quad (\forall v_t) : E(v_{t-1},v_t) \rightarrow \\
\qquad\qquad\qquad \mathrm{E}(v_t,u_1) \vee E(v_t,u_2) \vee \ldots \vee E(v_t,u_k),
\end{array}
$$

which is equivalent to

$$
\begin{array}{l}
(\exists u_1,\ldots,u_k)(\forall v_2)(\exists v_3)\ldots(\forall v_t) : \\
\quad (\wedge_{i\neq j}(u_i \neq u_j)) \wedge \\
\quad E(r,v_2) \rightarrow E(v_2,v_3) \wedge [E(v_3,v_4) \rightarrow E(v_4,v_5) \wedge [\ldots \\
\quad [\ldots E(v_{t-3},v_{t_2}) \rightarrow E(v_{t-2},v_{t-1}) \wedge [E(v_{t-1},v_t) \rightarrow E(v_t,u_1) \vee \ldots \vee E(v_t,u_k)] \\
\quad]\ldots].
\end{array}
$$

This yields a family $\{\phi_k\}$ of $\sum_t^{\mathrm{FO}}$ sentences defining MONOTONE-*WBES*(t), one for each parameter k. The corresponding single second-order existential formula defining MONOTONE-*WBES*(t), per Definition 3.1(c), is

$$
\begin{array}{l}
\Phi = (\exists U)(\forall v_2)(\exists v_3)\ldots(\forall v_t)(\exists u) : \\
\qquad E(r,v_2) \rightarrow E(v_2,v_3) \wedge [E(v_3,v_4) \rightarrow E(v_4,v_5) \wedge [\ldots \\
\qquad [\ldots E(v_{t-3},v_{t_2}) \rightarrow E(v_{t-2},v_{t-1}) \wedge [E(v_{t-1},v_t) \rightarrow (U(u) \wedge E(v_t,u))] \\
\qquad]\ldots].
\end{array}
$$

Among the cases of t odd, we first consider the case $t = 1$, which involves the "extra" layer of small OR gates. An instance to ANTI-MONOTONE *WBES*$(1,1)$ has the form $C = \wedge_{j=1}^{m}(\bar{x}_{j_1} \vee \bar{x}_{j_2})$, where the subscripts j_1 and j_2 run over $\{1,\ldots,n\}$. This has a satisfying assignment of weight k iff there is a set $S \subseteq \{1,\ldots,n\}$ of size k such that for all j, at least one of j_1, j_2 does not belong to S. In terms of the graph, this says that every $\vee$ node has at most one link to S. Following the above scheme with a directed tree, this yields the condition that C has a satisfying assignment of Hamming weight k iff

$$(\exists u_1,\ldots,u_k)(\forall v)[E(r,v) \implies \wedge_{i\neq j}(\bar{E}(v,u_i) \vee \bar{E}(v,u_j))].$$

However, this has $\sum_2$ form, not $\sum_1$. The key is that there are at most $\binom{n}{2}$ distinct small $\vee$ gates, and we can convert C into a DAG C' that has exactly $\binom{n}{2}$ small $\vee$ gates. Not all of these $\vee$ gates may be connected to the $\wedge$ gate—and in the analogous C' for $t \geq 3$ the graph is no longer a directed tree—but the point is that the instances of ANTI-MONOTONE *WBES*$(t,1)$ in the proof of the main theorem of [**DF95**] can be given this C' form to begin with. Now (in the $t = 1$ case) we draw attention to the $\binom{k}{2}$ of these $\vee$ gates that have both of their input lines coming from S. The idea is to accept iff none of these gates is connected to the $\wedge$ gate at the root. It does not work to define ϕ_k to be the assertion "there exist $\binom{k}{2}$-many distinct nodes not connected to r," because there can be sets of $\binom{k}{2}$-many $\vee$ gates that link to more than k negated inputs, in a way that defeats the assertion that a weight-k satisfying assignment exists. (For the same reason, one cannot simply regard the $\vee$ gates as a virtual input layer for the anti-monotone formula.) Instead, taking $\ell = \binom{k}{2}$, we define ϕ_k to be

$(\exists u_1, \ldots, u_k, w_1, \ldots, w_\ell)$
[These $k + \ell$ nodes are all distinct and the $u_1, \ldots, u_k$ are all sinks and each w_i has exactly two out-links and these two belong to $\{ u_1, \ldots, u_k \}$ and $\wedge_{i=1}^{\ell} \bar{E}(r, w_i)$].

This does the job, and is a $\sum_1^{\mathrm{FO}}$ formula. The second-order formula is actually simpler:

$$(\exists U)(\forall u_1, u_2, w) : (U(u) \to \bar{E}(u, v)) \wedge (U(u_1) \wedge U(u_2) \wedge u_1 \neq u_2 \wedge E(w, u_1) \wedge E(w, u_2)) \to \bar{E}(r, w).$$

This says that all nodes in the set U are sinks, and all nodes w that have two out-edges into the set U are not connected to r. By above remarks on promises about the structures of the graphs in the theorem that ANTI-MONOTONE $WBES(1,1)$ is $W[1]$-complete, this suffices for $W[1] \subseteq \langle \mathrm{SO}\exists \cdot \prod_1^{\mathrm{FO}} \rangle_{fpt}$.

In the case of odd $t \geq 3$, level t consists of large $\wedge$ gates, and the quantified v_t from this level plays the role of r in the above. It turns out not to matter whether the instance circuit has the layer of small OR gates below the inputs or not; the only change to the first-order formulas above is:

$(\exists u_1, \ldots, u_k, w_1, \ldots, w_\ell)$ [...as above and..]
$(\forall v_2) : E(r, v_2) \to$
$(\exists v_3) : E(v_2, v_3) \wedge$
$\ldots$
$(\exists v_t) : E(v_{t-1}, v_t) \wedge$
$\wedge_{i=1}^{\ell} \bar{E}(v_t, w_i)]$.

This converts to $\sum_t^{\mathrm{FO}}$ form. The second-order formula is similar to before. As remarked in the last section, all of these formulas can be converted to equivalent ones of the same quantifier structure over strings rather than graphs, with the help of the *PLUS* and *TIMES* operations. □

Remarks: The above definitions of [anti-]monotone $WBES(t)$ do not include the formal definition of being t-normalized (nor of being [anti-]monotone); rather, we have treated this condition on the underlying graph as a "promise." Adding this still leaves a $\sum_t^{\mathrm{FO}}$ formula, however. The monotonicity of the instance to $WBES(t)$ for even t appears to be *vital* to this construction, and likewise the anti-monotonicity in the case of t odd. The existential second-order formulas obtained are *monadic* in the language of graphs, and can be converted to monadic formulas in the language of strings with help from the arithmetical operations *PLUS*, *TIMES*. However, for standard languages, monadic second-order existential formulas in the language of strings using just the ordering $\leq$ on string positions define only the class of regular languages (Büchi [**Büc60**]).

Now we show the converse directions:

THEOREM 4.2. (a) *Let L be a parameterized language defined by a uniform family $\{ \phi_k \}$ of $\sum_t^{\mathrm{FO}}$ formulas whose quantifier blocks after the leading existential block have size independent of k. Then $L \in W[t]$.*

(b) *Alternatively, let L be represented, per Definition 3.1(c), by a second-order existential formula $\Phi = (\exists R)\phi$, where ϕ is a single $\prod_i^{\mathrm{FO}}$ formula. Then $L \in W[t]$.*

PROOF. We need only show that $L \leq_m^{fpt} WCS(t, h)$ for some fixed h. We suppose that the formulas ϕ_k and Φ are written in the language of strings, using the fixed (interpreted) relational symbols $\{ X(\cdot), =, \leq, PLUS, TIMES \}$. It will, however, be clear how to modify this proof for formulas involving other fixed relations, such as $E(\cdot, \cdot)$ for graphs. We give a general construction that takes x and k and lays out a circuit $C_{x,k}$ of size $n^{O(1)} \cdot g(k)$ and the required form for $WCS(t, h)$ such that $C_{x,k}$ has a weight-$f(k)$ satisfying assignment iff x makes ϕ_k true; here the functions f and g come from the family $\{ \phi_k \}$ of formulas.

First we consider the case of t even, $t \geq 2$. By assumption, each formula ϕ_k has the form

$$(\exists u_1, \ldots, u_{f(k)})(\forall v_{2,1}, \ldots, v_{2,i_2}) \ldots (\forall v_{t,1}, \ldots, v_{t,i_t})\ M_k,$$

where $i_2, \ldots, i_t$ are constants independent of k, and M_k is a quantifier-free Boolean formula that can depend on k. In this even-t case, we write each M_k in conjunctive normal form (CNF); i.e., as an AND of clauses, where each clause is an OR of atoms, and each atom is a possibly-negated instance of one of the relations in $\{ X(\cdot), =, \leq, PLUS, TIMES \}$. Let $c(k)$ stand for the number of clauses in ϕ_k, and $a(k)$ for the maximum number of atoms in a clause. The functions $c(k)$ and $a(k)$ may be arbitrary (computable) functions; the main point is that these quantities do not depend on the length n of x.

The circuit $C = C_{x,k}$ that we build has $nf(k)$ input nodes, thought of as $f(k)$ rows of n nodes, one row for each variable in the leading existential quantifier block. It has circuitry involving one large $\wedge$ gate of size $f(k)\binom{n}{2}$ that enforces the condition that at most one input in each row is set to '1'; this entails that any weight-$f(k)$ satisfying assignment to C must specify exactly one value in the domain $\{ 1, \ldots, n \}$ for each variable.

Level 1 of the circuit is a single $\wedge$ gate with n^{i_2} fan-in lines. Each of the n^{i_2} $\vee$ gates at level 2 corresponds to a different assignment to the variables $v_{2,1}, \ldots, v_{2,i_2}$. Each of these $\vee$ gates has n^{i_3} fan-in lines, and so on in a tree down to the $\wedge$ gates at level $t-1$, each of which has fan-in lines to n^{i_t} $\wedge$ gates, called *matrix nodes*. The size of this part of the circuit is $O(n^{i_2 + \cdots + i_t})$. Now we concentrate on the $\wedge$ gates for the matrix nodes at level t.

Each of these $\wedge$ gates connects to $c(k)$ *CNF clause nodes*. Each CNF clause node is an $\vee$ gate with fan-in from at most $a(k)$ *atomic nodes*. Each atomic node A is an OR gate and represents an atomic formula of the form $X(v)$, $v = w$, $v \leq w$, $PLUS(u, v, w)$, or $TIMES(u, v, w)$, where u, v, and w are variables or constants. If A involves r of the variables $u_1, \ldots, u_{f(k)}$, then A has fan-in "from all r-tuples of the *input nodes* that make A true." To explain this, first if A does not involve any of the variables $u_1, \ldots, u_{f(k)}$, then it is determined completely by the given x and the assignments to $v_{21}, \ldots, v_{t,i_t}$ along the unique path to that node from the root, and so A is filled in as a logical constant. If A has just one variable u_i from the leading block, then it is an $\vee$ gate connected to precisely those nodes in input row i whose corresponding value of u_i makes the atom true. If it has u_i and u_j where $i \neq j$, then it is an $\vee$ gate with fan-in lines to $\wedge$ gates for each pair of input values for u_i, u_j that makes the atom true. If it has three different variables, u_i,

u_j, and u_k, then there is an $\wedge$ gate for each triple of values that makes the atom true. Note that $r \leq 3$ in our string case, so the fan-in to A is at most n^3.

This completes the description of C. The two layers of AND gates for level $t-1$ and the matrix nodes can be coalesced into one layer, as can the two layers of OR gates for clause and atomic nodes. This C is a $\prod_t$ circuit plus a layer of small ANDs right at the inputs. The size of C, counting the number of gates, is bounded by

$$n^{i_2+\cdots+i_t} \cdot a(k) \cdot c(k) \cdot n^3 \cdot 3 + nf(k),$$

which in turn is bounded for all n and k by an expression of the form $g(k)n^\alpha$ with α independent of k. Thus C is in the correct form for $WCS(t,h)$, and C has a weight-$f(k)$ satisfying assignment iff x makes ϕ_k true. Hence $L \in W[t]$.

For odd t, we write the matrices M_k in disjunctive normal form (DNF). Now each matrix node is an OR gate of fan-in $t(k)$, and sends output to an OR gate at level $t-1$. Each input to a matrix node is a *term node*, which is an AND gate from $a(k)$-many atomic nodes. The problem now is that we want each atomic node A to be an AND gate.

If A involves one variable u_i from the leading block, then let A be an AND gate with *negated* input lines to all of the values in row i of the $nf(k)$ input nodes that make $A(u_i)$ *false*. Since there is "extra circuitry" enforcing that exactly one node in row i is set equal to 1, this has the same effect as before. If A uses u_i and u_j, then let A have an input line for every pair of values in those two rows that makes $A(u_i, u_j)$ false, and use an OR of two negated input wires to the nodes for each such pair. The case of three (or any fixed number of) occurrences of the u_is in an atom is handled similarly. The size of C and the rest of the analysis is the same as before. Note that in the case $t = 1$, the "extra circuitry" is an AND of small ORs (again, with negated input lines), and so this does not prevent the whole circuit from belonging to $WCS(1,h)$.

(b) In the case where we are given a second-order formula $\Phi = (\exists R)\phi$, with $\phi \in \prod_t^{\text{FO}}$, ϕ has the form

$$(\forall v_{1,1}, \ldots, v_{1,i_1})(\exists v_{2,1}, \ldots, v_{2,i_2}) \ldots (Q_t v_{t,1}, \ldots, v_{t,i_t})\ M,$$

where Q_t is '$\exists$' if t is odd, Q_t is '$\forall$' if t is even, and the matrix M now has additional atoms of the form $R(v_{j_1,k_1}, \ldots, v_{j_m,k_m})$, where m is the arity of R. Now the circuit C has t, not $t-1$, levels at the top (beginning with an AND gate of fan-in n^{i_1} at the output) for the first-order variables. Things are actually simpler here insofar as there is no dependence of any first-order variables on the parameter k, and all assignments are determined by the path through the first t levels. Hence each atom other than $R(\cdots)$ is filled in as a constant. Only the nodes for each occurrence of $R(\cdots)$ in M are connected to the input nodes. There are n^m input nodes, one for each possible sequence of m values $a_1, \ldots, a_n$ to the arguments of R. Every assignment (of weight k) to these nodes determines a unique relation R (of size k). So for each occurrence of $R(v_{j_1,k_1}, \ldots, v_{j_m,k_m})$ below a given level-t node, we read off the values $a_1, \ldots, a_n$ to $v_{j_1,k_1}, \ldots, v_{j_m,k_m}$ from the path to that level-t node, and send a single input line to the corresponding input node. □

5. Applications and Conclusions

For an example of how these theorems can be used to classify problems in the $W[t]$ hierarchy, consider the following.

DOMINATING THRESHOLD SET
INSTANCE: An undirected graph $G = (V, E)$, and integers $k, r > 0$.
PARAMETER: k, r.
QUESTION: Is there a set U of at most k vertices such that for all $v \in V$, the neighborhood of v contains at least r elements of U?

This is defined by the family $\{\phi_{k,r}\}$ of $\sum_2^{\text{FO}}$ formulas given by

$$\phi_{k,r} = (\exists u_1, u_2, \ldots, u_k)(\forall v)\ M,$$

where

$$M = (\vee_{\text{distinct}}\ i_1, \ldots, i_r \in \{1, \ldots, k\})(\wedge_{q=1}^{r})[v = u_{i_q} \vee E(v, u_{i_q})].$$

Hence DOMINATING THRESHOLD SET belongs to $W[2]$. Since it was previously known to be $W[2]$-hard under $\leq_m^{fpt}$, this classifies it as $W[2]$-complete. The same result was obtained by Downey, Fellows, and Koblitz [**DF96, DFK96**] by other techniques.

For a second example, from the section of the compendium [**HW96**] titled "$W[2]$-Hard," consider

DOMINATING CLIQUE
INSTANCE: An undirected graph $G = (V, E)$, and an integer $k > 0$.
PARAMETER: k.
QUESTION: Is there a set U of at most k vertices that forms a clique in G, such that for all $v \in V$, there is an edge from v to a node in U?

This is defined by the sequence $[\phi_k]$ with

$$\phi_k = (\exists u_1, u_2, \ldots, u_k)(\forall v)\ M,$$

where

$$M = [\wedge_{1 \leq i < j \leq k} E(u_i, u_j)] \bigwedge [\vee_{1 \leq i \leq k} E(v, u_i)].$$

Since this is also a "nice" sequence of formulas, DOMINATING CLIQUE belongs to $W[2]$, and since it is $W[2]$-hard, this classifies it as $W[2]$-complete.

Finally, the problem of "internal quantifier blocks that depend on k" provided part of the motivation for the following work in [**DF96**]. For any sequence of numbers $(a_1, a_2, \ldots, a_r)$, where we intend each a_i to be either "k" or "n", define

$(a_1, a_2, \ldots, a_r)$-WEIGHTED SATISFIABILITY
INSTANCE: A Boolean expression F that consists of an AND of a_1-many ORs of a_2-many ANDs of ... of a_r literals.
PARAMETER: k.
QUESTION: Is there an assignment a of weight exactly k that satisfies F?

In particular, (n,k,n)-WSAT is the case where F is an n-ary AND of k-ary ORs, with each of the k-many inputs to each OR being an AND of n literals. As before, if all of the literals are positive, the formula F is *monotone*; if they are all negated, F is *anti-monotone*. Downey and Fellows [**DF96**], starting with a more-general circuit definition of a classes $W^*[t]$ that extend the definition of $W[t]$, show that for all $t \geq t$, $(n,k)^t$-WSAT is complete for $W^*[t]$ under $\leq_m^{fpt}$, so $W^*[t] = \langle (n,k)^t\text{-WSAT} \rangle_{fpt}$. By techniques similar to those from [**DF95**] that we referenced in Section 3, they show that anti-monotone (n,k,n,k)-WSAT remains $W^*[2]$-complete. Finally they show that anti-monotone (n,k,n,k)-WSAT is in $W[2]$, thus proving $W^*[2] = W[2]$. That $W^*[1] = W[1]$ was shown in [**DFT96**]. They inquire whether $W^*[t] = W[t]$ for all t.

We can show that anti-monotone (n,k,n)-WSAT and (n,k,n,k)-WSAT belong to $W[2]$, by applying the proof (if not the statement) of Theorem 4.2. Let us represent an anti-monotone (n,k,n)-WSAT formula F by a leveled graph as before, with n nodes for the negated inputs at level 0, and a single node at level 3 for the n-ary output AND gate. Intuitively let "w" range over the k-ary OR gates at level 2, "v" over the n-ary AND gates at level 1, and "u" over the inputs. Then the formula F has a satisfying assignment of Hamming weight k if and only if

$$(\exists u_1, \ldots, u_k)(\forall w)(\text{"let } v_1, \ldots, v_k \text{ be the } k \text{ nodes connected to } w\text{"})M,$$

where M is the Boolean matrix $\vee_{i=1}^{k} \wedge_{j=1}^{k} [\neg E(v_i, u_j)]$. Except for the funny "let" construct, this yields a "nice" sequence of $\sum_2^{\text{FO}}$ formulas, since the $(\forall w)$ part is unchanged. Because the $v_1, \ldots, v_k$ are uniquely determined once w is fixed, the presence of the "let" does not affect the counting in the proof of Theorem 4.2, and the conclusion that (n,k,n)-WSAT belongs to $W[2]$ follows. The case of (n,k,n,k)-WSAT is analogous to how the "extra" level of OR gates was handled in the $t = 1$ case of Theorem 4.1—both the leading existential block and the matrix become larger and more complicated, but the form is the same.

For $t \geq 3$, however, the similar treatment of (anti-)(monotone) $(n,k)^t$-WSAT leads to formulas with internal quantifiers that depend on k, where this dependence carries through on attempting to interchange quantifier blocks and reduce the formula. Thus the problem of $\sum_t^{\text{FO}}$ formulas that are not "nice" relates to the open problems in [**DF96**]. There appear to be other connections to the fixed-parameter hierarchies over constant-depth circuit classes defined in [**DFR96**]. Finally, we note that Cai, Chen, Downey, and Fellows [**CCDF94**] have characterized the $W[t]$ classes via a certain model of alternating log-time Turing machines. By well-known connections between these machines and constant-depth circuit classes, we feel there *should be* a closer connection between the $W[t]$ classes and uniform AC^0 than we have shown, such as might follow from removal of the "niceness" restriction in our main results. All of this points to interesting new challenges posed by the parameter k in parameterized complexity, ones that may also yield much new knowledge about "standard" complexity classes.

Acknowledgments. The third author would like to thank Drs. Paul Beame and Sam Buss for their invitation to speak at the April 1996 DIMACS Workshop on "Feasible Arithmetics and Lengths of Proofs," and some of the participants, especially Søren Riis, for relevant conversations. The main theorems were improved between Exits 141 and 130 on the Garden State Parkway on the workshop's second day.

References

[ADF93] K. Abrahamson, R. Downey, and M. Fellows, *Fixed-parameter intractability II: On completeness for W[1]*, Proc. 10th Annual Symposium on Theoretical Aspects of Computer Science, Lect. Notes in Comp. Sci., vol. 665, Springer Verlag, 1993, pp. 374–385.

[ADF95] K. Abrahamson, R. Downey, and M. Fellows, *Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogs*, Annals of Pure and Applied Logic **73** (1995), 235–276.

[BFH94] H. Bodländer, M. Fellows, and M. Hallett, *Beyond NP-completeness for problems of bounded width: hardness for the W hierarchy*, Proc. 26th Annual ACM Symposium on the Theory of Computing, 1994, pp. 449–458.

[BIS90] D. Mix Barrington, N. Immerman, and H. Straubing, *On uniformity within NC^1*, J. Comp. Sys. Sci. **41** (1990), 274–306.

[Büc60] J. Büchi, *Weak second-order arithmetic and finite automata*, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik **6** (1960), 66–92.

[CC93] L. Cai and J. Chen, *Fixed-parameter tractability and approximation of NP-hard optimization problems*, Proc. 2nd Israel Symp. on Theory of Computing and Systems, 1993, pp. 118–126.

[CCDF94] L. Cai, J. Chen, R. Downey, and M. Fellows, *On the structure of parametrized problems in NP*, Proc. 11th Annual Symposium on Theoretical Aspects of Computer Science, Lect. Notes in Comp. Sci., vol. 775, Springer Verlag, 1994, pp. 509–520.

[CW90] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetical progressions*, J. Symbolic Computation **9** (1990), 251–280.

[DF93] R. Downey and M. Fellows, *Fixed-parameter tractability and completeness III: Some structural aspects of the W hierarchy*, Complexity Theory: Current Research (K. Ambos-Spies, S. Homer, and U. Schöning, eds.), Cambridge Univ. Press, 1993, pp. 191–226.

[DF94] R. Downey and M. Fellows, *Parameterized computational feasibility*, Feasible Mathematics II, Birkhäuser, 1994, to appear.

[DF95] R. Downey and M. Fellows, *Fixed-parameter tractability and completeness I: Basic theory*, SIAM J. Comput. **24** (1995), 873–921.

[DF96] R. Downey and M. Fellows, *Threshold dominating sets and an improved characterization of W[2]*, 1996, Draft paper.

[DFK96] R. Downey, M. Fellows, and N. Koblitz, *Techniques for exponential parameterized reductions in vertex-set problems (incomplete draft)*, February 1996.

[DFR96] R. Downey, M. Fellows, and K. Regan, *Parameterized circuit complexity and the W hierarchy*, Theor. Comp. Sci., Ser. A (1996), to appear.

[DFT96] R. Downey, M. Fellows, and U. Taylor, *The complexity of relational database queries and an improved characterization of W[1]*, Proceedings of DMTCS'96, Springer Verlag, 1996, to appear.

[Fag74] R. Fagin, *Generalized first-order spectra and polynomial-time recognizable sets*, Complexity of Computation: Proceedings of a Symposium in Applied Mathematics of the American Mathematical Society and the Society for Industrial and Applied Mathematics, Vol. VII (R. Karp, ed.), SIAM-AMS, 1974, pp. 43–73.

[HP93] P. Hájek and P. Pudlák, *Metamathematics of first-order arithmetic*, Springer Verlag, 1993.

[HW96] M. Hallett and H..T. Wareham, *A compendium of parameterized complexity results, version 2.0*, May 1996, URL: `http://www-csc.uvic.ca/home/harold/W_hier/compendium.html`.

[Imm83] N. Immerman, *Languages which capture complexity classes*, Proc. 15th Annual ACM Symposium on the Theory of Computing, 1983, pp. 347–354.

[Imm87] N. Immerman, *Languages which capture complexity classes*, SIAM J. Comput. 16 (1987), 760–778.

[Lin92] S. Lindell, *The invariant problem for binary string structures and the parallel complexity theory of queries*, J. Comp. Sys. Sci. 44 (1992), 385–410.

[Lin94] S. Lindell, *How to define exponentiation from addition and multiplication in first-order logic on finite structures*, 1994, Unpublished manuscript.

[Lyn82] J. Lynch, *Complexity classes and theories of finite models*, Math. Sys. Thy. 15 (1982), 127–144.

[NP85] J. Nesetril and S. Poljak, *On the complexity of the subgraph problem*, Commen. Math. Univ. Carol. **26** (1985), 415–419.
[PY91] C. Papadimitriou and M. Yannakakis, *Optimization, approximation and complexity classes*, J. Comp. Sys. Sci. **43** (1991), 425–440.
[Sip83] M. Sipser, *Borel sets and circuit complexity*, Proc. 15th Annual ACM Symposium on the Theory of Computing, 1983, pp. 61–69.
[Smo91] C. Smoryński, *Logical number theory I: An introduction*, Springer Verlag, 1991.
[Sto76] L. Stockmeyer, *The polynomial time hierarchy*, Theor. Comp. Sci. **3** (1976), 1–22.

Department of Mathematics, P.O. Box 600, Victoria University, Wellington, New Zealand
E-mail address: downey@maths.vuw.ac.nz

Department of Computer Science, University of Victoria, Victoria, British Columbia, V8W 3P6, Canada
E-mail address: mfellows@csr.uvic.ca

Department of Computer Science, State University of New York at Buffalo, 226 Bell Hall, UB North Campus, Buffalo NY 14260-2000 USA
E-mail address: regan@cs.buffalo.edu

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

Lower Bounds on Sizes of Cutting Planes Proofs for Modular Coloring Principles

Xudong Fu

Abstract. The modulo r coloring principle states that for any partition of a set of size n into groups of size r and any r-coloring of the set such that each group containing exactly one element of each color, then each color occurs at most $\frac{n}{r}$ times. Pudlák showed an exponential lower bound on the length of cutting planes proof system by reducing the lower bound proof to the proof of lower bound on the size of monotone circuits over the reals for Clique. In this paper we use the same reduction and show an $n^{\Omega(\log n)}$ lower bound on the length of cutting planes proof systems for modulo r coloring principles by extending Razborov's well-known lower bound on the size of monotone boolean circuits that compute Perfect Matching. The lower bound is extended to monotone circuits over the reals that compute the same function by a different presentation of the approximation method for proving lower bounds on the size of monotone circuits.

1. Introduction

The complementary problem of the prototypical $\mathcal{NP}$-complete problem SAT formalized by Cook [Coo71], called the TAUT problem, is fundamental to logic and computer science. It was shown by Cook and Reckhow [CR79] that there exists an polynomially bounded propositional proof system for TAUT if and only if $\mathcal{NP} = co\text{-}\mathcal{NP}$. This equivalence theorem underlined an arduous task as $\mathcal{NP} \neq co\text{-}\mathcal{NP}$ implies the non-polynomiality of all proof systems; even *ZFC* would not be a polynomially bounded proof system for the propositional tautologies. Thus it motivates us to study the power of particular proof systems. Presently we only know some weaker systems are not polynomially bounded.

The cutting planes proof system originated from Chvátal's formalization of Gomory's cutting plane method in integer programming [Gom63]. In [Chv73], Chvátal introduced the cutting planes method as a canonical way of proving that every integral solution of a given system of linear inequalities satisfies another given inequality. The cutting planes proof system was first considered as a propositional refutation system for proving the unsatisfiability of formulae in the propositional calculus in [CCT87]. The completeness was shown in [Sch80]. Similar to resolution, the cutting planes refutation of a set of inequalities derives a sequence of

1991 *Mathematics Subject Classification.* Primary 03F20, 03D15; Secondary 68Q25.

inequalities from this set by several sound inference rules so that the final inequality is $0 \geq 1$. The cutting planes proof system can be viewed as a natural generalization of resolution, and it is more powerful. A short cutting planes proof for the propositional formula encoding the propositional pigeonhole principle is given in [CCT87]. Since the formula encoding pigeonhole principle is hard for both Resolution ([Hak85]) and bounded-depth Frege system ([Ajt88] and [BIK+92]), the cutting planes proof system cannot be polynomially simulated by either Resolution or bounded-depth Frege systems.

The complexity of proof systems is closely related to the complexity of circuits, especially for cutting planes proof systems. Pudlák [Pud95] showed a version of the interpolation theorem for cutting planes proof systems which states that the proving of lower bounds on the number of inequalities in the cutting planes proof can be reduced to the proving of lower bounds on the size of monotone circuits over the reals. In [Pud95], Pudlák extended Razborov's [Raz85b] lower bound for Clique to the monotone circuits over the reals. Haken and Cook [HC95] proved an exponential lower bound on the sizes of the monotone circuits over the reals that solve the Broken Mosquito Screen (BMS) problem. Thus the first exponential lower bound on the number of inequalities in any cutting planes proof of $0 \geq 1$ from the unsatisfiable set of inequalities based on the clique function or the Broken Mosquito Screen problem can be obtained. A similar idea appears in several earlier papers [BPR95][Kra94b][Kra94a]. A different presentation of the approximation method appeared in [Lee90]. Here we also give a different presentation.

Assume that n is a multiple of r. The modulo r coloring principle states that for any r-coloring of a set of size n and any partition of that set into groups of size r, each group containing exactly one element of each color, each color occurs at most $\frac{n}{r}$ times. This is related to the modulo r counting principle, which says, for n not divisible by r, that a set of size n cannot be partitioned into sets of size r. Ajtai [Ajt90] showed that any constant-depth Frege proof of the propositional formulas encoding the modulo 2 counting principles must have super-polynomial size even if the pigeon-hole principle is given as an axiom schema. This lower bound was improved to exponential [BP94]. Later Ajtai [Ajt94] proved that whenever p, q are two different primes, the modulo q counting principles do not have polynomial size constant-depth Frege proofs from instances of the modulo p counting principles. A different proof was given in [BIK+94]. Urquhart [Urq95] observed that there is a short cutting planes proof for the modulo r counting principle similar to that for the pigeon-hole principle.

We reduce the lower bound on the length of cutting planes proofs of $0 \geq 1$ from the contradictory set of inequalities encoding the negation of the modulo r coloring principle to the lower bound on the size of monotone circuits over the reals compute Perfect Matching. We extend Razborov's well-known lower bound on the size of monotone boolean circuits computing Perfect Matching to monotone circuits over the reals computing the same function.

2. Monotone circuits over the reals and cutting planes proofs

A *circuit over the reals* is a directed acyclic graph with the indegree of each node at most 2. Each node, which is called the gate, computes an arbitrary real-valued function. The inputs of the circuit are the nodes of indegree 0 which are either real-valued variables or constants. One of the node is designated the output

gate. The *size* of the circuit is the number of its gates. A circuit over the reals is called *monotone* if its gates compute non-decreasing real-valued functions. If the inputs of a circuit over the reals are 0's or 1's and the output of circuit is 0 or 1 then the circuit over the reals computes a boolean function.

The proof lines of the cutting planes proof system on variables $x_1, \ldots, x_n$ are the inequalities of the form

$$\sum_{i=1}^{n} a_i x_i \geq A$$

where $x_1, \ldots, x_n$ and A are integral constants. Note that in a cutting planes inequality the variables appear at the left hand side and the constant appears at the right hand side with the left hand side greater or equal to the right hand side. A conjunctive normal form formula can be translated into a inequality in cutting planes proofs by translating its clause $\{x_{j_1}, \ldots, x_{j_k}, \neg x_{l_1}, \ldots, \neg x_{l_m}\}$ as

$$\sum_{i=1}^{k} x_{j_i} + \sum_{i=1}^{m} (1 - x_{l_i}) \geq 1$$

together with the inequalities $x \geq 0$ and $-x \geq -1$ for each variable x in the formula. Thus the cutting planes proof system can be used as a refutation system for CNF formulas. A cutting planes refutation of a set of inequalities is a cutting planes proof for the inequality $0 \geq 1$ from this initial set of inequalities. The *length* of a cutting planes proof is the number of inequalities in it.

Cutting planes proof systems have three kinds of sound proof rules.

1. *Addition Rule:* From two inequalities

   $$\sum a_i x_i \geq B \text{ and } \sum b_i x_i \geq B,$$

   we can derive inequality

   $$\sum (a_i + b_i) x_i \geq A + B;$$

2. *Multiplication Rule:* Let c be a non-negative integer. Then from inequality $\sum a_i x_i \geq A$, we can derive inequality

   $$\sum (c a_i) x_i \geq cA;$$

3. *Division Rule:* Let c be a positive integer that divides each a_i. Then from inequality $\sum a_i x_i \geq A$, we can derive inequality

   $$\sum \frac{a_i}{c} x_i \geq \left\lceil \frac{A}{c} \right\rceil.$$

Let the implication $A(\vec{p}, \vec{q}) \to B(\vec{p}, \vec{r})$ be a tautology, where A and B have variables $\vec{p} = \{p_1, \ldots, p_k\}$ in common, while variables $\vec{q} = \{q_1, \ldots, q_l\}$ occur only in A and variables $\vec{r} = \{r_1, \ldots, r_m\}$ occur only in B. Then Craig's Interpolation Theorem ([Cra57a], [Cra57b]) says that there exists a boolean circuit $C(\vec{p})$ on variables $\vec{p}$ such that

$$A(\vec{p}, \vec{q}) \to C(\vec{p}) = 1 \text{ and } C(\vec{p}) = 1 \to B(\vec{p}, \vec{r})$$

are tautologically valid.

The following interpolation theorem was proved in [Pud95].

THEOREM 2.1 ([Pud95]). *Assume P is a cutting planes proof of the inequality $0 \geq 1$ from two set of inequalities $A(\vec{p},\vec{q}) = \{\sum_{j=1}^{k} a_{i,j}p_j + \sum_{j=1}^{m} b_{i,j}q_j \geq A_i | i \in I\}$ and $B(\vec{p},\vec{r}) = \{\sum_{j=1}^{k} c_{i,j}p_j + \sum_{j=1}^{n} d_{i,j}r_j \geq B_i | i \in J\}$, where $\vec{p} = \{p_1,\ldots,p_k\}$, $\vec{q} = \{q_1,\ldots,q_n\}$, and $\vec{r} = \{r_1,\ldots,r_m\}$ are disjoint sets of variables, I, J are some finite sets and $a_{i,j}$, $b_{i,j}$, $c_{i,j}$, $d_{i,j}$, A_i, $B_i \in \mathbb{Z}$. Assume $a_{i,j} \geq 0$ for $\forall i \in I$ and $1 \leq j \leq k$, or $c_{i,j} \leq 0$ for $\forall i \in J$ and $1 \leq j \leq k$. Then there exists a monotone circuit over the reals $C(\vec{p})$ with integer-valued gates of fin-in at most 2, with its size bounded by a polynomial in the number of inequalities in P, and with output 0 or 1 such that for $\vec{u} \in \{0,1\}^k$, $C(\vec{u}) = 0 \Longrightarrow A(\vec{u},\vec{q})$ is unsatisfiable, and $C(\vec{u}) = 1 \Longrightarrow B(\vec{u},\vec{r})$ is unsatisfiable.*

COROLLARY 2.2. *Under the assumptions of theorem 2.1, if we have $Y, Z \subseteq \{0,1\}^k$, such that for $\vec{u} \in Y$, $A(\vec{u},\vec{q})$ is satisfiable and for $\vec{u} \in Z$, $B(\vec{u},\vec{r})$ is satisfiable. Then the circuit $C(\vec{p})$ can be constructed such that $C(\vec{u}) = 1$ for $\vec{u} \in Y$ and $C(\vec{u}) = 0$ for $\vec{u} \in Z$ if $c_{i,j} \leq 0$ for $\forall i \in J$ and $1 \leq j \leq k$, or $C(\vec{u}) = 0$ for $\vec{u} \in Y$ and $C(\vec{u}) = 1$ for $\vec{u} \in Z$ if $a_{i,j} \geq 0$ for $\forall i \in I$ and $1 \leq j \leq k$.*

3. Modular Coloring Principles

Let n be a multiple of r, and $V = \{1,\ldots,n\}$ be a set of cardinality n. Let $[V]^r$ denote the set of r-element subsets of V. We shall formalize the negation of the modulo r coloring principle into the contradictory set of inequalities in the cutting planes proofs as follows.

With each $e \in [V]^r$, we associate a variable p_e. The variable p_e is true if and only if the set e appears in a complete partition of V into subsets of size r. Let $q_{i,j}$ denote that element $i \in V$ colored by color j $(1 \leq j \leq r)$. We assume that for each variable x below we have the inequalities $x \geq 0$ and $-x \geq -1$. Then the negation of the modulo r coloring principle is the following set of inequalities, and it is denoted by MCP_n^r.

$$\sum_{\forall e, i \in e} p_e \geq 1,\ 1 \leq i \leq n, \tag{1}$$

$$-p_e - p_f \geq -1,\ \forall e, f \in [V]^r, e \cap f \neq 0, e \neq f, \tag{2}$$

$$-p_e - q_{i,k} - q_{j,k} \geq -2,\ \forall e \in [V]^r, \forall i,j \in e, i \neq j, 1 \leq k \leq r, \tag{3}$$

$$\sum_{1 \leq j \leq r} q_{i,j} \geq 1,\ 1 \leq i \leq n, \tag{4}$$

$$-\sum_{1 \leq j \leq r} q_{i,j} \geq -1,\ 1 \leq i \leq n, \tag{5}$$

$$\sum_{1 \leq i \leq n} q_{i,1} \geq \frac{n}{r} + 1. \tag{6}$$

The set of inequalities from (1) says that each element from V must be in some r-element set in a complete partition of V into subsets of size r. The set of inequalities from (2) says that any two non-disjoint r-element sets must not appear in a partition simultaneously. The set of inequalities from (4) and (5) says that each element i must be colored by exact one color of the r colors. The set of inequalities from (3) says that if a r-element set appears in the partition then any two of the elements in the r-element set must be colored differently. Thus the set of inequalities from (1) to (5) expresses the fact that a complete partition of V into subsets of size r defines a partition of V into r subsets of equal size (the elements with the same color are in the same set). But the inequality (6) says that the r sets are not of the same size. Hence the set of above inequalities from (1) to (6) is contradictory.

The modular coloring principle can be translated into a CNF formula of size polynomial in n by translating each inequality from 1 to 5 into a clause with the method mentioned in previous section and translating inequality 6 into a set of clauses with limited extension.

4. Lower bounds to the cutting planes proofs for modular coloring principles

We let Inq^+ consist of all the inequalities from (1) and (2) together with the inequalities $p_e \geq 0$ and $-p_e \geq -1$ for each variable p_e ($e \in [V]^r$, and Inq^- consist of all the inequalities from (3), (4), (5) and (6) together with the inequalities $q_{i,j} \geq 0$ and $-q_{i,j} \geq -1$ for each variable $q_{i,j}$ $(1 \leq i \leq n, 1 \leq j \leq r)$. When we apply Theorem 2.2, we do not directly assign 0 or 1 to the common variables p_e of Inq^+ and Inq^-, instead we let each common variable equal a monotone function on a new set of variables $\{x_{\{i,j\}} : i,j \in V\}$ which is disjoint from the set of all the variables in $Inq^+ \cup Inq^-$. We let $p_e = \bigwedge_{i,j \in e} x_{\{i,j\}}$.

A graph is a structure $G = \langle V, E\rangle$ in which V is a finite set of nodes (vertices) and $E \subseteq V \times V$ is a finite set of edges (unordered pairs). We shall define two sets of input graphs Mat_r and Par_r on vertex set V for the variables $x_{\{i,j\}}$ $(i,j \in V)$, where the $x_{\{i,j\}}$ is the variable identified with the edge connecting vertices i and j in such a way that $x_{\{i,j\}} = 1$ if and only if there is an edge connecting vertices i and j.

We assume $n = rm$. Let $\mathsf{Mat}_r = \{G = \langle V, E\rangle | E = [B_1]^2 \cup \ldots \cup [B_m]^2, B_1 \ldots B_m$ is a complete partition of V into m subsets of size $r\}$, that is Mat_r is a set of graphs on vertex set V that contain m disjoint r-cliques such that there are no edges between any two cliques. Let $\mathsf{Par}_r = \{G = \langle V, E\rangle | E = [C_1]^2 \cup \ldots \cup [C_r]^2$, $C_1 \cup \ldots \cup C_r = V$ and $C_1,\ldots,C_r$ are disjoint and do not contain the same number of vertices $\}$, that is Par_r is a set of r-partite graphs on vertex set V such that not all the r parts contain the same number of vertices.

THEOREM 4.1. *Any cutting planes proof of $0 \geq 1$ from the contradictory set of inequalities MCP_n^r must have at least $n^{\Omega(\log n)}$ inequalities, where n is a multiple of r.*

Proof: Suppose there is a cutting planes proof P of $0 \geq 1$ from $Inq^+ \cup Inq^-$. To apply Theorem 2.2 for $Inq^+ \cup Inq^-$, let Inq^+ be the first part of the initial inequalities in Theorem 2.2, and let Inq^- be the second part of the initial inequalities in Theorem 2.2. Then the common variables $\vec{p}$ of Inq^+ and Inq^- are the variables p_e ($e \in [V]^r$). Let $p_e = \bigwedge_{i,j \in e} x_{i,j}$.

The set of inequalities Inq^+ is true if and only if the set $V = \{1,\ldots,n\}$ can be partitioned into family $\mathcal{V}$ of $\frac{n}{r}$ subsets of size r, i.e. for each $e \in \mathcal{V}$, $p_e = 1$, and for each $e \in [V]^r \backslash \mathcal{V}$, $p_e = 0$. Since $p_e = \bigwedge_{i,j \in e} x_{i,j}$, after an input G from Mat_r supplied to $x_{i,j}$, Inq^+ is true.

To check that Inq^- is satisfiable after an input G from Par_r supplied to $x_{i,j}$, we color all the elements that correspond numerically to the vertices in the part that contains the maximal number of vertices among all the r parts of the the vertices

in G with the color 1, and each of the rest of $r-1$ parts with a different color. Then clearly all the inequalities in Inq^- are satisfied.

Since all the coefficients of p_e in Inq^- are negative, by Theorem 2.2 there is a monotone circuit over the reals on input variables p_e ($e \in [V]^r$) such that the number of gates in circuit is polynomial in the number of inequalities in P. After replacing each p_e with $\bigwedge_{i,j\in e} x_{i,j}$, we obtain a monotone circuit over the reals on input variables $x_{i,j}$ ($i,j \in V$) that outputs 1 on all the graphs from Mat_2 and outputs 0 on all the graphs from Par_2. Clearly the number of gates in the new circuit is polynomial in the number of inequalities in P. Finally by Corollary 5.2 the theorem follows. ∎

Note that we are applying a trivial version of interpolation theorem 2.2 here. It is easy to see that the non-monotone formula Inq^+ can separate Mat_r and Par_r.

5. Lower bounds on the size of monotone circuit over the reals compute Perfect Matching

In this section, we extend Razborov's lower bound on the size of boolean circuits computing Perfect Matching to monotone circuits over the reals that compute the same function. We present a different approach of the approximation method for proving lower bounds on the size of monotone circuits. Similar presentation also appeared in [Fu95] and [Set96]. The sets of graphs Mat_2, Par_2, Mat_r and Par_r are defined in previous section.

THEOREM 5.1. *Any monotone circuit over the reals that outputs 1 on all graphs from* Mat_2 *and outputs 0 on all graphs from* Par_2 *must have at least* $n^{\Omega(\log n)}$ *gates.*

COROLLARY 5.2. *Any monotone circuit over the reals that outputs 1 o n all graphs from* Mat_r *and outputs 0 on all graphs from* Par_r *must have at least* $n^{\Omega(\log n)}$ *gates.*

Proof: We use induction on r. The base case is proved in Theorem 5.1.

For $r \geq 2$, assume any monotone circuit over the reals that outputs 1 on all graphs from Mat_r and outputs 0 on all graphs from Par_r must have at least $n^{\Omega(\log n)}$ gates, where n is the number of vertices in the input graph and is a multiple of r.

Suppose there exists a monotone circuit over the reals C with at most $m^{\Omega(\log m)}$ gates that outputs 1 on all graphs from Mat_{r+1} and outputs 0 on all graphs from Par_{r+1}, where m is the number of vertices in the input graph and is a multiple of $r+1$. Let $m = n + \frac{n}{r}$ and partition the vertices of the input graph for C into two parts V_1 and V_2 such that $|V_1| = n$ and $|V_2| = \frac{n}{r}$. Let the input variables corresponding to the edges between V_1 and V_2 set to 1 and the input variables corresponding to the edges within V_2 set to 0. Then C becomes a circuit C' over input graphs with vertex set V_1. If an input graph for C' is from Mat_r then the input graph on vertex set $V_1 \cup V_2$ for C is a graph that contains a graph from Mat_{r+1} and so C' outputs 1 since C outputs 1 by supposition, and if an input graph for C' is from Par_r then the input graph on vertex set $V_1 \cup V_2$ for C is from Par_{r+1} and so C' outputs 0 since C outputs 0 by supposition. But C' has at most $(n+\frac{n}{r})^{\Omega(\log(n+\frac{n}{r}))} = n^{\Omega(\log n)}$ gates, which contradicts the induction hypothesis. Thus the corollary follows. ∎

The following is devoted to proving Theorem 5.1. We present a different approach of the approximation method for proving lower bounds on the size of monotone circuits. Similar presentation also appeared in [Fu95] and [Set96].

First we shall define or redefine several notations for graphs that we will use in this section. Let $G = \langle V_1, E_1\rangle$ and $H = \langle V_2, E_2\rangle$ be two graphs. The inclusion $G \subseteq H$ means that G is a subgraph of H, that is $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$. The intersection $G \cap H$ is a graph on vertex set $V_1 \cap V_2$ with edge set $E_1 \cap E_2$. The union $G \cup H$ is a graph on vertex set $V_1 \cup V_2$ with edge set $E_1 \cup E_2$. The difference $G \backslash H$ is a graph on vertex set $V_1 \backslash V_2$ with edge set $E_1 \cap [V_1 \backslash V_2]^2$.

DEFINITION 5.3. Let $V = \{1, \ldots, n\}$ be the vertex set of graphs in Mat_2 and Par_2. Let $\mathcal{G} = \mathcal{G}_s$ denote the set of graph $G = \langle V_1, E_1\rangle$ with $V_1 \subseteq V$ and $|V_1| = 2s$. Let $\mathcal{M} = \mathcal{M}_s \subseteq \mathcal{G}_s$ be the set of graph with the degree of each vertex exactly 1, that is a graph $M \in \mathcal{M}_s$ is a partial matching of V with at most s edges. We define the relation $\vdash$ on $\mathcal{M}^r \times \mathcal{M}$ by saying $\langle M_1, \ldots, M_r\rangle \vdash M_0$ (where $M_i \in \mathcal{M}$, $0 \leq i \leq r$) if and only if $\bigcup_{1\leq i<j\leq r} M_i \cap M_j \subseteq M_0$ ∎

We shall define a set of monotone functions from $\mathcal{G}_s$ to the real numbers with respect to the subgraph inclusion.

DEFINITION 5.4. Let $\mathcal{F}_\mathcal{G}$ be the set of monotone functions $f : \mathcal{G} \to \mathbb{R}$ satisfying $P(r,s,f)$ and $Q(r,s,f)$ (the r-closed functions), where $P(r,s,f)$ if and only if for all $\langle M_0, \ldots, M_r\rangle \in \mathcal{M}^{r+1}$, if $\langle M_1, \ldots, M_r\rangle \vdash M_0$ then $\min\{f(M_1), \ldots, f(M_r)\} \leq f(M)$, and $Q(r,s,f)$ if and only if for all $E \in \mathcal{G}_s \backslash \mathcal{M}_s$, $f(E) = \max\{f(M) | M \in \mathcal{M}_s \text{ and } M \subseteq E\}$.

For $f \in \mathcal{F}_\mathcal{G}$, let $\hat{f} : \{0,1\}^{[V]^2} \to \mathbb{R}$, $\hat{f}(G) = \sup\{f(E) | E \in \mathcal{G}_s \text{ and } E \subseteq G\}$. ∎

For an arbitrary monotone function $f : \mathcal{M} \to \mathbb{R}$, the *closure* of f, denoted by f^*, is the minimal function from $\mathcal{F}_\mathcal{M}$ such that $f \leq f^*$. We will describe the procedure for defining f^* from f.

DEFINITION 5.5. Let $f : \mathcal{G} \to \mathbb{R}$. Let the sequence of monotone functions $g_i : \mathcal{G} \to \mathbb{R} (i \in \omega)$ as follows.

Let $g_0(M) = f(M)$ for $M \in \mathcal{M}$. If $P(r,s,g_i)$ then $g_{i+1} = g_i$. Assume $P(r,s,g_i)$ is false. Let $F_i = \{\langle G_1, \ldots, G_r\rangle \in \mathcal{M}^r | \exists M \in \mathcal{M}, \langle G_1, \ldots, G_r\rangle \vdash M \text{ and } g_i(M) < \min\{g_i(G_1), \ldots, g_i(G_r)\}$, $a_i = \max\{\min\{g_i(G_1), \ldots, g_i(G_r)\} | \langle G_1, \ldots, G_r\rangle \in F_i\}$, and $B_i = \{M \in \mathcal{M} | \exists \langle G_1, \ldots, G_r\rangle \in F_i, \langle G_1, \ldots, G_r\rangle \vdash M \text{ and } g_i(M) < \min\{g_i(G_1), \ldots, g_i(G_r)\}\}$. Let $M_i \in B_i$ such that no elements in B_i strictly contained in M_i. Then let

$$g_{i+1}(M) = \begin{cases} \max\{g_i(M), a_i\} & \text{if } M_i \subseteq M \\ g_i(M) & \text{otherwise.} \end{cases}$$

Let $g : \mathcal{G} \to \mathbb{R} \cup \{\infty\}$, $g(M) = \max\{g_i(M) | i \in \omega\}$. (We will see that there is always an i such that $g(M) = g_i(M)$.)

Let $f^* : \mathcal{G} \to \mathbb{R} \cup \{\infty\}$,

$$f^*(G) = \begin{cases} g(G) & \text{if } M_i \subseteq M \\ \max\{g(M) | M \subseteq G \text{ and } M \in \mathcal{M}\} & \text{otherwise.} \end{cases}$$

LEMMA 5.6. *Let* f, f^*, g, $g_1, \ldots, g_i, \ldots$ *as in definition 5.5,* $n \geq 2$, $i_0 = \binom{n}{2}^s$. *Then* $P(r,s,g_{i_0})$ *is true (therefore* $g = g_{i_0-1}$*).*

Proof: In Definition 5.5, for the chosen M_i, $g_{i+1} = a_i$. Also $a_k \leq a_i$ for $k > i$. Hence M_i won't be chosen again. Since all $M_1, M_2, \ldots, M_i, \ldots$ are from $\mathcal{M}$ and $|\mathcal{M}| \leq \binom{n}{2}^s$, $P(r,s,g_{i_0})$ is true for $i_0 \geq \binom{n}{2}^s$. ∎

DEFINITION 5.7. For $f \in \mathcal{F}_{\mathcal{G}}$, $E \in \mathcal{G}$ is called *f-minimal* if and only if for every $E' \subset E$, $f(E') < f(E)$. ∎

LEMMA 5.8. *Let $f \in \mathcal{F}_{\mathcal{G}}$, then the number of f-minimal graphs is at most $(s+1)(r-1)^s$.*

Proof: To prove the lemma, we generalize the definition of r-closed. A function $f : \mathcal{G} \to \mathcal{R}$ is r-closed with parameters s and $\mathcal{G}$, if the conditions in the previous definition for r-closed with $s, \mathcal{G}, \mathcal{M}$ replaced by $s', \mathcal{G}', \mathcal{M}' = \mathcal{M}_{s'} \cap \mathcal{G}'$ hold for arbitrary $0 \leq s' \leq s$ and arbitrary $\mathcal{G}' \subseteq \mathcal{G}$ which is closed under union and subsets.

If $E \in \mathcal{G} \backslash \mathcal{M}$, then $f(E) = \max\{f(M) | M \in \mathcal{M}, M \subseteq E\}$. Thus by the definition of f-minimal, E is not a f-minimal graph. We proceed the proof of the assertion by induction on $r \geq 2$ for arbitrary s and $\mathcal{G}$.

Base case $r = 2$: Let H be the set of f-minimal graphs. If $|H| > s+1$, then H contains graphs M_1 and M_2 in $\mathcal{M}$ such that $M_1 \not\subseteq M_2$ and $M_2 \not\subseteq M_1$. If not then $H = \{M_1, \ldots, M_l\}$ with $M_1 \subset M_2 \subset \ldots \subset M_l$ and $l > s+1$ imply graph M_l must have at least $s+1$ edges since each graph $M_i \in \mathcal{M}$ $(1 \leq i \leq l)$ contains no isolated vertex. But $M_l \in \mathcal{M}$ contains at most s edges. Therefore $M_1 \cap M_2 \subset M_1$ and $\langle M_1, M_2 \rangle \vdash M_1 \cap M_2$, hence $f(M_1 \cap M_2) \geq \min\{f(M_1), f(M_2)\}$ which contradicts the minimality of M_1 and M_2.

Induction step from $r-1$ to r: Assume the assertion is true for every $(r-1)$-closed function from $\mathcal{G}_s$ to $\mathbb{R}$. Let $f \in \mathcal{F}_{\mathcal{G}}$ be r-closed with parameters s and $\mathcal{G}$. Let $W \in \mathcal{M}'$ be f-minimal, $f(W) = \max\{f(W') | W' \in \mathcal{M}'\}$. For each $D \subseteq W$, let $\mathcal{G}_D = \{F \backslash D : F \in \mathcal{G} \text{ and } F \cap W = D\}$ And $\mathcal{M}_D = \mathcal{M} \cap \mathcal{G}_D = \{F \backslash D | F \in \mathcal{M} \text{ and } F \cap W = D\}$. Define $f_D : \mathcal{G}_D \to \mathbb{R}$ by $f_D(G) = f(G \cup D)$. Note that for $G \in \mathcal{M}_D$, $G \cup D \in \mathcal{M}$. We claim that f_D is $(r-1)$-closed with parameters $s - |D|$ and $\mathcal{G}_D$. Assume $\langle M_1, \ldots, M_{r-1}, M \rangle \in \mathcal{M}'_D$ with $\bigcup_{1 \leq i < j \leq r-1} M_i \cap M_j \subset M$. Then

$$\begin{aligned} f_D(M) &= f(M \cup D) \geq \min\{f(M_1 \cup D), \ldots, f(M_{r-1} \cup D), f(W)\} \\ &= \min\{f(M_1 \cup D), \ldots, f(M_{r-1} \cup D)\} \\ &= \min\{f_D(M_1), \ldots, f_D(M_{r-1})\}. \end{aligned}$$

If $E \in \mathcal{G}_D \backslash \mathcal{M}_D$,

$$\begin{aligned} f_D(E) &= f(E \cup D) = \max\{f(M) | M \in \mathcal{M}, M \subseteq E \cup D\} \\ &= \max\{f(M) | M \in \mathcal{M}, D \subseteq M \subseteq E \cup D\} \\ &= \max\{f((M \backslash D) \cup D) M \in \mathcal{M}, M \subseteq E \cup D, M \cap W = D\} \\ &= \max\{f((M \backslash D) \cup D) | M \in \mathcal{M}, (M \backslash D) \subseteq E, M \cap W = D\} \\ &= \max\{f_D(M) | M \in \mathcal{M}_D, M \subseteq E\}. \end{aligned}$$

Let $G \in \mathcal{M}$ be f-minimal. Let $G' = G \backslash W$. If $G'' \subset G'$, then $G'' \cup D \subset G' \cup D = G$, $f_D(G'') = f(G'' \cup D) < f(G' \cup D) = f_D(G')$. Thus G' is f_D-minimal. Therefore $\{G | G \text{ is } f\text{-minimal}\} \subseteq \bigcup_{D \subseteq W} \{G \cap W = D, G \backslash W \text{ is } f_D\text{-minimal with parameters } s - |D| \text{ and } \mathcal{G}_D\}$, and we have

$$\begin{aligned} |\{G | G \text{ is } f\text{-minimal}\}| &\leq \sum_{D \subseteq W} |\{G | G \text{ is } f_D\text{-minimal}\}| \\ &\leq \sum_{i=0}^{s} \binom{s}{i} (s-i+1)(r-2)^{s-i} \\ &\leq (s+1) \sum_{i=0}^{s} \binom{s}{i} (r-2)^{s-i} \\ &= (s+1)(r-1)^s. \end{aligned}$$

∎

We shall approximate any circuit that outputs 1 on inputs from Mat_2 and outputs 0 on inputs from Par_2 by approximating each function F computed at each gate by such a function $\hat{f}$ where $f \in \mathcal{F}_{\mathcal{M}}$. Each input vector $\langle x_{\{i,j\}} \rangle_{i,j \in V}$ is 1-1 correspond to a graph with vertex set V in such a way that $x_{\{i,j\}} = 1$ if and only if $\{i,j\}$ is an edge in the graph.

DEFINITION 5.9. For each gate g in a monotone circuit over the reals with inputs $\langle x_{\{i,j\}} \rangle_{i,j \in V}$, we define a function $f_g \in \mathcal{F}_{\mathcal{G}}$. If g is an input gate $x_{\{i,j\}}$, then $f_g(G) = 1$ if and only if $\{i,j\} \in G$. If g is an unary gate $\diamond(g')$, then f_g is $\diamond \circ f_{g'}$ obtained by applying the unary operation $\diamond$ on $f_{g'}$. If g is a binary gate $\diamond(g', g'')$, then f_g is $(\diamond \circ \langle f_{g'}, f_{g''} \rangle)^*$, the closure of the function obtained by applying the binary operation $\diamond$ on $f_{g'}$ and $f_{g''}$.

The approximator function on gate g is $\hat{f}_g$. ∎

We shall estimate the number of errors introduced in each step of approximation. In our analysis we only monitor the behavior of the approximator functions on two sets of graphs Mat_2 and Par_2. We say that the approximator function has introduced a *false increase* if its value is greater than the value of the original function when a graph from Par_2 is supplied as inputs to both functions. Similarly We say that the approximator function has introduced a *false decrease* if its value is less than the value of the original function when a graph from Mat_2 is supplied as inputs to both functions. Clearly the approximation of a unary function doesn't introduce any new false increase or new false decrease. We only need to analyze the binary operations. The approximation of binary operation is done by taking closure of the operation on the two input approximators. In Definition 5.5, the values of the function on certain sets of partial matchings increase, it may introduce false increase. On the other hand, the value of the new approximator function on some input graph would be less than the value obtained by directly applying the binary operation to the values of the two approximator functions for the input functions of the binary operation. This happens when the input graph contains a large minimal partial matching for each input approximator function and the value of each minimal partial matching on its own function is greater than that on the other function. Thus this may introduce false decrease.

Now we fix $s = \lfloor \frac{1}{8} \log n \rfloor$ and $r = 2\lfloor n^{\frac{1}{4}} (\log n)^8 \rfloor$. We define a set of random graphs $\mathsf{Par}_2(h)$. Let $V = \{1, \ldots, n\}$ be the vertex set. Select a random labeling $h : V \to \{0,1\}$ of the vertices. Each labeling is chosen with probability 2^{-n}. The vertices i and j are connected by an edge in $\mathsf{Par}_2(h)$ if and only if $h(i) \neq h(j)$.

The following lemma is essentially the same as Lemma 4 in [**Raz85a**]. We reprove it for general graphs.

LEMMA 5.10. *Let $M = \{M_1, M_2, \ldots, M_r\}$, $M_i \in \mathcal{M}$, $1 \le i \le r$ such that $M_i \cap M_j = \emptyset$ whenever $i \neq j$. Then there exists a subset $\{T_1, T_2, \ldots, T_p\}$ of M such that $p \ge \frac{\sqrt{\frac{r}{2}}}{s}$ and for which the graph with edges $\bigcup_{i=1}^{p} T_i$ contains no cycles, i.e. a forest.*

Proof: Let $\{T_1, T_2, \ldots, T_p\}$ be a maximal size subset of M for which $\bigcup_{i=1}^{p} T_i$ is a forest. Suppose $p < \frac{\sqrt{\frac{r}{2}}}{s}$. Put $E = \bigcup_{i=1}^{p} T_i$. Let X be the vertices occur in at least one edge of E. Since $|T_i| \le s$, $|E| < \sqrt{\frac{r}{2}}$. Thus $|X| < \sqrt{2r}$ and $|X \times X| = \binom{|X|}{2} < r = |M|$. It follows from edge disjointness of matchings in M that we can find some matching $M_j \in M$ such that $M_j \cap (X \times X) = \emptyset$. Hence $M_j \cap E = \emptyset$. But E is a forest and M_j is a matching, so $E \cup M_j$ is also forest, which is a contradiction. ∎

The lemma below is Lemma 5 in [**Raz85a**]. For completeness we include the proof here.

LEMMA 5.11. *Let T be a forest over $V = \{1, \ldots, n\}$ which contains exactly p edges $\{\{i_k, j_k\} : 1 \le k \le p, 1 \le i_k, j_k \le n\}$. Then the events $\{\{i_k, j_k\}$ is an edge of the random graph $\mathsf{Par}_2(h)\}$ occur independently with probability $\frac{1}{2}$.*

Proof: It is sufficient to show that for any subset K of $\{1, 2, \ldots, p\}$ the probability of the event $E = \{\forall k \in K, \{i_k, j_k\} \in \mathsf{Par}_2(h); \forall k \notin K, \{i_k, j_k\} \notin \mathsf{Par}_2(h)\}$ is exactly 2^{-p}. Let $q \ge 1$ be the number of connected components then T contains exactly $p + q$ vertices. For each tree there are exactly 2 ways of labeling as long as the appearance of edge in $\mathsf{Par}_2(h)$ determined by K. Since each component may be labeled independently of others, it follows that there are 2^q labeling of the forest T. This leaves $n - p - q$ unlabeled vertices that are not the endpoint of any edge in T. Thus the probability of the event E is $2^{n-p-q} \times 2^q / 2^n = 2^{-p}$. ∎

LEMMA 5.12. *For a random graph $\mathsf{Par}_2(h)$, $\mathrm{Prob}[\mathsf{Par}_2(h) \notin \mathsf{Par}_2] \le \sqrt{\frac{2}{\pi n}}$.*

Proof: A random graph $\mathsf{Par}_2(h) \notin \mathsf{Par}_2$ if and only if the number of vertices of $V = \{1, \ldots, n\}$ labeled with i ($i = 0$, or 1) by a random labeling h equals $\frac{n}{2}$. Thus

$$\mathrm{Prob}[\mathsf{Par}_2(h) \notin \mathsf{Par}_2] = \mathrm{Prob}[\sum_{j=1}^{n} h(j) = \frac{n}{2}] = \binom{n}{\frac{n}{2}} 2^{-n}$$

By Stirling's formula $n! = n^n e^{-n} \sqrt{2\pi n} e^{\alpha_n}$ where $\frac{1}{12n+1} < \alpha_n < \frac{1}{12n}$, we have

$$\binom{n}{\frac{n}{2}} 2^{-n} = \frac{n!}{(\frac{n}{2}!)^2} 2^{-n} \le \frac{(n/e)^n \sqrt{2\pi n} e^{\frac{1}{12n}}}{((n/2e)^{\frac{n}{2}} \sqrt{\pi n} e^{\frac{1}{6n+1}})^2} 2^{-n}$$

$$= 2^n \sqrt{\frac{2}{\pi n}} e^{\frac{1}{12n} - \frac{1}{6n+1}} 2^{-n} \le \sqrt{\frac{2}{\pi n}}$$

as required. ∎

LEMMA 5.13. *For a random graph $\mathsf{Par}_2(h)$, and functions $f_1, f_2 \in \mathcal{F}_{\mathcal{G}}$,*

$$\mathrm{Prob}[\widehat{(f_1 \circ f_2)^*}(\mathsf{Par}_2(h)) > \widehat{f_1 \circ f_2}(\mathsf{Par}_2(h))] \le \binom{n}{2}^s (1 - 2^{-s})^{\frac{\sqrt{\frac{r}{2}}}{s}}.$$

Proof. Since the value of $\widehat{f_1 \circ f_2}(\mathsf{Par}_2(h))$ increases, in Definition 5.5, it must be the case that there exists some r-tuple $\langle G_1, \ldots, G_r \rangle \in F_i$ and $M_i \in B_i$ for which $\langle G_1, \ldots, G_r \rangle \vdash M_i$ such that $M_i \subseteq \mathsf{Par}_2(h)$ and $\widehat{f_1 \circ f_2}(\mathsf{Par}_2(h)) < \min\{f_1 \circ f_2(G_1), \ldots, f_1 \circ f_2(G_r)\}$. Since $f_1 \circ f_2$ and $\widehat{f_1 \circ f_2}$ are monotone, $G_j \not\subseteq \mathsf{Par}_2(h)$ for $j = 1, \ldots, r$.

Consider the set of matchings $\{U_j | U_j = G_j - M_i\}$. By the definition of $\vdash$, we have $U_k \cap U_l = \emptyset$ whenever $k \neq l$. We claim that any U_j is non-empty. To prove the claim we suppose the contrary, then $G_j \subseteq M_i$. Since $f_1 \circ f_2$ is monotone, $(f_1 \circ f_2)(G_j) \le (f_1 \circ f_2)(M_i)$. By the definition of the extension function $\widehat{f_1 \circ f_2}$ and the fact that $M_i \subseteq \mathsf{Par}_2(h)$, we have

$$\widehat{f_1 \circ f_2}(\mathsf{Par}_2(h)) \ge (f_1 \circ f_2)(M_i) \ge (f_1 \circ f_2)(G_j)$$

which contradicts $\widehat{f_1 \circ f_2}(\mathsf{Par}_2(h)) < \min\ (f_1 \circ f_2)(G_j)$. Thus conditions of Lemma 5.10 holds for the set $\{U_1, \ldots, U_r\}$ and we can find a subset $T = \{T_1, \ldots, T_p\}$ of this set such that $p \ge \sqrt{\frac{r}{2}}/s$ and for which $\bigcup_{i=1}^{p} T_i$ is a forest.

Then by Lemma 5.6 and Lemma 5.11, we have

$$\begin{aligned}
&\mathrm{Prob}[\widehat{(f_1 \circ f_2)^*}(\mathsf{Par}_2(h)) > \widehat{f_1 \circ f_2}(\mathsf{Par}_2(h))] \\
&\le \binom{n}{2}^s \mathrm{Prob}[M \subseteq \mathsf{Par}_2(h) \ \&\ \forall i\ M_i \not\subseteq \mathsf{Par}_2(h)] \\
&= \binom{n}{2}^s \mathrm{Prob}[M \subseteq \mathsf{Par}_2(h) \ \&\ \forall i\ U_i \not\subseteq \mathsf{Par}_2(h)] \\
&\le \binom{n}{2}^s \mathrm{Prob}[\forall i\ U_i \not\subseteq \mathsf{Par}_2(h)] \\
&\le \binom{n}{2}^s \mathrm{Prob}[\forall T_j \in T,\ T_j \not\subseteq \mathsf{Par}_2(h)] \\
&= \binom{n}{2}^s \prod_{j=1}^{p} \mathrm{Prob}[T_j \not\subseteq \mathsf{Par}_2(h)] \\
&\le \binom{n}{2}^s (1 - 2^{-s})^{\frac{\sqrt{\frac{r}{2}}}{s}}
\end{aligned}$$

as required. ∎

LEMMA 5.14. *For every binary operation $\circ$ and for all $f_1, f_2 \in \mathcal{F}_{\mathcal{M}}$, the number of graphs $A \in \mathsf{Mat}_2$ such that $\widehat{(f_1 \circ f_2)^*}(A) < \widehat{f_1}(A) \circ \widehat{f_2}(A)$ is at most*

$$\frac{n!}{\frac{n}{2}! 2^{\frac{n}{2}}} 2(s+1) \left(\frac{r-1}{n-2s+1}\right)^{\lceil \frac{s+1}{2} \rceil} \Big/ \left(1 - \frac{r-1}{n-2s+1}\right).$$

Proof: For all $M \in \mathcal{M}$, $(f_1 \circ f_2)^*(M) \ge (f_1 \circ f_2)(M)$. For all $E \in \mathcal{G} \backslash \mathcal{M}$ we have

$$\begin{aligned}
(f_1 \circ f_2)^*(E) &= \max_{M \subseteq E, M \in \mathcal{M}} (f_1 \circ f_2)^*(M) \\
&\ge \max_{M \subseteq E, M \in \mathcal{M}} (f_1 \circ f_2)(M) \\
&= (\max_{M \subseteq E, M \in \mathcal{M}} f_1(M)) \circ (\max_{M \subseteq E, M \in \mathcal{M}} f_2(M)) \\
&= f_1(E) \circ f_2(E) = (f_1 \circ f_2)(E).
\end{aligned}$$

The second last equality is true since f_1, f_2 and $f_1 \circ f_2$ are monotone. Thus $\widehat{(f_1 \circ f_2)^*}(A) \ge \widehat{f_1 \circ f_2}(A)$. By the definition of $\widehat{f_i}$ and the definition of f_i-minimal graph, there exist M_1, $M_2 \in \mathcal{M}$ such that M_i is f_i-minimal and $\widehat{f_i}(A) = f_i(M_i)$ ..

such that $M_i \subseteq A$, $i = 1, 2$. Again by definition of $\widehat{f_1 \circ f_2}$, $\widehat{f_1 \circ f_2}(A) \geq (f_1 \circ f_2)(E) = f_1(E) \circ f_2(E)$, for all $E \subseteq A$ and $E \in \mathcal{G}$. Hence

$$f_1(E) \circ f_2(E) < f_1(M_1) \circ f_2(M_2) \tag{7}$$

for all $E \subseteq A$ and $E \in \mathcal{G}$.

Case 1: The graph $M_1 \cup M_2$ has at most s edges. Then the number of vertices in the graph $M_1 \cup M_2$ is at most $2s$. Thus $M_1 \cup M_2 \in \mathcal{G}$. Hence we take $E = M_1 \cup M_2$ to obtain $f_1(M_1 \cup M_2) \circ f_2(M_1 \cup M_2) \geq f_1(M_1) \circ f_2(M_2)$ which contradicts (7) since f_1 and f_2 are monotone.

Case 2: The graph $M_1 \cup M_2$ has at least $s+1$ edges. Then one of the M_1 or M_2 includes at least $\lceil \frac{s}{2} \rceil$ edges. Thus a f_1-minimal graph or f_2-minimal graph with at least $\lceil \frac{s}{2} \rceil$ edges is included in A. Since each f_i-minimal graph with $\lceil \frac{s}{2} \rceil \leq k \leq s$ edges can be responsible only for $(n-2k)!/((\frac{n}{2}-k)!2^{\frac{n}{2}-k})$ graphs from Mat_2. Hence the number of graphs $A \in \mathsf{Mat}_2$ such that $(\widehat{f_1 \circ f_2})^*(A) < \widehat{f_1}(A) \circ \widehat{f_2}(A)$ is bounded by

$$\sum_{\lceil \frac{s+1}{2} \rceil \leq k \leq s+1} 2(k+1)(r-1)^k \frac{(n-2k)!}{(\frac{n}{2}-k)!2^{\frac{n}{2}-k}} \leq \frac{n!}{\frac{n}{2}!2^{\frac{n}{2}}} 2(s+1) \sum_{\lceil \frac{s+1}{2} \rceil \leq k \leq s+1} \left(\frac{r-1}{n-2s+1}\right)^k$$
$$\leq \frac{n!}{\frac{n}{2}!2^{\frac{n}{2}}} 2(s+1) \left(\frac{r-1}{n-2s+1}\right)^{\lceil \frac{s}{2} \rceil} \Big/ \left(1 - \frac{r-1}{n-2s+1}\right).$$

∎

Lemma 5.14 gives upper bounds on the number of false decreases and Lemma 5.13 gives the upper bound on the probability of a random graph being a false increase introduced by each approximator function in each step. We next show that any approximator function $\widehat{f}$ where $f \in \mathcal{F}_{\mathcal{M}}$ for a monotone function that outputs 1 on all graphs from Mat_2 and outputs 0 on all graphs from Par_2 must introduce either a large number of false increases or a large number of false decreases.

LEMMA 5.15. *Let $f \in \mathcal{F}_{\mathcal{M}}$ and suppose $\widehat{f}(A) \geq 1$ for some $A \in \mathsf{Mat}_2$. Then for a random graph $\mathsf{Par}_2(h)$, $\mathsf{Prob}[\widehat{f}(\mathsf{Par}_2(h)) \geq 1] \geq 2^{-s}$.*

Proof: By the definition of $\widehat{f}$ there exists a f-minimal partial matching $M \in \mathcal{M}$ such that $\widehat{f}(A) = f(M)$ and $M \subseteq A$. Thus $f(M) \geq 1$. By the monotonicity of f, we have

$$\mathsf{Prob}[\widehat{f}(\mathsf{Par}_2(h)) \geq 1] \geq \mathsf{Prob}[\mathsf{Par}_2(h) \supseteq M]$$
$$= 2^{-|M|} \geq 2^{-s},$$

since M is obviously a forest, the above equality follows from Lemma 5.11. ∎

Proof of Theorem 5.1. 8 Let F be the monotone function computed by any monotone circuit over the reals that outputs 1 on all graphs from Mat_2 and outputs 0 on all graphs from Par_2. Let $\widehat{f}$ be the approximator function for F, where $f \in \mathcal{F}_{\mathcal{M}}$.

If $\widehat{f}(A) < 1$ for all graphs $A \in \mathsf{Mat}_2$, then $\widehat{f}$ introduces false decrease on all $n!/(\frac{n}{2}!2^{\frac{n}{2}})$ graphs in Mat_2. By Lemma 5.14, the number of the gates in the circuit

must be at least

$$\frac{1}{2(s+1)} \left(1 - \frac{r-1}{n-2s+1}\right) \left(\frac{n-2s+1}{r-1}\right)^{\lceil \frac{s}{2} \rceil - 1}$$
$$\geq \frac{1}{2(s+1)} \left(1 - \frac{r-1}{n-2s+1}\right) \left(\frac{n-2\lfloor \frac{1}{8} \log n \rfloor + 1}{2\lfloor n^{\frac{1}{4}} (\log n)^8 \rfloor - 1}\right)^{\frac{1}{2}\lfloor \frac{1}{8} \log n \rfloor - \frac{1}{2}}$$
$$\geq n^{\Omega(\log n)}$$

for sufficiently large n.

Since any graph from Par_2 contains no perfect matchings, if there exists some $A \in \mathsf{Mat}_2$ such that $\widehat{f}(A) \geq 1$ then by Lemma 5.12, Lemma 5.13 and Lemma 5.15, the number of gates in the circuit must be at least

$$\frac{\mathsf{Prob}[\widehat{f}(\mathsf{Par}_2(h)) \geq 1] - \mathsf{Prob}[\mathsf{Par}_2(h) \notin \mathsf{Par}_2]}{\mathsf{Prob}[\widehat{f^*}(\mathsf{Par}_2(h)) \geq \widehat{f}(\mathsf{Par}_2(h))]}$$
$$\geq \frac{2^{-s} - \frac{1}{\sqrt{n}}}{\binom{n}{2}^s (1 - 2^{-s})^{\frac{\sqrt{r}}{2}}}$$
$$\geq \frac{1}{2} n^{-\frac{1}{8}} \exp\left(\frac{2^{-s}\sqrt{r/2}}{s}\right) n^{-2s}$$
$$\geq \frac{1}{2} n^{-\frac{1}{8} - \frac{1}{4}\log n} \exp\left(\frac{8 n^{-\frac{1}{8}} n^{\frac{1}{8}} (\log n)^4}{\log n}\right)$$
$$\geq n^{(\log n)^2}$$

for sufficiently large n. Thus the theorem follows. ∎

Acknowledgment

I am grateful to Steve Cook for bringing this problem to my attention and many invaluable suggestions. I am indebted to Arnold Rosenbloom for the conversation which leads to the proof of Corollary 5.2. I would like to thank Charlie Rackoff, Paul Beame, Toni Pitassi and Alasdair Urquhart for their insightful comments.

References

[Ajt88] Miklós Ajtai, *The complexity of the pigeonhole principle*, Proceedings of the 29th Annual IEEE Symposium on the Foundations of Computer Science, 1988, pp. 346–355.

[Ajt90] Ajtai, *Parity and the pigeonhole principle*, Feasible Mathematics: A Mathematical Sciences Institute Workshop, Birkhauser, 1990.

[Ajt94] Ajtai, *The independence of the modulo p counting principles*, ACM Symposium on Theory of Computing (STOC), 1994.

[BIK+92] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, Pavel Pudlák, and Alan Woods, *Exponential lower bounds for the pigeonhole principle*, Proceedings of the 24th Annual ACM Symposium on the theory of computing, 1992, pp. 200–220.

[BIK+94] Beame, Impagliazzo, Krajicek, Pitassi, and Pudlak, *Lower bounds on hilbert's nullstellensatz and propostional proofs*, IEEE Symposium on Foundations of Computer Science (FOCS), 1994.

[BP94] Paul Beame and Toniann Pitassi, *An exponential separation between the matching principle and the pigeonhole principle*, Annals of Pure and Applied Logic, 1994.

[BPR95] M. Bonet, T. Pitassi, and R. Raz, *Lower bounds for cutting planes proofs with small coefficients*, Proceedings of the 27th Annual ACM Symposium on the theory of computing, 1995.

[CCT87] W. Cook, C.R. Coullard, and Gy. Turan, *On the complexity of cutting plane proofs*, Discrete Applied Mathematics 18 (1987), 25–38.

[Chv73] Vašek Chvátal, *Edmonds polytopes and a hierarchy of combinatorial problems*, Discrete Math. 4 (1973), 305–337.

[Coo71] Stephen A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the 3rd Annual Symposium on the Theory of Computing (Shaker Heights, Ohio, May 3-7). ACM, New York, 1971, pp. 151–158.

[CR79] S. A. Cook and R. A. Reckhow, *The relative efficiency of propositional proof systems*, Journal of Symbolic Logic 44 (1979), no. 1, 36–50.

[Cra57a] W. Craig, *Linear reasoning: A new form of the herbrand-gentzen theorem*, Journal of Symbolic Logic 22 (1957), no. 3, 250–278.

[Cra57b] W. Craig, *Three uses of the herbrand-gentzen theorem in relating model theory and proof theory*, Journal of Symbolic Logic 22 (1957), no. 3, 269–285.

[Fu95] Xudong Fu, *Modular 2 counting formulas are hard for cutting planes proofs*, Submitted to the Twenty-Eighth ACM Symposium on Theory of Computing (STOC), 1995.

[Gom63] R.E. Gomory, *An algorithm for integer solutions of linear programs*, in: R.L. Graves and P. Wolfe, eds., Recent Advances in Mathematical Programming, *McGraw-Hill, New York* (1963), 269–302.

[Hak85] A. Haken, *The intractability of resolution*, Theoretical Computer Science 39 (1985), 279–308.

[HC95] A. Haken and S. Cook, *An exponential lower bound for the size of monotone real circuits*, preprint, 1995.

[Kra94a] Jan Krajíček, *Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic*, preprint, 1994.

[Kra94b] Jan Krajíček, *Lower bounds to the size of constant-depth propositional proofs*, Journal of Symbolic Logic 59 (1994), 73–86.

[Pud95] Pavel Pudlák, *Lower bounds for resolution and cutting planes proofs and monotone computations*, preprint, 1995.

[Raz85a] Alexander A. Razborov, *A lower bounds for the monotone complexity of logical permanent, mat. zametki, vol 37, pp. 887-901, 1985 (in russian). english translation in*, Mathem. Notes of the Acad. of Sci. of the USSR 37 (1985), 485–493.

[Raz85b] Alexander A. Razborov, *Lower bounds for the monotone complexity of some boolean functions, dokl. ak. nauk. sssr, vol 281, pp. 798 -801, 1985 (in russian). english translation in*, Sov. Math. Dokl. 3 (1985), 354–357.

[Sch80] A. Schrijver, *On cutting planes*, in: Combinatorics 79, part II, Annals of Discrete Math. 9, eds. M. Deza and I.G. Rosenberg, *North-Holland* (1980), 291–296.

[Set96] Anton Setzer, *Handout on xudong fu:*modular 2 counting formulas are hard for cutting planes proofs, manuscript, 1996.

[Urq95] Alasdair Urquhart, *Personal communication*, 1995.

[Lee90] J. van Leeuwen, *Handbook of theoretical computer science*, The MIT Press, 1990.

School of Computer Science, McGill University, 3480 University Street, McConnell Engineering Building Room 318, Montreal, QC, Canada,
E-mail address: xudongopus.cs.mcgill.ca

Département d'informatique et de recherche opérationnelle, Université de Montréal, CP 6128, Succ. Centre-Ville, Montréal, Québec, Canada H3C 3J7,
E-mail address: fuiro.umontreal.ca

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

Equational Calculi and Constant Depth Propositional Proofs

Jan Johannsen

Abstract. We define equational calculi for proving equations between functions in the complexity classes $ACC(2)$ and TC^0, and we show that proofs in these calculi can be simulated by polynomial size, constant depth proofs in Frege systems with counting modulo 2 and threshold connectives respectively.

Introduction

To motivate our work, we give a brief overview of the theory of propositional proof systems, for a more detailed exposition see e.g. the recent survey [18]. A propositional proof system is a polynomial time computable function whose range is the set of propositional tautologies. The usual proof systems fall under this definition if we associate with them the function mapping a valid proof to the tautology proved by it, and every other string to some fixed tautology.

A proof system is *polynomially bounded* if for every tautology A, there is a proof in it of length polynomial in the length of A. The existence of a polynomially bounded proof system is equivalent to $NP = co\text{-}NP$, hence the quest for lower bounds on the length of propositional proofs can be considered an approach to this problem from computational complexity theory.

A proof system P_1 *polynomially simulates* P_2, if for each proof p in P_2, there is a proof in P_1 of the same tautology whose length is polynomial in the length of p. Two proof systems are *polynomially equivalent* if they polynomially simulate each other.

A *Frege system* is a usual proof system for tautologies in a language with finitely many connectives, given by finitely many axiom schemes and inference rules, which are implicationally complete in the sense that if the formulas $B_1, \ldots, B_m$ semantically entail A, then there must be a proof of A from the hypotheses $B_1, \ldots, B_m$. All Frege systems are polynomially equivalent [14]. An *extended Frege system* is a Frege system extended by the substitution rule. An important open question is whether Frege systems are polynomially bounded, or whether they can polynomially simulate extended Frege systems.

In a *constant depth Frege system*, the depth of formulas appearing in proofs is required to be bounded by a constant, where the depth of formulas is measured as

1991 *Mathematics Subject Classification.* Primary 03F20; Secondary 03F30, 68Q15.
This paper is in final form and no version of it will be submitted elsewhere for publication.

if the binary connectives were of unbounded arity. Constant depth Frege systems and some extensions of these by additional, non-schematic axioms (like pigeonhole and counting principles) are known not to be polynomially bounded [1, 4, 2, 5, 3].

A recurring theme in the theory of propositional proof systems is the correspondence of certain proof systems to certain complexity classes. So e.g. extended Frege systems correspond to P, Frege systems to NC^1 and constant depth Frege systems to AC^0.

The first of these correspondences was made precise by S. Cook in his classic paper [13], where he defined an equational calculus PV for proving equations between polynomial time computable functions, based on Cobham's characterization of this class as a function algebra [12]. He then showed that proofs in PV can be simulated by polynomial size families of extended Frege proofs.

In the same vein, P. Clote [10] defined calculi ALV and AV for equations between functions in NC^1 and AC^0 resp., and showed that proofs in these calculi can be simulated by polynomial size Frege proofs and constant depth Frege proofs respectively.

Recently, extensions of Frege systems by modular counting [3] and threshold connectives [15, 7] were introduced, where constant depth proofs in these intuitively correspond to the circuit complexity classes $ACC(m)$ and TC^0. We support this intuition by defining equational calculi $A2V$ for functions in $ACC(2)$ and TV for functions in TC^0 and showing that proofs in these calculi can be simulated by polynomial size, constant depth proofs in the corresponding proof systems.

Two propositional proof systems

Let PK denote the propositional part of the classical sequent calculus LK, with the connectives $\wedge$, $\vee$ and $\neg$. It is well-known that PK is polynomially equivalent to any Frege system [14]. Moreover the mutual simulations do not increase the depth of formulas occurring in a proof by more than a constant, provided that the Frege system has the same underlying set of connectives.

We extend PK by the binary connective $\oplus$ (exclusive disjunction) and the following inference rules for its introduction:

$$\oplus\text{-left1}: \frac{\Gamma \Longrightarrow A, \Delta \quad \Gamma \Longrightarrow B, \Delta}{A \oplus B, \Gamma \Longrightarrow \Delta} \qquad \oplus\text{-left2}: \frac{A, \Gamma \Longrightarrow \Delta \quad B, \Gamma \Longrightarrow \Delta}{A \oplus B, \Gamma \Longrightarrow \Delta}$$

$$\oplus\text{-right1}: \frac{\Gamma \Longrightarrow A, \Delta \quad B, \Gamma \Longrightarrow \Delta}{\Gamma \Longrightarrow A \oplus B, \Delta} \qquad \oplus\text{-right2}: \frac{A, \Gamma \Longrightarrow \Delta \quad \Gamma \Longrightarrow B, \Delta}{\Gamma \Longrightarrow A \oplus B, \Delta}$$

We call this extension $PK\oplus$. Define the formulas $\bigoplus^n(A_1, \dots, A_n)$ for $n \geq 2$ inductively by $\bigoplus^2(A, B) := A \oplus B$ and

$$\bigoplus^{n+1}(A_1, \dots, A_{n+1}) := A_1 \oplus \bigoplus^n(A_2, \dots, A_{n+1}) .$$

Let $B(p_1, \dots, p_n)$ be a formula built up from the variables $p_1, \dots, p_n$ using only one kind of binary connective, and let $A_1, \dots, A_n$ be formulas with an outermost connective of a different kind. If d is the maximum of the depths of the formulas A_i, then $B(A_1, \dots, A_n)$ is a formula of depth $d+1$. With this notion of depth, $PK\oplus$ is polynomially equivalent to a Frege system with Mod_2 connectives $F(Mod_2)$ as introduced e.g. in [3] and to the Frege system with biconditional considered in [17]. In both cases, the mutual simulations do not increase the formula-depth in a proof by more than a constant.

Propositional threshold logic, as introduced in [7], has the unary connective $\neg$ and for each $n \geq 1$ and $1 \leq k \leq n$ the n-ary threshold connective T_k^n, where $T_k^n(A_1, \dots, A_n)$ is intended to be true if at least k of the A_i are true. The depth of a threshold logic formula is simply its syntactic depth, and its size is the sum of the sizes of the variables and connectives in it, where the variables and $\neg$ are of size 1 and T_k^n is of size $n+k+1$. Note that n-ary conjunction and disjunction are the special cases T_n^n and T_1^n of threshold connectives.

The sequent calculus PTK for propositional threshold logic has the initial sequents $A \Longrightarrow A$, the usual structural rules, cut rule and rules for negation plus the following versions of the rules for conjunction

$$\wedge\text{-left}: \frac{A_1, \dots, A_n, \Gamma \Longrightarrow \Delta}{T_n^n(A_1, \dots, A_n), \Gamma \Longrightarrow \Delta}$$

$$\wedge\text{-right}: \frac{\Gamma \Longrightarrow A_1, \Delta \quad \cdots \quad \Gamma \Longrightarrow A_n, \Delta}{\Gamma \Longrightarrow T_n^n(A_1, \dots, A_n), \Delta}$$

and the dual rules for disjunction. Additionally, for $n \geq 3$ there are the following rules for T_k^n with $1 < k < n$:

$$T_k^n\text{-left}: \frac{T_k^{n-1}(A_2, \dots A_n), \Gamma \Longrightarrow \Delta \qquad A_1, T_{k-1}^{n-1}(A_2, \dots, A_n), \Gamma \Longrightarrow \Delta}{T_k^n(A_1, \dots, A_n), \Gamma \Longrightarrow \Delta}$$

$$T_k^n\text{-right}: \frac{\Gamma \Longrightarrow A_1, T_k^{n-1}(A_2, \dots A_n), \Delta \qquad \Gamma \Longrightarrow T_{k-1}^{n-1}(A_2, \dots, A_n), \Delta}{\Gamma \Longrightarrow T_k^n(A_1, \dots, A_n), \Delta}$$

The correctness and completeness of PTK was proved in [7]. Furthermore it was proved in [8] that PTK is polynomially equivalent to a Frege system with threshold connectives FC introduced in [15], and that the mutual simulations increase the formula-depth in a proof at most by a constant.

The sequent calculus PTK^* is defined exactly like PTK, but where the rules T_k^n-right and T_k^n-left are replaced by the rules

$$T_k^n\text{-right1}: \frac{\Gamma \Longrightarrow A_1, \Delta \qquad \Gamma \Longrightarrow T_{k-1}^{n-1}(A_2, \dots, A_n), \Delta}{\Gamma \Longrightarrow T_k^n(A_1, \dots, A_n), \Delta}$$

$$T_k^n\text{-right2}: \frac{\Gamma \Longrightarrow T_k^{n-1}(A_2, \dots A_n), \Delta}{\Gamma \Longrightarrow T_k^n(A_1, \dots, A_n), \Delta}$$

$$T_k^n\text{-left1}: \frac{A_1, \Gamma \Longrightarrow \Delta \qquad T_k^{n-1}(A_2, \dots, A_n), \Gamma \Longrightarrow \Delta}{T_k^n(A_1, \dots, A_n), \Gamma \Longrightarrow \Delta}$$

$$T_k^n\text{-left2}: \frac{T_{k-1}^{n-1}(A_2, \dots A_n), \Gamma \Longrightarrow \Delta}{T_k^n(A_1, \dots, A_n), \Gamma \Longrightarrow \Delta} .$$

It is easily shown that PTK and PTK^* are polynomially equivalent, and that the mutual simulations do not increase the formula-depth.

Function algebras and equational calculi

Let $BASE$ denote the set of functions consisting of the constant 0, s_0, s_1, $mod2$, len, $trunc$ and $\#$, where $s_0(x) = 2x$, $s_1(x) = 2x+1$, $mod2(x) := x \mod 2$, $len(x) := |x| = \lceil \log_2(x+1) \rceil$, $trunc(x, y) := \lfloor \frac{x}{2^{|y|}} \rfloor$ and $x\#y := 2^{|x| \cdot |y|}$, together with the projections π_k^n for $1 \leq k \leq n \in \mathbb{N}$, where $\pi_k^n(x_1, \dots, x_n) := x_k$.

Let g be an n-ary function and h_0, h_1 be $n+1$-ary functions with $h_i(\vec{x}, y) \leq 1$ for $i = 0, 1$. Then the $n+1$-ary function f is defined by concatenation recursion

on notation (CRN) from g and h_0, h_1 if f is the unique function satisfying

$$\begin{aligned} f(\bar{x}, 0) &= g(\bar{x}) \\ f(\bar{x}, 2y) &= 2f(\bar{x}, y) + h_0(\bar{x}, y) \qquad \text{for } y > 0 \\ f(\bar{x}, 2y+1) &= 2f(\bar{x}, y) + h_1(\bar{x}, y) \end{aligned}$$

The following characterization of the functions in AC^0 was given in [9]:

PROPOSITION 1. *AC^0 is the smallest class of functions containing the $BASE$ functions and closed under composition and CRN.*

Let $count(x)$ be the number of bits equal to 1 in the binary representation of x, and let $parity(x) := count(x) \mod 2$. The following characterizations of the functions in $ACC(2)$ and TC^0 can be extracted from the proofs of Thm. 2.1 and 2.2 in [11]:

PROPOSITION 2. *$ACC(2)$ is the smallest class of functions that contains the $BASE$ functions and parity and is closed under composition and CRN, and TC^0 is the smallest class of functions containing the $BASE$ functions and count and closed under composition and CRN.*

Based on the characterization given in Prop. 1, the equational calculus AV was defined in [10]. It has an infinite set of variables denoted $x, y, \ldots$, possibly with subscripts. Function symbols and terms of AV are defined inductively as follows:

- The constant 0 and the variables are terms.
- s_0, s_1, tr, $mod2$, S and len are unary function symbols, $trunc$ and $\#$ are binary function symbols and $cond$ is a ternary function symbol. These are the *primitive* function symbols of AV.
- If t is a term whose free variables are among $x_1, \ldots, x_n$, then $[\lambda x_1 \ldots x_n.t]$ is an n-ary function symbol.
- If g is an n-ary function symbol and h_0, h_1 are $(n+1)$-ary function symbols, then $CR[g, h_0, h_1]$ is an $(n+1)$-ary function symbol.
- If f is an n-ary function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

For sake of readability, the function symbol $\#$ is written infix, and we write $|t|$ for $len(t)$, 1 for $s_1(0)$ and $t0$ and $t1$ for $s_0(t)$ and $s_1(t)$ respectively. The function symbol $mod2$ is denoted $parity$ in [10]. Furthermore, AV as defined there has an additional function symbol pad, which is redundant since it can be defined as $CR[[\lambda x.x], [\lambda xy.0], [\lambda xy.0]]$. AV has a set of axioms that are sufficient to evaluate every closed term to a normal form built up from 0, s_0 and s_1 only. Some of these axioms of AV are

$$\begin{gathered} s_0(0) = 0, \quad mod2(x0) = 0,\ mod2(x1) = 1, \quad S(x0) = x1,\ S(x1) = s_0(S(x)), \\ cond(0, y, z) = y,\ cond(x0, y, z) = cond(x, y, z),\ cond(x1, y, z) = z, \\ [\lambda \bar{x}.t](\bar{x}) = t \\ CR[g, h_0, h_1](\bar{x}, y0) = cond(y, g(\bar{x}), cond(h_0(\bar{x}, y0), \bar{c}0, \bar{c}1)) \\ CR[g, h_0, h_1](\bar{x}, y1) = cond(h_1(\bar{x}, y1), \bar{c}0, \bar{c}1) \end{gathered}$$

where in the last two lines $\bar{c}$ is an abbreviation for $CR[g, h_0, h_1](\bar{x}, y)$. The rules of AV are the ususal rules of equational logic (symmetry, transitivity, congruence and substitution) and a special rule of induction on notation:

$$\frac{\begin{array}{cc} t_1[0] = t_2[0] & \\ t_1[x0] = v_0[t_1[x]] & t_2[x0] = v_0[t_2[x]] \\ t_1[x1] = v_1[t_1[x]] & t_2[x1] = v_1[t_2[x]] \end{array}}{t_1[x] = t_2[x]}$$

By Prop. 1, the function symbols in AV represent exactly the functions in AC^0. Based on Prop. 2, we define the equational calculi $A2V$ and TV whose function symbols represent exactly the functions in $ACC(2)$ and TC^0 respectively. They are defined like AV, but have additional primitive function symbols with axioms on them. $A2V$ has the additional unary function symbol $parity$ with the axioms

$$(\dagger) \qquad \begin{gathered} parity(0) = 0\,,\ parity(x0) = parity(x)\,, \\ parity(x1) = cond(parity(x), 1, 0)\,. \end{gathered}$$

TV has the additional unary function symbol $count$ with the axioms

$$(\ddagger) \qquad count(0) = 0\,,\ count(x0) = count(x)\,,\ count(x1) = S(count(x))\,.$$

The simulation

For every equation $t = u$ of AV, a family of propositional tautologies $|t = u|^n$ for $n \geq 0$ is defined, where $|t = u|^b$ expresses the fact that the equality $t = u$ holds for all values of the variables whose lengths are bounded by $\bar{b}$. We shall only sketch this definition, the reader is referred to [10] for the complete definition.

First, for every function symbol f a bounding polynomial $bound_f$ is defined, e.g. we define

$$\begin{gathered} bound_{s_0}(b) = bound_{s_1}(b) = bound_S(b) = b + 1\,, \\ bound_{cond}(b_1, b_2, b_3) = b_2 + b_3\,. \end{gathered}$$

This definition is extended inductively to arbitrary terms by

$$\begin{gathered} bound_0 = 0\,, \qquad bound_x(b) = b\,, \\ bound_{f(t_1(\bar{x}), \ldots, t_m(\bar{x}))}(\bar{b}) = bound_f(bound_{t_1(\bar{x})}(\bar{b}), \ldots, bound_{t_m(\bar{x})}(\bar{b}))\,. \end{gathered}$$

The polynomial $bound_{t(\bar{x})}(\bar{b})$ has the property that for values of the variables whose lengths are bounded by $\bar{b}$, the value of $t(\bar{x})$ is bounded in length by $bound_{t(\bar{x})}(\bar{b})$.

For every variable x of AV let $Q_i[x]$ and $P_i[x]$ be propositional variables who are intended to say $|x| > i$ and "the ith bit in x is 1" respectively. Furthermore let P be a variable that is different from all these, and let $\bot$ and $\top$ abbreviate $P \wedge \neg P$ and $P \vee \neg P$ respectively. Then for each term t whose variables are among the $x_1, \ldots, x_m$ and non-negative integers i and $b_1, \ldots, b_m$ two propositional formulas $q_i^{b_1,\ldots,b_m}[t]$ and $p_i^{b_1,\ldots,b_m}[t]$ in these variables are defined. The intended meaning of these formulas is $|t| > i$ and "the ith bit in t is 1", provided that $|x_j| \leq b_j$ for each $j \leq m$.

These formulas are defined inductively. First we define $q_i[0] = p_i[0] = \bot$, and for a variable x

$$q_i^b[x] = \begin{cases} Q_i[x] & \text{if } i < b \\ \bot & \text{else} \end{cases} \qquad p_i^b[x] = \begin{cases} P_i[x] & \text{if } i < b \\ \bot & \text{else} \end{cases}$$

Then the formulas $q_i^{\bar{b}}[t]$ and $p_i^{\bar{b}}[t]$ are defined for terms consisting of a primitive function symbol applied to variables, e.g.

$$q_i^b[x0] = \begin{cases} q_0^b[x] & \text{if } i = 0 \\ q_{i-1}^b[x] & \text{else} \end{cases} \qquad p_i^b[x0] = \begin{cases} \bot & \text{if } i = 0 \\ p_{i-1}^b[x] & \text{else} \end{cases}$$

$$q_i^b[x1] = \begin{cases} \bot & \text{if } i = 0 = b \\ \top & \text{if } i = 0 < b \\ q_{i-1}^b[x] & \text{else} \end{cases} \qquad p_i^b[x1] = \begin{cases} \bot & \text{if } i = 0 = b \\ \top & \text{if } i = 0 < b \\ p_{i-1}^b[x] & \text{else} \end{cases}$$

$$q^{b_1,b_2,b_3}[cond(x,y,z)] = (\neg q_0^{b_1}[x] \wedge q_i^{b_2}[y]) \vee (q_0^{b_1}[x] \wedge q_i^{b_3}[z])$$

$$p^{b_1,b_2,b_3}[cond(x,y,z)] = (\neg q_0^{b_1}[x] \wedge p_i^{b_2}[y]) \vee (q_0^{b_1}[x] \wedge p_i^{b_3}[z])$$

$$q_i^b[S(x)] = \begin{cases} q_i^b[x] \vee \bigwedge\limits_{j<i} p_j^b[x] & \text{if } i < b \\ \bot & \text{else} \end{cases}$$

$$p_i^b[S(x)] = \begin{cases} (p_i^b[x] \wedge \bigvee\limits_{j<i} \neg p_j^b[x]) \vee (\neg p_i^b[x] \wedge \bigwedge\limits_{j<i} p_j^b[x]) & \text{if } i > 0 \\ \neg p_i^b[x] & \text{else} \end{cases}$$

Now let $t = f(t_1(\bar{x}), \dots, t_m(\bar{x}))$, and let σ be the substitution replacing $Q_j[y_k]$ by $q_j^{\bar{b}}[t_k(\bar{x})]$ and $P_j[y_k]$ by $p_j^{\bar{b}}[t_k(\bar{x})]$ for each $j \leq m$, then we define

$$q_i^{\bar{b}}[t] := \sigma\big(q_i^{bound_{t_1(\bar{x})}(\bar{b}),\dots,bound_{t_m(\bar{x})}(\bar{b})}[f(y_1,\dots,y_m)]\big)$$

and $p_i^{\bar{b}}[t]$ analogously. The definition of $q_i^b[f(\bar{x})]$ and $p_i^b[f(\bar{x})]$ for compound function symbols is quite involved and is omitted here for sake of brevity.

For a variable x the formula $con^b[x]$ is defined as

$$\bigwedge_{i=0}^{b-2} q_{i+1}^b[x] \to q_i^b[x] \;\wedge\; \bigwedge_{i=0}^{b-1} p_i^b[x] \to q_i^b[x] \;\wedge\; \bigwedge_{i=1}^{b} len_i^b[x] \to p_{i-1}^b[x]$$

where $len_i^b[x]$ is defined as $q_{i-1}^b[x] \wedge \neg q_i^b[t]$. The formula $|t = u|_k^{\bar{b}}$ is

$$\bigwedge_{i=1}^{m} con^{b_i}[x_i] \to \bigwedge_{i=0}^{k-1} (q_i^{\bar{b}}[t] \iff q_i^{\bar{b}}[u]) \wedge (p_i^{\bar{b}}[t] \iff p_i^{\bar{b}}[u])$$

where the variables of t and u are among $x_1, \dots, x_m$. Finally, let $maxb_{t,u}(\bar{b})$ be an abbreviation for $\max(bound_t(\bar{b}), bound_u(\bar{b}))$, then we let

$$|t = u|^b := |t = u|^{b,\dots,b}_{maxb_{t,u}(b,\dots,b)} .$$

Now we are ready to state the following theorem, which was proved in [10].

THEOREM 3. *If $AV \vdash t = u$, then the tautologies $|t = u|^n$ for $n \geq 0$ have polynomial size, constant depth Frege proofs.*

We shall now extend the translation defined above in such a way that equations in the languages of $A2V$ and TV are mapped to families of tautologies in the languages of $PK\oplus$ and PTK respectively.

The definition of the bounding polynomials $bound_t$ is extended by the clauses for the additional function symbols

$$bound_{parity}(b) = 1 \qquad bound_{count}(b) = b .$$

The definition of the formulas $q_i^{\bar{b}}[t]$ and $p_i^{\bar{b}}[t]$ is also extended by clauses for the additional primitive function symbols. For the additional function symbol *parity* of $A2V$ we define

$$q_0^b[parity(x)] = p_0^b[parity(x)] = \begin{cases} \bot & \text{if } b = 0 \\ p_0^b[x] & \text{if } b = 1 \\ \bigoplus^b(p_0^b[x], \dots, p_{b-1}^b[x]) & \text{else} \end{cases},$$

$$q_i^b[parity(x)] = p_i^b[parity(x)] = \bot \quad \text{for } i > 0.$$

For the additional function symbol *count* of TV we first define the PTK-formula $cnt_i^b[x]$

$$cnt_i^b[x] = \begin{cases} \neg T_1^b(p_0^b[x], \dots, p_{b-1}^b[x]) & \text{if } i = 0 \\ T_i^b(p_0^b[x], \dots, p_{b-1}^b[x]) \wedge \neg T_{i+1}^b(p_0^b[x], \dots, p_{b-1}^b[x]) & \text{if } 1 \leq i < b \\ T_b^b(p_0^b[x], \dots, p_{b-1}^b[x]) & \text{if } i = b \\ \bot & \text{else} \end{cases},$$

and then

$$q_i^b[count(x)] = \begin{cases} T_{2^i}^b(p_0^b[x], \dots, p_{b-1}^b[x]) & \text{if } 2^i \leq b \\ \bot & \text{else} \end{cases},$$

$$p_i^b[count(x)] = \bigvee_{\substack{j \leq b \\ j \ni i}} cnt_j^b[x] ,$$

where $j \ni i$ means that the ith bit in j is 1. With these additional clauses, the families $|t = u|^n$ for equations $t = u$ of $A2V$ and TV are defined as above, and we can state our main theorem.

THEOREM 4. *If $A2V \vdash t = u$, then the tautologies $|t = u|^n$ for $n \geq 0$ have polynomial size, constant depth proofs in $PK\oplus$. If $TV \vdash t = u$, then the tautologies $|t = u|^n$ for $n \geq 0$ have polynomial size, constant depth proofs in PTK^*.*

By the above mentioned equivalences it follows that proofs in $A2V$ and TV can be simulated by polynomial size, constant depth proofs in $F(Mod_2)$ and FC respectively.

PROOF. Since both $PK\oplus$ and PTK^* can polynomially simulate a Frege system, where the simulations increase the depth at most by a constant, there are polynomial size, constant depth proofs of $|t = u|^n$ for every axiom $t = u$ of AV by Thm. 3. In Lemmas 5 and 6 below, we shall show that the translations of the additional axioms of $A2V$ and TV have polynomial size, constant depth proofs in $PK\oplus$ and PTK^*, respectively.

To complete the proof, it remains to show that for the rules of the equational calculi $A2V$ and TV, we get a polynomial size, constant depth proof of the conclusion from polynomial size, constant depth proofs of the premises, in both $PK\oplus$ and PTK^*.

Since the rules of $A2V$ and TV are the same as those of AV and constant depth Frege proofs of polynomial size can be simulated by polynomial size, constant depth proofs in $PK\oplus$ and PTK^*, the proof of this for the case of AV in [10] can be adapted to our case. The only change necessary is the incorporation of the additional function symbols in those places where the proof uses induction on the

Then the formulas $q_i^{\bar{b}}[t]$ and $p_i^{\bar{b}}[t]$ are defined for terms consisting of a primitive function symbol applied to variables, e.g.

$$q_i^b[x0] = \begin{cases} q_0^b[x] & \text{if } i = 0 \\ q_{i-1}^b[x] & \text{else} \end{cases} \qquad p_i^b[x0] = \begin{cases} \bot & \text{if } i = 0 \\ p_{i-1}^b[x] & \text{else} \end{cases}$$

$$q_i^b[x1] = \begin{cases} \bot & \text{if } i = 0 = b \\ \top & \text{if } i = 0 < b \\ q_{i-1}^b[x] & \text{else} \end{cases} \qquad p_i^b[x1] = \begin{cases} \bot & \text{if } i = 0 = b \\ \top & \text{if } i = 0 < b \\ p_{i-1}^b[x] & \text{else} \end{cases}$$

$$q^{b_1,b_2,b_3}[cond(x,y,z)] = (\neg q_0^{b_1}[x] \wedge q_i^{b_2}[y]) \vee (q_0^{b_1}[x] \wedge q_i^{b_3}[z])$$

$$p^{b_1,b_2,b_3}[cond(x,y,z)] = (\neg q_0^{b_1}[x] \wedge p_i^{b_2}[y]) \vee (q_0^{b_1}[x] \wedge p_i^{b_3}[z])$$

$$q_i^b[S(x)] = \begin{cases} q_i^b[x] \vee \bigwedge\limits_{j<i} p_j^b[x] & \text{if } i < b \\ \bot & \text{else} \end{cases}$$

$$p_i^b[S(x)] = \begin{cases} (p_i^b[x] \wedge \bigvee\limits_{j<i} \neg p_j^b[x]) \vee (\neg p_i^b[x] \wedge \bigwedge\limits_{j<i} p_j^b[x]) & \text{if } i > 0 \\ \neg p_i^b[x] & \text{else} \end{cases}$$

Now let $t = f(t_1(\bar{x}), \dots, t_m(\bar{x}))$, and let σ be the substitution replacing $Q_j[y_k]$ by $q_j^{\bar{b}}[t_k(\bar{x})]$ and $P_j[y_k]$ by $p_j^{\bar{b}}[t_k(\bar{x})]$ for each $j \leq m$, then we define

$$q_i^{\bar{b}}[t] := \sigma\big(\, q_i^{bound_{t_1(x)}(\bar{b}),\dots,bound_{t_m(x)}(\bar{b})}[f(y_1,\dots,y_m)]\,\big)$$

and $p_i^{\bar{b}}[t]$ analogously. The definition of $q_i^b[f(\bar{x})]$ and $p_i^b[f(\bar{x})]$ for compound function symbols is quite involved and is omitted here for sake of brevity.

For a variable x the formula $con^b[x]$ is defined as

$$\bigwedge_{i=0}^{b-2} q_{i+1}^b[x] \to q_i^b[x] \;\wedge\; \bigwedge_{i=0}^{b-1} p_i^b[x] \to q_i^b[x] \;\wedge\; \bigwedge_{i=1}^{b} len_i^b[x] \to p_{i-1}^b[x]$$

where $len_i^b[x]$ is defined as $q_{i-1}^b[x] \wedge \neg q_i^b[t]$. The formula $|t = u|_k^{\bar{b}}$ is

$$\bigwedge_{i=1}^{m} con^{b_i}[x_i] \to \bigwedge_{i=0}^{k-1} (q_i^{\bar{b}}[t] \iff q_i^{\bar{b}}[u]) \wedge (p_i^{\bar{b}}[t] \iff p_i^{\bar{b}}[u])$$

where the variables of t and u are among $x_1, \dots, x_m$. Finally, let $maxb_{t,u}(\bar{b})$ be an abbreviation for $\max(bound_t(\bar{b}), bound_u(\bar{b}))$, then we let

$$|t = u|^b := |t = u|^{b,\dots,b}_{maxb_{t,u}(b,\dots,b)}\,.$$

Now we are ready to state the following theorem, which was proved in [10].

THEOREM 3. *If $AV \vdash t = u$, then the tautologies $|t = u|^n$ for $n \geq 0$ have polynomial size, constant depth Frege proofs.*

We shall now extend the translation defined above in such a way that equations in the languages of $A2V$ and TV are mapped to families of tautologies in the languages of $PK\oplus$ and PTK respectively.

The definition of the bounding polynomials $bound_t$ is extended by the clauses for the additional function symbols

$$bound_{parity}(b) = 1 \qquad bound_{count}(b) = b\,.$$

The definition of the formulas $q_i^{\bar{b}}[t]$ and $p_i^{\bar{b}}[t]$ is also extended by clauses for the additional primitive function symbols. For the additional function symbol *parity* of $A2V$ we define

$$q_0^b[parity(x)] = p_0^b[parity(x)] = \begin{cases} \bot & \text{if } b = 0 \\ p_0^b[x] & \text{if } b = 1 \\ \bigoplus^b(p_0^b[x],\dots,p_{b-1}^b[x]) & \text{else} \end{cases},$$

$$q_i^b[parity(x)] = p_i^b[parity(x)] = \bot \quad \text{for } i > 0.$$

For the additional function symbol *count* of TV we first define the PTK-formula $cnt_i^b[x]$

$$cnt_i^b[x] = \begin{cases} \neg T_1^b(p_0^b[x],\dots,p_{b-1}^b[x]) & \text{if } i = 0 \\ T_i^b(p_0^b[x],\dots,p_{b-1}^b[x]) \wedge \neg T_{i+1}^b(p_0^b[x],\dots,p_{b-1}^b[x]) & \text{if } 1 \leq i < b \\ T_b^b(p_0^b[x],\dots,p_{b-1}^b[x]) & \text{if } i = b \\ \bot & \text{else} \end{cases},$$

and then

$$q_i^b[count(x)] = \begin{cases} T_{2^i}^b(p_0^b[x],\dots,p_{b-1}^b[x]) & \text{if } 2^i \leq b \\ \bot & \text{else} \end{cases},$$

$$p_i^b[count(x)] = \bigvee_{\substack{j \leq b \\ j \ni i}} cnt_j^b[x]\,,$$

where $j \ni i$ means that the ith bit in j is 1. With these additional clauses, the families $|t = u|^n$ for equations $t = u$ of $A2V$ and TV are defined as above, and we can state our main theorem.

THEOREM 4. *If $A2V \vdash t = u$, then the tautologies $|t = u|^n$ for $n \geq 0$ have polynomial size, constant depth proofs in $PK\oplus$. If $TV \vdash t = u$, then the tautologies $|t = u|^n$ for $n \geq 0$ have polynomial size, constant depth proofs in PTK^*.*

By the above mentioned equivalences it follows that proofs in $A2V$ and TV can be simulated by polynomial size, constant depth proofs in $F(Mod_2)$ and FC respectively.

PROOF. Since both $PK\oplus$ and PTK^* can polynomially simulate a Frege system, where the simulations increase the depth at most by a constant, there are polynomial size, constant depth proofs of $|t = u|^n$ for every axiom $t = u$ of AV by Thm. 3. In Lemmas 5 and 6 below, we shall show that the translations of the additional axioms of $A2V$ and TV have polynomial size, constant depth proofs in $PK\oplus$ and PTK^*, respectively.

To complete the proof, it remains to show that for the rules of the equational calculi $A2V$ and TV, we get a polynomial size, constant depth proof of the conclusion from polynomial size, constant depth proofs of the premises, in both $PK\oplus$ and PTK^*.

Since the rules of $A2V$ and TV are the same as those of AV and constant depth Frege proofs of polynomial size can be simulated by polynomial size, constant depth proofs in $PK\oplus$ and PTK^*, the proof of this for the case of AV in [10] can be adapted to our case. The only change necessary is the incorporation of the additional function symbols in those places where the proof uses induction on the

complexity of a term in AV. It is possible, although tedious, to show that these inductive arguments remain valid for terms in $A2V$ and TV. □

It remains to prove the promised lemmas, which will almost take the rest of the paper.

LEMMA 5. *The translations of the axioms* (†) *of A2V have polynomial size, constant depth proofs in* $PK\oplus$.

PROOF. The formulas $|parity(0) = 0|^n$ do not depend on n and are true, hence they obviously have polynomial size, constant depth proofs in $PK\oplus$.

Now we have to prove in $PK\oplus$ the formulas $|parity(x0) = parity(x)|_1^b$. The formulas $q_0^b[parity(x0)]$ and $p_0^b[parity(x0)]$ are both

$$\bigoplus^{b+1}(p_0^{b+1}[x0], \dots, p_b^{b+1}[x0]) = \bigoplus^{b+1}(\bot, P_0[x], \dots, P_{b-1}[x]) ,$$

and the formulas $q_0^b[parity(x)]$ and $p_0^b[parity(x0)]$ are both $\bigoplus^b(P_0[x], \dots, P_{b-1}[x])$. Thus we prove both required equivalences without using the assumption $con^b[x]$ by giving short proofs of

$$\bigoplus^{k+1}(\bot, A_1, \dots, A_k) \iff \bigoplus^k(A_1, \dots, A_k)$$

for propositional variables $A_1, \dots, A_k$. These proofs have a constant number of steps, hence are of linear size, since we defined $\bigoplus^{k+1}$ by association to the left.

Finally we have to give proofs of $|parity(x1) = cond(parity(x), 1, 0)|_1^b$. The formulas $q_0^b[parity(x1)]$ and $p_0^b[parity(x1)]$ are by definition both

$$\bigoplus^{b+1}(\top, P_0[x], \dots, P_{b-1}[x]) ,$$

and the formulas $q_0^b[cond(parity(x), 1, 0)]$ and $p_0^b[cond(parity(x), 1, 0)]$ are

$$(\neg q_0^b[parity(x)] \wedge \top) \vee (q_0^b[parity(x)] \wedge \bot) . \tag{1}$$

Their equivalence can again be proved without use of the assumption $con^b[x]$. The formulas (1) are shown to be equivalent to $\neg q_0^b[parity(x)]$ by short, constant depth proofs without use of the $\oplus$-rules, hence it remains to prove

$$\bigoplus^{k+1}(\top, A_1, \dots, A_k) \iff \neg\bigoplus^k(A_1, \dots, A_k)$$

for propositional variables $A_1, \dots, A_k$. These equivalences are again easily seen to have short proofs in $PK\oplus$. □

LEMMA 6. *The translations of the axioms* (‡) *of TV have polynomial size, constant depth proofs in* PTK^*.

PROOF. The formulas $|count(0) = 0|^n$ are again true formulas that do not depend on n, hence there are trivially polynomial size, constant depth proofs in PTK^* of them.

We have to give PTK^*-proofs of the formulas $|count(x0) = count(x)|_{b+1}^b$. So under the hypothesis $con^b[x]$, which will in fact not be needed, we have to deduce

$$q_i^b[count(x0)] \iff q_i^b[count(x)] \qquad \text{and} \qquad p_i^b[count(x0)] \iff p_i^b[count(x)]$$

for every $i \le b$. For i with $2^i \le b+1$, the formula $q_i^b[count(x0)]$ is defined as $T_{2^i}^{b+1}(p_0^{b+1}[x0], \dots, p_b^{b+1}[x0])$, which is $T_{2^i}^{b+1}(\bot, P_0[x], \dots, P_{b-1}[x])$. On the other hand, $q_i^b[count(x)]$ is $T_{2^i}^b(P_0[x], \dots, P_{b-1}[x])$ for i with $2^i \le b$, and $\bot$ for $2^i > b$. Furthermore, the formulas $p_i^b[count(x0)]$ and $p_i^b[count(x)]$ are

$$\bigvee_{\substack{j \le b+1 \\ j \ni i}} cnt_j^{b+1}[x0] \qquad \text{and} \qquad \bigvee_{\substack{j \le b \\ j \ni i}} cnt_j^b[x] .$$

To show their equivalence, we have to prove $cnt_j^b[x] \iff cnt_j^{b+1}[x0]$ for every $j \le b$ and $\neg cnt_{b+1}^{b+1}[x0]$ in PTK^*. All the required formulas are deduced by short, constant depth proofs from

$$T_k^{m+1}(\bot, A_1, \dots, A_m) \iff T_k^m(A_1, \dots, A_m)$$

for $k \le m$ and $\neg T_{m+1}^{m+1}(\bot, A_1, \dots, A_m)$, for variables $A_1, \dots, A_m$. Short proofs of these equivalences are easily given using the rules for T_k^n.

The most difficult part is to give proofs of $|count(x1) = S(count(x))|_{b+1}^b$. First we will give proofs of the equivalences $q_i^b[count(x1)] \iff q_i^b[S(count(x))]$ for $i \le b$ without using the assumption $con^b[x]$.

The formula $q_i^b[count(x1)]$ is by definition $T_{2^i}^{b+1}(\top, P_0[x], \dots, P_{b-1}[x])$ if $2^i \le b+1$ and $\bot$ else, and the formula $q_i^b[S(count(x))]$ is

$$q_i^b[count(x)] \vee \bigwedge_{j<i} \bigvee_{\substack{k \le b \\ k \ni j}} cnt_k^b[x] .$$

hence we have to show

$$\text{(I)} \quad T_{2^i}^{b+1}(\top, \tilde{P}[x]) \iff T_{2^i}^b(\tilde{P}[x]) \vee \bigwedge_{j<i} \bigvee_{\substack{k \le b \\ k \ni j}} cnt_k^b[x] \qquad \text{for } 2^i \le b,$$

$$\text{(II)} \quad T_{2^i}^{b+1}(\top, \tilde{P}[x]) \iff \bigwedge_{j<i} \bigvee_{\substack{k \le b \\ k \ni j}} cnt_k^b[x] \qquad \text{for } 2^i = b+1,$$

$$\text{(III)} \quad \neg \bigwedge_{j<i} \bigvee_{\substack{k \le b \\ k \ni j}} cnt_k^b[x] \qquad \text{for } 2^i > b+1,$$

where $\tilde{P}[x]$ is short for $P_0[x], \dots, P_{b-1}[x]$. For this, we shall need short proofs of the sequents

$$T_j^k(A_1, \dots, A_k) \Longrightarrow T_{j-1}^k(A_1, \dots, A_k) \tag{2}$$

for every $1 < j \le k$ and variables $A_1, \dots, A_k$. These are easily deduced using the rules T_k^n-left2 and T_k^n-right2. By use of (2), one can give proofs of the sequents $cnt_\mu^b[x], cnt_\nu^b[x] \Longrightarrow$ for every $\mu < \nu \le b$, of size $O(b(\nu - \mu))$.

We treat (II) first. The direction from left to right of the equivalence is obtained by a $\wedge$-right inference from the sequents

$$T_{2^i}^{b+1}(\top, \tilde{P}[x]) \Longrightarrow \bigvee_{\substack{k \le b \\ k \ni j}} cnt_k^b[x] \qquad \text{for } j < i,$$

which we get by weakening and $\vee$-right from $T_{2^i}^{b+1}(\top, \tilde{P}[x]) \Longrightarrow cnt_b^b[x]$, since $b = 2^i - 1$, and hence $b \ni j$ for every $j < i$. These last sequents are by definition

$T^{b+1}_{b+1}(\top, \tilde{P}[x]) \implies T^b_b(\tilde{P}[x])$ and are easily deduced by the $\wedge$-rules. For the other direction, we have to give proofs of

$$\bigwedge_{j<i} \bigvee_{\substack{k\leq b \\ k\ni j}} cnt^b_k[x] \implies P_\ell[x]$$

for each $\ell \leq b-1$, from which together with $\implies \top$ we obtain the desired sequent by a $\wedge$-right inference. The sequent above is obtained by $\wedge$-left and a cut from $cnt^b_b[x] \implies P_\ell[x]$, which are easily easily derived as $b = 2^i - 1$, and

$$\bigvee_{\substack{k\leq b \\ k\ni 0}} cnt^b_k[x], \dots, \bigvee_{\substack{k\leq b \\ k\ni(i-1)}} cnt^b_k[x] \implies cnt^b_b[x]\,. \tag{3}$$

Each of the disjunctions on the left has $\lceil \frac{b+1}{2} \rceil$ terms. This sequent is deduced by two applications of $\vee$-left from $\lceil \frac{b+1}{2} \rceil^2$ sequents of the form

$$cnt^b_\mu[x], cnt^b_\nu[x], \bigvee_{\substack{k\leq b \\ k\ni 2}} cnt^b_k[x], \dots, \bigvee_{\substack{k\leq b \\ k\ni(i-1)}} cnt^b_k[x] \implies cnt^b_b[x]\,.$$

For $\mu \neq \nu$, these sequents have short proofs using (2), and the remaining ones with $\mu = \nu$ are again obtained by $\vee$-left from $\lceil \frac{b+1}{2} \rceil$ premises of the form

$$cnt^b_\nu[x], cnt^b_\nu[x], cnt^b_{\mu'}[x], \bigvee_{\substack{k\leq b \\ k\ni 3}} cnt^b_k[x], \dots, \bigvee_{\substack{k\leq b \\ k\ni(i-1)}} cnt^b_k[x] \implies cnt^b_b[x]$$

for each such ν. But there are only $\lceil \frac{b+1}{4} \rceil$ values of ν for which $cnt^b_\nu[x]$ occurs in the first *and* second disjunction in (3). Again, most of these sequents have short proofs using (2), except for the $\lceil \frac{b+1}{8} \rceil$ of them with $\mu' = \nu$. After $i-1$ iterations of this process, the only remaining sequent to be deduced is the trivial

$$cnt^b_b[x], \dots, cnt^b_b[x] \implies cnt^b_b[x]\,,$$

since $b = 2^i - 1$ is the only value $k \leq b$ for which $k \ni j$ holds for every $j < i$. The size of these derivations can be calculated as follows: Each of the short proofs using (2) is of size $O(b^2)$, hence the whole proof is of size

$$O(b^2) \cdot \left\lceil \frac{b+1}{2} \right\rceil \cdot \sum_{1\leq j<i} \left\lceil \frac{b+1}{2^j} \right\rceil = O(b^4)\,.$$

For case (III), we have to deduce the sequent

$$\bigvee_{\substack{k\leq b \\ k\ni 0}} cnt^b_k[x], \dots, \bigvee_{\substack{k\leq b \\ k\ni(i-1)}} cnt^b_k[x] \implies .$$

A proof of this is constructed analogously to the proof of (3) above, where this time there is no sequent remaining after the $i-1$ steps, since there is no value $k \leq b$ for which $k \ni j$ holds for every $j < i$.

For case (I), observe that the following sequent is easily deduced:

$$T^b_{2^i-1}(\tilde{P}[x]) \implies T^b_{2^i}(\tilde{P}[x]), cnt^b_{2^i-1}[x]\,.$$

Since $2^i - 1 \ni j$ for every $j < i$, we get from this like in the proof of the first direction of (II)

$$T^b_{2^i-1}(\tilde{P}[x]) \implies T^b_{2^i}(\tilde{P}[x]), \bigwedge_{j<i} \bigvee_{\substack{k\leq b \\ k\ni j}} cnt^b_k[x]\,,$$

from which we obtain the left-to-right direction of (I) by $T^{b+1}_{2^i}$-left2. For the other direction, we first need proofs of

$$\bigwedge_{j<i} \bigvee_{\substack{k\leq b \\ k\ni j}} cnt^b_k[x] \implies T^b_{2^i-1}(\tilde{P}[x])\,. \tag{4}$$

These proofs can again be constructed by the method given for (3) in (II), since every value k for which $k \ni j$ for each $j < i$ is at least $k \geq 2^i - 1$. Now from (4) and $\implies \top$ one gets by a $T^{b+1}_{2^i}$-right1

$$\bigwedge_{j<i} \bigvee_{\substack{k\leq b \\ k\ni j}} cnt^b_k[x] \implies T^{b+1}_{2^i}(\top, \tilde{P}[x])$$

and from this and $T^b_{2^i}(\tilde{P}[x]) \implies T^{b+1}_{2^i}(\top, \tilde{P}[x])$, a $\vee$-left yields the right-to-left direction of (I), which completes the proof of $q^b_i[count(x1)] \iff q^b_i[S(count(x)]$.

Now we give proofs $p^b_i[count(x1)] \iff p^b_i[S(count(x)]$, again without using the assumption $con^b[x]$. For this, we first need short proofs of the equivalence

$$cnt^b_j[x] \iff cnt^{b+1}_{j+1}[s_1(x)]\,, \tag{5}$$

which can easily be given using (2). For $i = 0$, by definition we have to prove the equivalence

$$\bigvee_{\substack{j\leq b \\ j\ni 0}} cnt^b_j[x1] \iff \neg \bigvee_{\substack{j\leq b \\ j\ni 0}} cnt^b_j[x]\,.$$

For the left-to-right direction, by (5) it is sufficient to deduce

$$\bigvee_{\substack{j\leq b \\ j \text{ odd}}} cnt^b_j[x], \bigvee_{\substack{j\leq b \\ j \text{ even}}} cnt^b_j[x] \implies ,$$

which is obtained by two applications of $\vee$-left from $\lceil \frac{b+1}{2} \rceil^2$ premises of the form $cnt^b_\mu, cnt^b_\nu \implies$ for $\mu \leq b$ odd and $\nu \leq b$ even. These premises have, as noted above, short proofs using (2). For the other direction, we first show by induction that there are proofs of the sequents

$$T^b_k(\tilde{P}[x]) \implies cnt^b_k[x], \dots, cnt^b_b[x] \tag{6}$$

for every $k \leq b$ of size $O(b(b-k+1))$. This is trivial for $k = b$, and a proof of the sequent (6) for $k-1$ is easily given using (6) for k. This yields a proof of size $O(b^2)$ of the sequent

$$\implies cnt^b_0[x], cnt^b_1[x], \dots, cnt^b_b[x]\,.$$

Using the equivalence (5), we can deduce from this

$$\implies \bigvee_{\substack{j\leq b \\ j \text{ odd}}} cnt^b_j[x], \bigvee_{\substack{j\leq b+1 \\ j \text{ odd}}} cnt^{b+1}_j[x1]\,,$$

which yields the desired right-to-left direction and thus completes the case $i = 0$.

For $i > 0$, the formula $p_i^b[S(count(x))]$ is

$$\big(p_i^b[count(x)] \wedge \bigvee_{j<i} \neg p_j^b[count(x)]\big) \vee \big(\neg p_i^b[count(x)] \wedge \bigwedge_{j<i} p_j^b[count(x)]\big)\,,$$

thus the left-to-right direction of $p_i^b[count(x1)] \iff p_i^b[S(count(x))]$ follows by a short, constant depth proof using (5) from the two sequents

$$(7)\qquad \bigvee_{\substack{j\le b\\(j+1)\ni i}} cnt_j^b[x]\,,\, p_i^b[count(x)] \implies \neg \bigwedge_{j<i} p_j^b[count(x)]$$

$$(8)\qquad \bigvee_{\substack{j\le b\\(j+1)\ni i}} cnt_j^b[x]\,,\, \neg p_i^b[count(x)] \implies \bigwedge_{j<i} p_j^b[count(x)]\,.$$

Recalling the definition of $p_j^b[count(x)]$, we see that the sequent (7) is obtained from at most $\lceil \frac{b+1}{2} \rceil$ sequents of the form

$$cnt_\nu^b[x]\,,\, \bigvee_{\substack{j\le b\\j\ni 0}} cnt_j^b[x]\,,\,\ldots\,,\, \bigvee_{\substack{j\le b\\j\ni i}} cnt_j^b[x] \implies,$$

where ν is such that $(\nu+1) \ni i$. Since $\nu \ni k$ for all $k \le i$ would imply $(\nu+1) \not\ni i$, the formula $cnt_\nu^b[x]$ cannot appear in all of the disjunctions. Let k_0 be such that $\nu \not\ni k_0$, then we obtain the sequent above from at most $\lceil \frac{b+1}{2} \rceil$ sequents of the form

$$cnt_\nu^b[x]\,,\, cnt_\kappa^b[x]\,,\, \bigvee_{\substack{j\le b\\j\ni 0}} cnt_j^b[x]\,,\ldots\,,\, \bigvee_{\substack{j\le b\\j\ni i}} cnt_j^b[x] \implies,$$

for each κ with $\kappa \ni k_0$ and hence $\kappa \neq \nu$, which have short constant depth PTK^*-proofs. Next (8) is obtained from at most $\lceil \frac{b+1}{2} \rceil$ sequents of the form

$$(9)\qquad cnt_\nu^b[x] \implies \bigvee_{\substack{j\le b\\j\ni i}} cnt_j^b[x]\,,\, \bigwedge_{j<i} \bigvee_{\substack{k\le b\\k\ni j}} cnt_k^b[x]$$

with $(\nu+1) \ni i$. Now if $\nu \ni i$, then (9) is obtained by weakening and $\vee$-right from an axiom since $cnt_\nu^b[x]$ appears in the first disjunction. Otherwise $\nu \ni j$ must hold for every $j < i$, hence we get (9) from i sequents

$$cnt_\nu^b[x] \implies \bigvee_{\substack{j\le b\\j\ni i}} cnt_j^b[x]\,,\, \bigvee_{\substack{k\le b\\k\ni j}} cnt_k^b[x]$$

for $j < i$, which can then be obtained as above since $cnt_\nu^b[x]$ must appear in the second disjunction.

Finally the right-to-left direction $p_i^b[S(count(x))] \to p_i^b[count(x1)]$ is deduced by short proofs using (5) from the two sequents

$$(10)\qquad p_i^b[count(x)] \implies \bigwedge_{j<i} p_j^b[count(x)]\,,\, \bigvee_{\substack{j\le b\\(j+1)\ni i}} cnt_j^b[x]$$

$$(11)\qquad p_0^b[count(x)]\,,\ldots\,,\, p_{i-1}^b[count(x)] \implies p_i^b[count(x)]\,,\, \bigvee_{\substack{j\le b\\(j+1)\ni i}} cnt_j^b[x]\,.$$

The sequent (10) is obtained by $\vee$-left and $\wedge$-right from $i \cdot \lceil \frac{b}{2} \rceil$ sequents of the form

$$cnt_\nu^b[x] \implies \bigvee_{\substack{k\le b\\k\ni j}} cnt_k^b[x]\,,\, \bigvee_{\substack{j\le b\\(j+1)\ni i}} cnt_j^b[x]$$

for ν with $\nu \ni i$ and $j < i$. Now if ν is such that $(\nu+1) \ni i$, then $cnt_\nu^b[x]$ appears in the second disjunction. Otherwise it must be the case that $\nu \ni j$ for every $j < i$, hence $cnt_\nu^b[x]$ appears in the first disjunction. In both cases the sequent above is deduced by weakenings and $\vee$-right from an initial sequent.

By the method used for (3) above, (11) can be deduced using (2) from the sequents

$$cnt_\nu^b[x],\ldots,cnt_\nu^b[x] \implies \bigvee_{\substack{j\le b\\j\ni i}} cnt_j^b[x]\,,\, \bigvee_{\substack{j\le b\\(j+1)\ni i}} cnt_j^b[x]$$

for every ν with $\nu \ni k$ for every $k < i$. Now if $\nu \ni i$, then $cnt_\nu^b[x]$ appears in the first disjunction, and otherwise $(\nu+1) \ni i$, hence $cnt_\nu^b[x]$ appears in the second disjunction, hence in either case this sequent is easily deduced. □

Conclusion

We have presented equational calculi that prove equations between functions in $ACC(2)$ and TC^0, and shown that proofs in these can be simulated by polynomial size, constant depth proofs in Frege systems extended by modulo 2 counting and threshold connectives, respectively. It seems to be straightforward to define analogous calculi for the classes $ACC(m)$ for $m > 2$ and show these can be simulated by constant depth proofs in $F(Mod_m)$ in the same way. Besides supporting the intuitive correspondence between these complexity classes and proof systems, this provides us with a tool for proving the existence of polynomial size, constant depth proofs in these proof systems.

Actually, the relationship between PV and extended Frege proofs is much tighter than those presented in [10] and the present paper, in that extended Frege proofs are the maximal proof system among those whose correctness can be proved in PV. It should be possible, although tedious, to establish a similarly close connection between ALV from [10] and Frege proofs, using the fact that evaluation of boolean formulas can be done in NC^1 [6] (cf. also [16] for an effort in this direction).

To establish such a tight connection between TV, $A2V$ and AV and their corresponding proof systems, we have to overcome the obstacle that evaluation of boolean formulas is complete for NC^1, hence it is not possible in AC^0 and $ACC(2)$, and in TC^0 only if $TC^0 = NC^1$. Therefore it is not clear if the correctness of proofs can be expressed in these calculi.

The following remedy was suggested by P. Clote: The evaluation of threshold formulas of a fixed maximal depth d should be possible in TC^0, and by formalizing that we could then express the correctness of PTK-proofs of depth d by TV-terms. Then TV should be able to prove the correctness of PTK-proofs of depth d, for every d. A similar relationship might hold between $A2V$ and $F(Mod_2)$-proofs, as well as AV and Frege proofs.

References

[1] Miklos Ajtai, *The complexity of the pigeonhole principle*, 29th Sympos. on Foundations of Computer Science, IEEE, 1988, pp. 346–355.

[2] ———, *Parity and the pigeonhole principle*, Feasible Mathematics (Samuel R. Buss and Philip J. Scott, eds.), Birkhäuser, Boston, 1990, pp. 1–24.

[3] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák, *Lower bounds on Hilbert's Nullstellensatz and propositional proofs*, Proc. London Math. Soc. **73** (1996), 1–26.

[4] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, Pavel Pudlák, and Alan Woods, *Exponential lower bound for the pigeonhole principle*, Proc. 24th Sympos. Theory of Computing, 1992, pp. 200–221.

[5] Paul Beame and Toniann Pitassi, *An exponential separation between the matching principle and the pigeonhole principle*, Proc. LICS '93, 1993, pp. 308–319.

[6] Samuel R. Buss, *The Boolean formula value problem is in ALOGTIME*, Proceedings of the 19th Sympos. Theory of Computing, ACM, 1987, pp. 123–131.

[7] Samuel R. Buss and Peter Clote, *Cutting planes, connectivity and threshold logic*, Arch. Math. Logic **35** (1995), 34 ff.

[8] ———, *Threshold logic proof systems*, unpublished manuscript, 1995.

[9] Peter Clote, *Sequential, machine independent characterizations of the parallel complexity classes ALogTIME, AC^k, NC^k and NC*, Feasible Mathematics (Samuel R. Buss and Philip J. Scott, eds.), Birkhäuser, Boston, 1990, pp. 49–69.

[10] ———, *ALOGTIME and a conjecture of S. A. Cook*, Ann. Math. Artificial Intelligence **6** (1992), 57–106.

[11] Peter Clote and Gaisi Takeuti, *First order bounded arithmetic and small boolean circuit complexity classes*, Feasible Mathematics II (Peter Clote and Jeffrey Remmel, eds.), Birkhäuser, Boston, 1995, pp. 154–218.

[12] Alan Cobham, *The intrinsic computational difficulty of functions*, Proc. 2nd International Congress on Logic, Methodology and Philosophy of Science, 1965, pp. 24–30.

[13] Stephen A. Cook, *Feasible constructive proofs and the propositional calculus*, Proc. 7th Sympos. Theory of Computing, 1975, pp. 83–97.

[14] Stephen A. Cook and Robert A. Reckhow, *The relative efficiency of propositional proof systems*, J. Symbolic Logic **44** (1979), 36–50.

[15] Jan Krajíček, *On Frege and Extended Frege proof systems*, Feasible Mathematics II (Peter Clote and Jeffrey Remmel, eds.), Birkhäuser, Boston, 1995, pp. 284–319.

[16] François Pitt, *A quantifier-free theory based on a string algebra for NC^1*, this volume.

[17] Alasdair Urquhart, *Hard examples for resolution*, J. Assoc. Comput. Mach. **34** (1987), 209–219.

[18] ———, *The complexity of propositional proofs*, Bull. Symbolic Logic **1** (1995), 425–467.

INFORMATIK 1, UNIVERSITÄT ERLANGEN-NÜRNBERG, ERLANGEN, GERMANY

Current address: Department of Mathematics, University of California, San Diego, La Jolla, California 92093-0112

E-mail address: johannsn@math.ucsd.edu

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume **39**, 1998

Exponential Lower Bounds for Semantic Resolution

STASYS JUKNA

ABSTRACT. In a semantic resolution proof we operate with clauses only but allow *arbitrary* rules of inference: consistency is the only requirement. We prove a very simple exponential lower bound for the size of bounded fanin semantic resolution proofs of a general *Hitting Set Principle* stating that, for any set system with hitting set number τ, no set of size less than τ can be a hitting set. The pigeonhole principle and known blocking principles for finite (affine and projective) planes are special cases of this general principle.

1. Introduction

The resolution proof system introduced by Blacke [2] and further developed by Davis and Putnam [12] and Robinson [17] is one of the first and simplest in the hierarchy of propositional proof systems. This system operates with clauses and has one rule of inference

$$\frac{C_1 \vee x_i \quad C_2 \vee \neg x_i}{C_1 \vee C_2}$$

called the resolution rule. First exponential lower bound for regular resolution (these are resolution proofs with the additional restriction that along every path every particular variable x_i can be resolved at most once) was proved by Tseitin [18] almost 30 years ago. However, despite its apparent simplicity, the first lower bounds for non-regular resolution were proven only in 1985 by Haken [13]. These bounds were achieved for the pigeonhole principle PHP_n^{n+1} asserting that $n+1$ pigeons can not sit in n holes so that every pigeon is alone in his hole. Buss and Turán [8] extended this bound to $\exp\left(\Omega\left(n^2/m\right)\right)$ for more general form PHP_n^m of the pigeonhole principle in which the number of pigeons, m is another parameter. Haken's argument was further refined and applied to other tautologies by Urquhart [19] and Chvátal and Szemerédi [9].

In a recent work [1] Beame and Pitassi have found a direct and elegant proof of Haken's [13] lower bound for PHP_n^{n+1}. In this paper we simplify and generalize the combinatorial part of their argument and show that it in fact works: (i) for other principles than PHP_n^m, and (ii) for proof systems that generalize resolution. Examples of these new principles are, so-called, blocking principles for finite (affine and projective) planes. The model generalizing resolution is that of *semantic resolution*. Like in standard resolution proof, here we operate with clauses only. The

1991 *Mathematics Subject Classification.* Primary: 03F20, 68Q15; Secondary: 51E15, 51E21, 68R05.

Research supported by DFG grant Me 1077/10–1.

main difference is that we allow arbitrary inference rules

$$\frac{C_1,\ldots,C_l}{C}$$

Their *consistency* is the only requirement: every truth assignment satisfying all the hypotheses $C_1,\ldots,C_l$, must also satisfy the conclusion C. The number of hypotheses, l is the *fanin* of that rule. The resolution rule is a very special case of this general rule with $l=2$. The *size* (or *length*) of a proof is the total number of clauses in it.

We will prove a very simple exponential lower bound for the size of semantic resolution proofs of a general principle, which we call the *Hitting Set Principle*. This principle, $HS(\mathcal{F})$ states that for any set system $\mathcal{F}$ with hitting set number τ, no set A of size less than τ can be a hitting set. The *hitting* (or *blocking*) *set* for $\mathcal{F}$ is a set which hits (i.e. intersects) every edge of $\mathcal{F}$; the *hitting number* $\tau(\mathcal{F})$ is the minimal possible size of such a set. This number is hard-coded into the CNF, and the underlying variables are the variables describing A and other variables may be used to explain that A has the right size. This principle is a generalization of the pigeonhole principle PHP_n^m: here $\mathcal{F}$ consists of m mutually disjoint n-element sets. The sets A of interest are all sets of n points which define partial 1-to-1 mappings from n of the pigeons to the n holes. The pigeonhole principle states precisely that either A is too large, or some edge is not intersected by A (see also Example 2.2 below).

Our main result (Theorem 2.1) says that as long as the set system $\mathcal{F}$ satisfies certain combinatorial conditions, then any CNF formula formalizing the hitting set principle for $\mathcal{F}$ requires an exponentially long semantic resolution proof. Our argument is essentially the same argument employed by Beame and Pitassi [1] in the case of PHP_n^{n+1}. Using simple "greedy" algorithm, we first show that some small restriction will kill off all long clauses if the proof is short. Then we complete the proof with a direct argument that the remaining restricted proof cannot exist because there are no long clauses in it.

2. The lower bound

In this section we state our main result and describe several its applications. We first need to setup some notation. A *hypergraph* (or *set system*) over a set X is simply a family $\mathcal{F}$ of its subsets; elements of X are *points*, and sets in $\mathcal{F}$ are *edges*. We will be interested in the size of semantic resolution proofs for the hitting set principles $HS(\mathcal{F})$. In any such proof we have $|\mathcal{F}|$ leaves labeled by (positive) clauses $C_E=\bigvee_{i\in E}x_i$, one for each edge $E\in\mathcal{F}$. We call these leaves *primary*. All other leaves are *secondary* and may be labeled by arbitrary clauses. In particular, besides the x-variables (corresponding to points of $\mathcal{F}$) these clauses may contain any other variables. We require only that the conjunction of all these secondary clauses must be satisfiable on any set of size less than $\tau(\mathcal{F})$. For example, one can take as secondary the following set of clauses:

$$y_{i1}\vee\cdots\vee y_{im},\quad \neg y_{ik}\vee\neg y_{jk},\quad \neg y_{ik}\vee\neg x_k$$

where $1\le i\ne j\le m-\tau(\mathcal{F})+1$, $1\le k\le m$ and $m=|X|$ is the total number of points in the hypergraph $\mathcal{F}$. It is easy to see that, given an assignment to x-variables, this set of clauses is satisfiable if and only if the number of ones in that assignment is less than $\tau(\mathcal{F})$. For now, let us point out that our lower bounds

argument *does not* depend on the actual form of these secondary clauses: important will be only combinatorial properties of primary clauses, corresponding to the edges of $\mathcal{F}$.

Simple (but useful for the rest of the paper) observation is that, due to consistency of inference rules, any semantic resolution proof for $HS(\mathcal{F})$ is in fact a nondeterministic algorithm for the following *search problem*: Given a set A of size less than $\tau(\mathcal{F})$, find an edge $E\in\mathcal{F}$ such that $A\cap E=\emptyset$. To see this, associate with each such set A an assignment u_A to the remaining variables so that $f(v_A,u_A)=1$, where f is the conjunction of all secondary leaves and v_A is the incidence vector of A. Fix this injection $A\mapsto u_A$, and traverse the proof starting from the last clause of the proof (which is empty) by always choosing that of hypotheses C, for which $C(v_A,u_A)=0$. Consistency ensures that proceeding this way we will necessarily reach a leaf. The fact that $C(v_A,u_A)=1$ for all the secondary leaves C, ensures that the reached leaf is primary, i.e. has the form $C_E=\bigvee_{i\in E}x_i$ with $E\in\mathcal{F}$. Since $C_E(v_A,u_A)=0$, the edge E avoids our set A, and we are done: the desired edge is found.

This observation allows us to concentrate on the lower bounds problem on the length of semantic resolution proofs, solving the search problem for particular hypergraphs $\mathcal{F}$: any such bound is immediately a lower bound on the length of a shortest semantic resolution proof of any CNF, describing the hitting set principle for $\mathcal{F}$.

We will be particularly interested in special k-partite hypergraphs. Let $S_1,\ldots,S_k$ be mutually disjoint subsets of X, called *blocks*. A *partial transversal* is a set $B\subseteq X$ which intersects each block in at most one point; B is a *transversal* if $|B|=k$ (in this case B intersects each block in exactly one point).

Definition. We call a hypergraph $\mathcal{F}$ a (k,b,λ,d)-*design* if there exist k mutually disjoint blocks $S_1,\ldots,S_k$ such that:

1. Every edge of $\mathcal{F}$ is a transversal for $S_1,\ldots,S_k$;
2. $|S_i|\le b$ for all $i=1,\ldots,k$;
3. $|E\cap F|\le\lambda$ for all edges $E\ne F\in\mathcal{F}$;
4. Every point belongs to at most d edges of $\mathcal{F}$.

Such a design $\mathcal{F}$ is *large* if every transversal of $S_1,\ldots,S_k$ avoids at least one edge of $\mathcal{F}$. The corresponding *edge-search problem* for $\mathcal{F}$ is to find such an edge. Note that any design, with more than kd edges, is large, but there also are large designs with smaller number of edges.

Our main result is the following general lower bound for semantic resolution.

THEOREM 2.1. *Let $\mathcal{F}$ be a large (k,b,λ,d)-design, and G be a semantic resolution proof of fanin at most l. Let s and t be integers satisfying*

$$ls\le\min\{|\mathcal{F}|-dt,k-t\}\quad\text{and}\quad t\ge k/2 \tag{2.1}$$

If G solves the edge-search problem for $\mathcal{F}$ then

$$\text{size}(G)\ge 2^{M/b}\quad\text{where}\quad M=\frac{s(k-t-ls+1)^2}{k+\lambda\cdot(s-1)} \tag{2.2}$$

In particular, if $|\mathcal{F}|\ge k(d+1)/2$ then

$$\text{size}(G)\ge\exp\left(\Omega\left(\frac{k^2}{b(l+\lambda)}\right)\right). \tag{2.3}$$

Few words about the parameters. The bound (2.3) follows from (2.2) by taking $t = \lceil k/2 \rceil$ and $s = \lceil k/(4l) \rceil$. The bound itself becomes trivial if the block size b is near to the square of their number k. But this is inherent weakness of our argument (as well as all previous lower bound proofs for PHP_n^m): in order to eliminate a single variable we are forced to do this simultaneously for all the variables in whole block (see also Remark 2 in Section 5).

To motivate the rest of the paper, let us mention several applications of Theorem 2.1.

EXAMPLE 2.2. **(Pigeonhole principle).** The generalized pigeonhole principle PHP_n^m ($m \geq n+1$) says that if each of n holes may be occupied by only one of m pigeons then at least one pigeon must have no hole. The corresponding search problem is to find such a pigeon. More exactly, given an $n \times m$ $(0,1)$-matrix M with $m > n$ and exactly one 1 in each row, the problem is to find an all-0 column. In this case we have a hypergraph $\mathcal{F}$ with m edges, corresponding to columns, and n blocks, corresponding to rows. Since $|\mathcal{F}| = m > n$, this hypergraph is a large (k, b, λ, d)-design with $k = n$, $b = m$, $\lambda = 0$ and $d = 1$. Since $|\mathcal{F}| = m > n = k(d+1)/2$, we can apply (2.3), which yields the lower bound $2^{\Omega\left(n^2/(ml)\right)}$. Recall that $2^{\Omega(n^2/m)}$ is the best known lower bound for the minimal length of a resolution refutation proof of PHP_n^m [**13, 19, 8, 10**]. So, the reason why PHP_n^m is hard for resolution, seems to lie not in the weakness of the resolution rule itself, but rather in the impossibility to keep enough information about possible outcomes, using small (up to l) sets of clauses.

EXAMPLE 2.3. **(Affine planes).** Take an affine plane $\mathrm{AG}(2,q)$ of order q. Every point lies on $q+1$ lines, and there are $q(q+1)$ lines, each two of which intersect in at most one point. It is known (see [**14, 4**]) that every set of less than $2q-1$ points misses at least one line of $\mathrm{AG}(2,q)$. This result leads to the following *line search problem for* $\mathrm{AG}(2,q)$. We have $n = q(q+1)$ variables $x_1, \ldots, x_n$ corresponding to points, and n leaves, labeled by clauses $C_L = \bigvee_{i \in L} x_i$, corresponding to lines L. Given a set of at most $2(q-1)$ points, the problem is to find a line with no point in this set. By the result, mentioned above, this problem is well defined. Any semantic resolution proof for this problem solves the edge-search problem for the following design $\mathcal{F}$. Take any set $L_1, \ldots, L_q$ of q parallel (i.e. mutually disjoint) lines, and consider the hypergraph $\mathcal{F}$, the edges of which are all the remaining q^2 lines. Since every such line intersects each of the lines $L_1, \ldots, L_q$ in exactly one point, the hypergraph $\mathcal{F}$ is a (k, b, λ, d)-design with $k = b = d = q$ and $\lambda = 1$. To verify the largeness of this design, let B be a transversal of $L_1, \ldots, L_q$. If B would intersect all the lines in $\mathcal{F}$ then it would intersect *all* the lines of $\mathrm{AG}(2,q)$, which is impossible because $|B| = q < 2q-1$. Thus, every transversal B avoids at least one line of $\mathcal{F}$, and hence, $\mathcal{F}$ is large. Since $|\mathcal{F}| = q^2 > q(q+1)/2 = k(d+1)/2$, we we can apply (2.3), which yields the lower bound $2^{\Omega(q/l)} = \exp\left(\sqrt{|\mathcal{F}|}/l\right)$ on the size of any semantic resolution proof of fanin at most l, solving the edge-search problem for $\mathcal{F}$, and hence, for any such proof solving the line search problem for $\mathrm{AG}(2,q)$.

EXAMPLE 2.4. **(Projective planes).** Take a projective plane $\mathrm{PG}(2,q)$ of order q. It has the same number $n = q^2+q+1$ of lines and points; each line has $q+1$ points and every point lies in $q+1$ lines; any two lines share exactly one point. It is known (see [**5, 6**]) that any set of at most $q+\sqrt{q}$ points must either contain a line or must avoid a line. This result leads to the following *line search problem for* $\mathrm{PG}(2,q)$. We have $n = q^2+q+1$ variables $x_1, \ldots, x_n$ corresponding to points, and and $2n$ leaves, labeled by clauses $C_L^+ = \bigvee_{i \in L} x_i$ and $C_L^- = \bigvee_{i \in L} \neg x_i$. Given a set of at most $q+\sqrt{q}$ points, the problem is to find a line which lies entirely either in this set or in its complement. This problem reduces to the line search problem in affine planes. The idea is to use the well-known fact that deletion of any one line L_0 from $\mathrm{PG}(2,q)$ (together with all its points) gives us affine plane $\mathrm{AG}(2,q)$; the lines of this new plane are sets $L \setminus L_0$ where $L \neq L_0$ are lines of the projective plane. Let now G be a fanin-l semantic resolution proof solving the line search problem for $\mathrm{PG}(2,q)$. Fix an arbitrary line L_0 of $\mathrm{PG}(2,q)$ and set to 0 all the variables x_i with $i \in L_0$. This restriction kills (evaluates to 1) all negative leaves of G and deletes (i.e. evaluates to 0) exactly one variable from each positive leaf. The restriction $L_0 \mapsto 0$ corresponds to deletion of L_0 from $\mathrm{PG}(2,q)$, and hence, leads to $\mathrm{AG}(2,q)$. Thus, we obtain a proof which solves the line search problem for $\mathrm{AG}(2,q)$. As shown in the previous example, this proof (and hence the original proof G) must have at least $2^{\Omega(q/l)}$ clauses.

3. Combinatorics

The proof of Theorem 2.1 consists of three simple steps.

1. Firstly, we replace each clause in the original proof G by a *positive* clause so that the resulting proof G^+ still solves the original edge-search problem for $\mathcal{F}$.
2. The goal of the second 'killing large clauses' step is to show that, if the proof G^+ would have less than $2^{M/b}$ clauses, with M defined by (2.2), then it would be possible to set some t variables to constants so that all long clauses in G^+ are killed (i.e. are evaluated to 1). Conditions (2.1) are necessary to ensure that we do not kill too many primary leafs, i.e. that the whole search problem becomes not much easier after this restriction. Our restrictions are *deterministic*. Thus the whole argument avoids randomness.
3. The goal of the final 'forcing large clauses' step is to show that *any* proof solving the desired search problem, must have at least one long clause. Here we essentially use the fact that edges of our design are almost disjoint, i.e. that $|E \cap F| \leq \lambda$ for any $E \neq F \in \mathcal{F}$.

This implies that G could not have less than $2^{M/b}$, as desired.

All the combinatorics we need is accumulated in two easy lemmas: the 'killing' lemma and the 'forcing' lemma.

KILLING LEMMA . *Let $\mathcal{A}$ be a hypergraph over a set X, and $S_1, \ldots, S_k$ be a partition of X into sets of cardinality at most b. If $|\mathcal{A}| < \left(\frac{k}{k-t}\right)^{r/b}$ and each edge of $\mathcal{A}$ has more than r points then there is a partial transversal T of $S_1, \ldots, S_k$ such that $|T| \leq t+1$ and T intersects all the edges of $\mathcal{A}$.*

PROOF. Let $n = |X|$. We construct the set T via the following "greedy" procedure. Let $\mathcal{A}^1 = \mathcal{A}$ and $X^1 = X$. For each i, $1 \leq i \leq t$, include in T the element $x_i \in X^i$ which occurs in the largest number of sets of $\mathcal{A}^i$. Then remove from X^i all the points of that block, which contains x_i, to obtain X^{i+1}, and remove all the sets containing x_i from $\mathcal{A}^i$ to obtain $\mathcal{A}^{i+1}$. Sets deleted after $t+1$ steps intersect the set $\{x_1, \ldots, x_{t+1}\}$. Since $n \leq kb$, the number of remaining sets in $\mathcal{A}$

is bounded from above by $\alpha \cdot |\mathcal{A}|$ where

$$\alpha = \left(1-\frac{r}{n}\right)\left(1-\frac{r}{n-b}\right)\cdots\left(1-\frac{r}{n-bt}\right) \le \left(\frac{k}{k-t}\right)^{-r/b}.$$

Since $\mathcal{A}$ has less than α^{-1} sets, all the sets of $\mathcal{A}$ are already intersected by T, as desired. □

Let now $\mathcal{F}$ be a large (k,b,λ,d)-design with blocks $S_1,\ldots,S_k$. Fix an arbitrary partial transversal T of these blocks, and consider only those transversals which contain this particular transversal T. Let A be a set of points and $\mathcal{H} \subseteq \mathcal{F}$. We say that $\mathcal{H}$ is a *witness* of A if, for every transversal B containing T, we have that either $B \cap A \neq \emptyset$ or $B \cap E = \emptyset$ for at least one $E \in \mathcal{H}$ (or both). Put otherwise, every extension of T, intersecting all the edges of $\mathcal{H}$, must also intersect the set A. Given a set of points $A \subseteq X$, define its *weight*, $w_T(A)$ to be the minimum number of edges in a witness for A.

FORCING LEMMA . *Let T be a partial transversal, $t = |T|$, and let $A \subseteq X$ be a set of points of weight $s = w_T(A)$. Then*

$$|A| \ge \frac{s(k-t-s+1)^2}{k+\lambda(s-1)}. \tag{3.1}$$

PROOF. Lemma follows directly from the following two claims. Let $\mathcal{H} = \{E_1,\ldots,E_s\} \subseteq \mathcal{F}$ a minimal set of edges witnessing the weight of A.

CLAIM 3.1. The set A intersects every edge of $\mathcal{H}$ in at least $k-t-s+1$ points.

PROOF. Take an arbitrary edge $E \in \mathcal{H}$. Since $\mathcal{H}$ is minimal, there must be a transversal $B \supseteq T$ which intersects all the edges of $\mathcal{H}' = \mathcal{H} \setminus \{E\}$ but avoids both sets A and E. For each edge $E' \in \mathcal{H}'$ choose any one point from the intersection $B \cap E'$, and let I be the set of these choosed $\le |\mathcal{H}'| = s-1$ points. Let $\tilde{E}$ denote the set of all points in E, which belong to no of the blocks intersecting $I \cup T$. Since every edge E is a transversal, every block contains only one point of E, and hence, $|\tilde{E}| \ge |E| - |T| - |I| \ge k-t-s+1$. It remains therefore to prove that $A \supseteq \tilde{E}$ for every edge $E \in \mathcal{H}$.

To prove this, take an edge $E \in \mathcal{H}$ and an arbitrary point $x \in \tilde{E}$. Our goal is to show that x belongs to A. Let S be the (unique) block containing this point x. The fact that point x belongs to $\tilde{E}$ implies that this block S is disjoint from both T and I. Since B is a partial transversal and $B \cap \tilde{E} = \emptyset$, the block S intersects B in some other point $y \neq x$. Remove from B the point y and add the point x. The resulting set $(B \setminus \{y\}) \cup \{x\}$ intersects the edge E. Moreover, $B \setminus \{y\} \supseteq I \cup T$, because $y \in S$ and $S \cap (I \cup T) = \emptyset$. Therefore, the set $(B \setminus \{y\}) \cup \{x\}$ contains T and intersects all the remaining edges in $\mathcal{H}'$ (since I intersects them). Since $\mathcal{H}$ is a witness for A, we have that $A \cap ((B \setminus \{y\}) \cup \{x\}) \neq \emptyset$. This together with $A \cap B = \emptyset$, implies that $x \in A$. □

CLAIM 3.2. Let $\mathcal{A}$ be a hypergraph with s edges such that $u \le |E| \le v$ and $|E \cap F| \le \lambda$ for all $E \neq F \in \mathcal{A}$. Let $X = \bigcup_{E\in\mathcal{A}} E$ be the underlying set of points. Then $|X| \ge (u^2 s)/(v+(s-1)\lambda)$.

PROOF. The proof is a slight modification of a similar counting argument used by K. Corrádi [11] in the case when $u = v$. For a point $x \in X$, let $d(x)$ be the number of sets in $\mathcal{A}$ containing x. Then, for each edge E, $\sum_{x\in E} d(x) = \sum_{F\in\mathcal{A}} |E \cap F| \le v + (s-1)\lambda$. Summing over all edges we get

$$\sum_{E\in\mathcal{A}}\sum_{x\in E} d(x) = \sum_{x\in X} d(x)^2 \ge \frac{1}{|X|}\left(\sum_{x\in X} d(x)\right)^2 = \frac{1}{|X|}\left(\sum_{E\in\mathcal{A}} |E|\right)^2 \ge \frac{(us)^2}{|X|}.$$

Using the previous estimate we obtain $(us)^2 \le s \cdot |X|\,(v+(s-1)\lambda)$, which gives the desired lower bound on $|X|$. □

Now we can finish the proof of Forcing Lemma as follows. By Claim 3.1 there exist s subsets $\tilde{E}_i \subseteq E_i$ such that $A \supseteq \tilde{E}_1 \cup \cdots \cup \tilde{E}_s$ and $u \le |\tilde{E}_i| \le v$ with $u = k-t-s+1$ and $v = k$. Since sets E_i belong to the witness $\mathcal{H}$ (and hence, to the design $\mathcal{F}$), no two of them intersect in more than λ points, and Claim 3.2 yields the desired lower bound on $|\tilde{E}_1 \cup \cdots \cup \tilde{E}_s|$, and hence, the desired lower bound (3.1) on $|A|$. □

4. Proof of Theorem 2.1

Since G solves the edge-search problem for the design $\mathcal{F}$, it is possible to associate with each transversal B, a truth assignment u_B to the remaining variables so that $f(v_B, u_B) = 1$, where f is the conjunction of all secondary leaves and v_B is the incidence vector of B. Fix this injection $B \mapsto u_B$, and call a truth assignment *legal* if it has a form (v_B, u_B) for some transversal B.

Our first goal is to replace the clauses in G by clauses without negated main variables. The idea of this transformation is similar to that used by Buss [7] in case of the pigeonhole principle. For a point i, let $S(i) = S \setminus \{i\}$ where S is the (unique) block containing this point i. Replace every clause C of G by the clause C^+ which is obtained from C by replacing each negated literal $\neg x_i$ by the set of positive literals $\{x_j : j \in S(i)\}$. Since for any transversal B we have that $i \in B \iff B \cap S(i) = \emptyset$, it follows that $C^+(v_B, u_B) = C(v_B, u_B)$, and hence, the resulting proof G^+ still solves the edge-search problem for $\mathcal{F}$. Moreover, G^+ has at most $\ell = \text{size}(G)$ clauses. Let r be the smallest number for which

$$\ell < \left(\frac{k}{k-t}\right)^{r/b} \tag{4.1}$$

By Killing Lemma, there is a partial transversal T of size at most $t+1$ such that, setting to 1 all the variables x_i with $i \in T$, we kill off of all the clauses in G^+ with at least r main variables. Let G' be the resulting proof.

Since every primary leaf of G corresponds to an edge of $\mathcal{F}$ and each point belongs to no more than d edges of $\mathcal{F}$, at least $|\mathcal{F}| - dt$ of these leaves survive the restriction. We will use them to weight the clauses of G'. Namely, define the weight, $W(C)$ of a clause C to be the minimum number of primary leaves of G' whose conjunction implies C on all the legal truth assignments (v_B, u_B) with $B \supseteq T$. Note that primary leafs have weight 1, whereas secondary leaves have zero weight (they are satisfied by every legal assignment). On the other hand, the root must have weight larger than $\min\{|\mathcal{F}| - dt, k-t\}$, since we have at least $|\mathcal{F}| - dt$ primary leaves, for any $k-t$ of them we can find a transversal $B \supseteq T$ such that $B \setminus T$ intersects all of them (recall that $|B| = k$ and $|T| = t$). Since (by soundness) the weight of every clause is at most the sum of the weights of the (at most l) clauses from which it is derived, we can find a clause C such that $s \le W(C) \le ls$,

as long as ls does not exceed the weight of the root, which is ensured by (2.1). This clause has the form $C = \bigvee_{i \in A} x_i \vee C'$ where C' is the auxiliary part of C. Since primary leaves have only x-variables, the weight $W(C)$ of C is exactly the weight $w_T(A)$ of the corresponding set of points A. By Forcing Lemma this set has at least

$$M = \frac{s(k-t-ls+1)^2}{k+\lambda(s-1)}$$

points. Since all clauses with at least r main variables, are already killed, we have that $M < r$. Since r was minimal for which (4.1) holds, this means that

$$\text{size}(G) = \ell \geq \left(\frac{k}{k-l}\right)^{M/b},$$

which is $\geq 2^{M/b}$ since $t \geq k/2$, as desired. This completes the proof of Theorem 2.1. □

5. Concluding remarks

1. The input size of an edge-search problem for a hypergraph $\mathcal{F}$ is the number $|\mathcal{F}|$ of edges. It is interesting to compare the lower bounds which we obtain for searching problems, resulting from the generalized pigeonhole principle and from the line search problem in finite geometries. By Theorem 2.1, the general lower bound is exponential in $\Omega(k^2/b)$ if $\mathcal{F}$ is k-partite with block size b. Thus, in case of PHP_k^b, the bound is $\exp\left(k^2/|\mathcal{F}|\right)$, which is super-polynomial only if $|\mathcal{F}| = o\left(k^2/\log n\right)$, and it is still not known if it remains such for $|\mathcal{F}| \geq k^2$. (Recall that k is the number of holes and $|\mathcal{F}| = b$ is the number of pigeons). In this respect, the lower bound for $\text{AG}(2,q)$ is better: here we have $|\mathcal{F}| = k^2$ (with $k = q$) and the bound is $\exp\left(\sqrt{|\mathcal{F}|}\right)$.

2. The reason, why our argument (as well as previous arguments, based on Haken's "bottlenecks counting" idea [**13, 19, 8, 10**]) does not work for PHP_k^b with $b \geq k^2$, is that we *a priori* restrict our search domain to transversals only. This makes possible the transformation $G \mapsto G^+$ but binds our hands when trying to kill long clauses, since now our killing set T must be (partial) transversal. Note that without this last restriction, we could replace the bound $\left(\frac{k}{k-t}\right)^{r/b}$ in Killing Lemma by $\left(\frac{n}{n-t}\right)^r$, which does not depend on the block size b at all (!). The overall conclusion is that, in order to get lower bounds for PHP_k^b with $b \geq k^2$, one should learn more on how to force large clauses which are not assumed be positive. Quite recently, Razborov, Wigderson and Yao [**16**] have made an interesting attempt to overcome this k^2 barrier. Using a novel technique they where able to prove exponential lower bounds (for arbitrarily large b!) on the size of some restricted versions of regular resolution proofs for PHP_k^b.

3. In this paper we have shown that the combinatorics of semantic Resolution is captured by two simple "killing" and "forcing" lemmas. Next logical step could be to understand the combinatorics of *cutting planes* proofs. All the known superpolynomial lower bounds for the length of such proofs follow from the corresponding lower bounds on the size of monotone Boolean circuits via appropriate interpolation theorems (see, e.g., [**15**] for a survey). Thus, these bounds capture the weakness of corresponding circuits rather than the weakness of cutting planes themselves. Moreover, this approach fails in the situations where the corresponding problems (like all three examples in Section 2) *have* small circuits. To get more insight into the combinatorial nature of cutting planes proofs, it would be interesting to understand the cutting plane complexity of blocking principles for finite geometries. These geometries have more structure then the pigeonhole principle, and the corresponding principles have very natural formulation in terms of linear inequalities. The Jamison-Brower-Schrijev's theorem [**14, 4**] for $\text{AG}(2,q)$ is given by the system of $2n+1$ inequalities:

$$\sum_{i \in L_j} x_i \geq 1, \quad \sum_{i=1}^{n} x_i \leq 2(q-1), \quad 0 \leq x_i \leq 1 \qquad i,j = 1,\ldots,n$$

Bruen's theorem [**5**] for $\text{PG}(2,q)$ also can be stated as a system of $3n+1$ inequalities:

$$1 \leq \sum_{i \in L_j} x_i \leq q-1, \quad \sum_{i=1}^{n} x_i \leq q+\sqrt{q}, \quad 0 \leq x_i \leq 1 \qquad i,j = 1,\ldots,n.$$

What is the cutting plane complexity of these systems? The "quadratic counting" trick used in Bruen's proof makes plausible the conjecture that this system does *not* have a short cutting planes proof, unless we allow quadratic inequalities and/or multiplication of two inequalities. Both answers - a short cutting planes proof of Bruen's theorem or the absence of such proof - would be interesting.

Acknowledgment

I thank Alexander Razborov for turning my attention to resolution proofs and very interesting discussions. I also thank the anonymous referee for several helpful suggestions concerning the presentation; the name "Hitting Set Principle" was one of these suggestions.

References

[1] P. Beame and T. Pitassi, *Simplified and improved resolution lower bounds*, Proc. 37th IEEE Sympos. on Foundations of Computer Science, 1996.
[2] A. Blacke, *Canonical expressions in Boolean algebra*, PhD thesis, University of Chicago, 1937.
[3] A. Blokhuis, *On the size of a blocking set in PG(2,p)*, Combinatorica **14** (1994), 111–114.
[4] A.E. Brower and A. Schrijev, *The blocking number of an affine space*, J. Comb. Theory (A) **24** (1978), 251–253.
[5] A. A. Bruen, *Baer subplanes and blocking sets*, Bull. Amer. Math. Soc. **76** (1970), 342-344.
[6] ———, *Blocking sets in finite projective planes*, SIAM J. Appl. Math. **21** (1971), 380–392.
[7] S. Buss, *Polynomial size proofs of the propositional pigeonhole principle*, J. Symbolic Logic **52** (1987), 916–927.
[8] S. Buss and G. Turán, *Resolution proofs of generalized pigeonhole principles*, Theor. Comput. Sci. **62** (1988), 311–317.
[9] V. Chvátal and E. Szemerédi, *Many hard examples for resolution*, Journal of the ACM **35**:4 (1988), 759–768.
[10] S. Cook and T. Pitassi, *A feasibly constructive lower bound for resolution proofs*, Inform. Process. Lett. **34** (1990), 81–85.
[11] K. Corrádi, *Problem at the Schweitzer competition*, Mat. Lapok **20** (1969), 159–162.
[12] M. Davis and H. Putnam, *A computing procedure for quantification theory*, Journal of the ACM **7**(3) (1960), 210–215.
[13] A. Haken, *The intractability of resolution*, Theor. Comput. Sci. **39** (1985), 297–308.
[14] R. Jamison, *Covering finite fields with cosets of subspaces*, J. Comb. Theory (A) **22** (1977), 253–266.

[15] A. Razborov, *Lower bounds for propositional proofs and independence results in bounded arithmetic*, Proc. 23rd Int. Colloq. Automata, Languages and Programming, ICALP'96 (Paderborn, Germany), 1996.
[16] A. Razborov, A. Wigderson and A. Yao, *Read-once branching programs, rectangular proofs of the pigeonhole principle and the transversal calculus*, manuscript, 1996.
[17] J. A. Robinson, *A machine-oriented logic based on the resolution principle*, Journal of the ACM **12**:1 (1965), 23–41.
[18] G. C. Tseitin, *On the complexity of derivations in propositional calculus*, Studies in mathematics and mathematical logic, Part II, ed. A. O. Slisenko, 1968, pp. 115–125.
[19] A. Urquhart, *Hard examples for resolution*, Journal of the ACM **34**:1 (1987), 209–219.

Department of Mathematical Logic, Institute of Mathematics, Akademijos 4, Vilnius 2000, Lithuania
Current address: Department of Computer Science, University of Trier, D-54286, Trier, Germany
E-mail address: jukna@ti.uni-trier.de

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume **39**, 1998

Bounded Arithmetic: Comparison of Buss' Witnessing Method and Sieg's Herbrand Analysis.

Barbara Kauffmann

Abstract. The Σ_1^b-*definable functions* of theory S_2^i of bounded arithmetic are the $\Box_i^p$-functions. This result was obtained by Buss in [1]. Later Sieg proved the same result in [8] using *Herbrand analysis* instead of Buss' *witnessing method*. Buss' witnessing method and Sieg's Herbrand analysis provide a method to canonically extract terms from derivations. These terms (represent specific algorithms that) compute Σ_1^b-definable functions whose totality is provable in S_2^i. We say that such functions are the Σ_1^b-*definable functions* of our theory. If we restrict our attention to the polynomial time computable functions, which are Σ_1^b-definable in S_2^1, the problem of finding the *least complex* algorithm to compute such functions becomes an interesting problem from a computational complexity point of view.

In this paper I compare Buss' and Sieg's methods for S_2^1. Both methods have been adapted to a particular system, a modified Tait-style sequent calculus. I show that if all inessential parts are stripped away from Buss' method, then it behaves in exactly the same way as (my modified version of) Sieg's method. Finally I will analyze different strategies that can be used to extract terms (denoting algorithms) that compute polynomial functions and compare them.

1. Introduction.

Bounded arithmetic was proposed by Parikh in 1971 [6] for the first time. Parikh was interested in problems related to the length of proofs. His system, called PB, is what we generally know under the name of $I\Delta_0$. His work was further developed by Paris and Wilkie who built a system called $I\Delta_0$-Ω_1, and, successively, Dimitracopoulos, Kaye and Woods. Cook in 1975 [2] constructed an equational theory PV, a subsystem of bounded arithmetic, proving a relation between this system and propositional logic. Finally Buss in [1] introduced a system of bounded arithmetic in order to treat problems related to the question $P =^? NP$. He formulated bounded arithmetic S_2 as a proof theoretical setting to show that fragments of S_2

1991 *Mathematics Subject Classification.* Primary 03F30: Secondary 03F05.

can express functions in the polynomial hierarchy[1]. The same result was later obtained by Sieg [8] using an alternative proof-theoretic method to Buss' *witnessing method*. In [8] Sieg applied his method, *Herbrand analysis*, to fragments of arithmetic and bounded arithmetic among others. Both methods investigate the relation between the complexity of the quantified formulas of a theory and the complexity of the relations that such formulas define, measured in terms of the polynomial hierarchy. The strength of a theory is therefore measured in terms of the functions that are Σ_1^b-definable in it.

In this work I introduce and compare Buss' and Sieg's methods for a particular fragment of bounded arithmetic, S_2^1. Buss and Sieg use two different calculi: Gentzen's original sequent calculus and Tait's version of the sequent calculus. In order to make the procedures uniform and the comparison of the *computational complexities* of the terms[2] extracted by Buss' and Sieg's methods possible, I use a modified Tait style sequent calculus. Buss' and Sieg's proofs are therefore transformed accordingly. Consider now the following result, a restriction of Buss' and Sieg's theorems ([Buss 86], [Sieg 91]).

THEOREM 1.1. *Let δ be an S_2^1-derivation of $\forall \vec{x}\, \exists! y\, \theta(\vec{x}, y)$[3] θ open. Then there is a function $f \in \Box_1^p$ so that for every $\vec{n} \in N$, $N \models \theta(\vec{n}, f(\vec{n}))$.*

The derivation δ contains information about quantifiable properties such as term growth, type of induction, nestings of induction, conjunctions and cuts. The extraction of the term that describes f uses such information, but also depends on the extraction procedure itself. This consists of two main steps: 1. *normalization* of derivation δ by means of a (partial) cut-elimination, 2. construction of the term (representing the algorithm) that computes f (the actual extraction procedure). Such a term, assigned to y in the conclusion of the I-normal derivation δ' obtained from δ, is built inductively by *updating* terms extracted at previous steps of δ'. The *updating* depends on the structure of the derivation itself and on the particular way of *storing* information chosen by each method.

The witnessing method assigns a *witness* to each line (*sequent*) of the derivation. This witness codifies a sequence of values which are either terms assigned to the *existential variables* occurring in the sequent or 0s that stand

[1] S_2 is a conservative extension of system $I\Delta_0$-Ω_1.

[2] Both methods provide a canonical way of extracting terms from derivations, in our case S_2^1-derivation. Such terms denote polynomial time computable functions. From the terms extracted it is possible to build algorithms computing those functions. We will compare the *computational complexities* (or *running times*) of the algorithms associated to the terms extracted. Nevertheless, for simplicity in this paper we will talk about computational complexity of the *terms* extracted.

[3] The intended meaning of $\forall \vec{x}\, \exists! y\, \theta(\vec{x}, y)$: $\forall \vec{x}\, \exists y\, (\theta(\vec{x}, y) \wedge \forall \vec{x}\, \forall y\, \forall z\, \theta(\vec{x}, y) \wedge \theta(\vec{x}, z) \rightarrow y = z)$.

for empty sequences assigned to *non-quantified formulas* in the sequent. Moving from one line of the derivation to the next a new witness is built by updating the previous one (or ones, in the case of inferences with two premises). In the case of Sieg's method, at each step of the derivation a term (*Skolem term*) is assigned to each existential variable which results from an *update* of the term (or terms, in the case of two-premise inferences) previously assigned. Notice that Sieg's method does not assign any term to quantifier-free formulas.

The different way of storing information influences the computational complexity of the term extracted. I will prove that the term extracted by Herbrand analysis has computational complexity *at most* equal to the complexity of the term extracted by the witnessing method (theorem 4.1, corollary 4.2 and corollary 4.3). I will eventually explore and compare alternative extraction strategies. The way in which sequences of formulas in a derivation are treated influences the computational complexity of the final term extracted from that derivation.

The relevance of this work lies in the fact that the Σ_1^b-definable functions of S_2^1 are the polynomial time computable functions. The problem of mechanically extracting from the derivation the term computing f efficiently becomes therefore an important goal.

As one can notice from the formulation of theorem 1.1, θ is open. I chose the simplest possible case to compare Buss' and Sieg's methods. The results contained in this paper can be easily extended to the case where θ is any Σ_1^b-formula ([3]).

The present work is related to my Ph.D. dissertation and has been influenced by early conversations with Wilfried Sieg who suggested to me to explore this topic. I would like to thank Toniann Pitassi, Wilfried Sieg and Jeremy Avigad whose precious suggestions and helpful insights made this paper possible.

2. Proof System.

We introduce the Tait-style sequent calculus modified to derive sequences of formulas. The formulas are built in the language of S_2^1, containing logical symbols, $\wedge$, $\vee$, $\neg$, $\forall$ and $\exists$, where $\neg$ is the symbol introduced as primitive for negation of atomic formulas. For simplicity we will use sometimes the extended language containing negation of more complex formulas. Here the negation of a non-atomic formula A is a certain formula A', in the restricted language, corresponding to A with negation pushed towards its atomic parts and with boolean connectives and quantifiers changed according to the De Morgan's rules and the rules for negation of quantifiers. Non-logical function symbols of the language of S_2^1 are 0, $S(x)$, $+$, $\cdot$, $|x|$, $\lfloor \frac{1}{2}x \rfloor$, $x\#y$, $x\lceil n$ and the predicate symbol $\leq$. Here $|x|$ denotes the length of the binary representation of x (i.e. $\lceil (\log x) + 1 \rceil$, with $|0| = 0$). $\lfloor \frac{1}{2}x \rfloor$ denotes the greatest integer less

than or equal to $x/2$. $x\#y$ is equal to $2^{|x|\cdot|y|}$. $x\lceil n$ denotes the first n bits of x in binary notation. It corresponds to Buss' $MSP(x,|x| \dot{-} n)$ [1], and Ferreira's $x|_z$ where z is a word of length n [5].

Bounded formulas are treated as a special case of quantified formulas. They contain an internal bound for quantified variables[4]. Moreover let us call *purely existential* formulas of the form $\exists\vec{y}\,\phi(\vec{a},\vec{y})$ or $\exists\vec{y}\,(\,\vec{y} \le \vec{t}(\vec{a}) \wedge \phi(\vec{a},\vec{y})\,)$ where ϕ is open. *Purely bounded existential* (or purely Σ^b_1) are formulas of the form $\exists\vec{y}\,(\,\vec{y} \le \vec{t}(\vec{a}) \wedge \phi(\vec{a},\vec{y})\,)$, with ϕ open. We will consider formulas with just one existential (or bounded existential) variable, as multiple occurrences of (bounded) existential quantifiers are reducible to just one occurrence ($\exists y\,\exists z\,\phi(\vec{a},y,z) \equiv \exists w\,\phi(\vec{a},(w)_1,(w)_2)$). As we already said the following results can be extended to the case of arbitrary Σ^b_1-formulas (i.e., formulas containing sharply bounded quantifiers).

The calculus.

- (LA) $\quad \Gamma,\ \phi,\ \neg\phi$ $\quad$ ϕ is atomic;
- $(\wedge)$ $\quad \dfrac{\Gamma,\ \phi_0 \quad \Gamma,\ \phi_1}{\Gamma,\ \phi_0 \wedge \phi_1}$;
- $(\vee)$ $\quad \dfrac{\Gamma,\ \phi_i}{\Gamma,\ \phi_0 \vee \phi_1}$ $\quad i = 0,1$;
- (cut) $\quad \dfrac{\Gamma,\ \phi \quad \Gamma,\ \neg\phi}{\Gamma}$

  $\neg\phi$ is ϕ with negation pushed inside towards the atomic part of ϕ;
- $(\forall)$ $\quad \dfrac{\Gamma,\ \phi(a)}{\Gamma,\ \forall x\,\phi(x)}$ $\quad a$ not free in Γ;
- $(\exists)$ $\quad \dfrac{\Gamma,\ \phi(t)}{\Gamma,\ \exists x\,\phi(x)}$;

plus the following structural rules:

- (C) $\quad \dfrac{\Gamma,\ \phi,\ \phi}{\Gamma,\ \phi}$ $\quad$ contraction;
- (W) $\quad \dfrac{\Gamma}{\Gamma,\ \phi}$ $\quad$ weakening;
- (E) $\quad \dfrac{\Gamma,\ \phi_0,\ \phi_1}{\Gamma,\ \phi_1,\ \phi_0}$ $\quad$ exchange.

The theory S^1_2 consists of the axioms $(BASIC)$ introduced by Buss in [1], together with the axioms for the new symbol $\lceil$ and Σ^b_1-induction. The axioms for the new function symbol $\lceil$ are

[4]We will not extend the language to allow symbols for bounded quantifiers. Nevertheless we will make use of such symbols when we want to stress the difference between a Σ^b_1-formula and an $\exists\Delta^b_0$-formula. In such cases we will write $\exists\vec{y} \le \vec{t}(\vec{a})\,\phi(\vec{a},\vec{y})$ ($\phi \in \Delta^b_0$) instead of $\exists\vec{y}(\vec{y} \le \vec{t}(\vec{a}) \wedge \phi(\vec{a},\vec{y}))$.

1. $y\lceil 0 = 0$,
2. $|y| \le z \rightarrow y\lceil z = y$,
3. $z < |y| \rightarrow |y\lceil z| = z$,
4. $z < |y| \rightarrow \lfloor \frac{1}{2}(y\lceil(z+1))\rfloor = y\lceil z$.

The induction principle is formulated as follows for just Σ^b_1-formulas ϕ

$$\neg(\phi(\vec{a},0) \wedge \forall y\,\neg\phi(\vec{a},\lfloor\tfrac{1}{2}y\rfloor)) \vee \phi(\vec{a},y))) \vee \forall y\phi(\vec{a},y).$$

Notice that the induction axiom is equivalent to the induction rule. Here we use indifferently the above axiom or the corresponding rule:

$$\frac{\Gamma(\vec{a}),\ \phi(\vec{a},0) \qquad \Gamma(\vec{a}),\ \neg\phi(\vec{a},\lfloor\tfrac{1}{2}b\rfloor),\ \phi(\vec{a},b)}{\Gamma(\vec{a}),\ \phi(\vec{a},t(\vec{a}))},$$

where b is a new variable (*eigenvariable*) not occurring in Γ.

3. The two procedures.

Consider theorem 1.1 introduced in section 1. Buss' witnessing method and Sieg's Herbrand analysis use different procedures to extract the term (denoting the algorithm) to compute f from δ. These procedures share a similar start but diverge very soon.

3.1. Starting point.

- **cut-elimination**. The derivation δ is first transformed into a derivation δ' with cuts *only* on substitutional instances of atoms, axioms and formulas that are principal in an induction inference. So δ' may contain Σ^0_1 and Σ^b_1-formulas. We call δ' an I-normal derivation[5] following Sieg's [8][6]. At this point $\forall$-inversion can be applied to δ' without affecting the length of the derivation.
- **Bootstrapping**. We want to enable the theory to talk about new functions. Buss *bootstraps* S^1_2 by adding defining axioms for some polynomial functions, in order to allow the theory to recognize, build and decompose sequences and extract elements from them among other functions. The bootstrapping of S^1_2 corresponds to a conservative extension of S^1_2.

  Sieg proposes a conservative extension of S^1_2, that 'explicitly' extends the language[7] of S^1_2. That theory, called $S^1_2(\Box^p_1)$, is defined on the language of S^1_2 augmented with additional function symbols for all polynomial time computable functions, and consists of an extension

[5]An I-normal derivation is equivalent to what Buss calls *free-cut-free* derivation.

[6]Look at Sieg's paper [8] for the proof.

[7]Buss did not explicitly extend the language of S^1_2. He used instead metamathematical abbreviations to represent the new functions introduced by their defining axioms.

of the axioms of S^1_2 obtained by adding the defining axioms for all polynomial computable functions.

The bootstrapping of S^1_2 can be extended to all $\Box^p_1$-functions and the extension of the language rendered explicit. Thus we may assume, without loss of generality, that both methods deal with the conservative extension $S^1_2(\Box^p_1)$ of S^1_2.

It is important to notice that this technically trivial step was not taken by Buss; thus it was necessary to bound all occurrences of $\exists\Delta^b_0$-formulas in order to insure that the term extracted to "witness" the existential variable in $\exists y\,\theta(\vec{a}, y)$ is still a $\Box^p_1$-function. Such bound can be provided by applying Parikh's theorem to the original derivation δ' of $\exists y\,\theta(\vec{a}, y)$. So as we can see the difference between extension of the language and construction of new functions from existing ones is not trivial. Here we assume an explicit extension of the language of S^1_2 by adding terms for all $\Box^p_1$-functions.

After these initial steps the two methods diverge. The differences arise in the procedures used to extract the term describing the polynomial function f.

3.2. Buss' method.

To prove theorem 1.1 Buss' method applies Parikh's theorem to the I-normal derivation δ' of $\forall\vec{x}\,\exists y\,\phi(\vec{x}, y)$, providing a bound for all existential variables of $\exists\Delta^0_1$-formulas occurring in δ'. Derivation δ' will therefore be transformed into an I-normal derivation of (purely existential) Σ^b_1-formulas, δ^P.

THEOREM 3.1. *Consider* $S^1_2(\Box^p_1) \vdash \forall\vec{x}\,\exists y\,\phi(\vec{x}, y)$, *where* ϕ *is open. Then there exists a term* $s(\vec{x})$ *in the language of* $S^1_2(\Box^p_1)$, *so that* $S^1_2(\Box^p_1) \vdash \forall\vec{x}\,\exists y \leq s(\vec{x})\ \phi(\vec{x}, y)$.

Once Parikh's theorem is applied and the derivation δ^P of purely Σ^b_1-formulas has been obtained from the I-normal derivation δ', Buss' witnessing theorem can be applied.

I will show that Parikh's result is not necessary. Parikh's theorem is needed by Buss' method in the proof of the witnessing theorem. In the original presentation of Buss the witness for the conclusion of a Σ^b_1-induction inference is found by using limited recursion on notation. In order to insure that such a witness is still a polynomial time computable function we need to prove that it has a bound. The extraction of such witness requires that all side formulas contain bounded variables. In our case we can avoid such bounds because we have an explicit expansion of the language, and we use a different strategy.

Let us now introduce the following definition that will be used to prove theorem 3.3.

3.2. Definition. Let ψ be a purely Σ^b_1-formula and $\vec{a}$ a vector of free variables which includes all free-variables of ψ. Define then a formula $Witness_\psi$ by induction on the complexity of ψ as follows:

- If ψ is atomic, then
$$Witness_{\psi(\vec{a})}(w, \vec{a}) \;\Leftrightarrow\; \psi(\vec{a}),$$
where w can be anything, in particular 0, which stands for the empty sequence.

- If $\psi(\vec{a})$ is $\phi_1(\vec{a})@\phi_2(\vec{a})$, where @ is either $\wedge$ or $\vee$, then
$$\begin{aligned} Witness_{\psi(\vec{a})}(w, \vec{a}) \;&\Leftrightarrow\; Seq(w) \wedge Len(w) = 2 \wedge \\ Witness_{\phi_1(\vec{a})}(\beta(1, w), \vec{a}) \;&@\; Witness_{\phi_2(\vec{a})}(\beta(2, w), \vec{a}), \end{aligned}$$
where Seq, Len, β are polynomial time computable functions and thus available in $S^1_2(\Box^p_1)$.

- If $\psi(\vec{a})$ is $\exists y \leq s(\vec{a})\ \phi(\vec{a}, y)$, then
$$\begin{aligned} Witness_{\psi(\vec{a})}(w, \vec{a}) \;&\Leftrightarrow\; Seq(w) \wedge Len(w) = 2 \wedge \\ \beta(1, w) \leq s(\vec{a}) \;&\wedge\; Witness_{\phi(\vec{a}, y)}(\beta(2, w), \vec{a}, \beta(1, w)), \end{aligned}$$
where $\beta(2, w)$ is the witness of $\phi(\vec{a}, \beta(1, w))$.

- If $\psi(\vec{a})$ is $\exists y\,\phi(\vec{a}, y)$, then
$$\begin{aligned} Witness_{\psi(\vec{a})}(w, \vec{a}) \;\Leftrightarrow\; & Seq(w) \wedge Len(w) = 2 \wedge \\ & Witness_{\phi(\vec{a}, y)}(\beta(2, w), \vec{a}, \beta(1, w)), \end{aligned}$$
where $\beta(2, w)$ is the witness of $\phi(\vec{a}, \beta(1, w))$. This step, not present in the original definition of Buss, was added to introduce witnesses for (purely) Σ^0_1-formulas.

- Recall that if $\psi(\vec{a})$ is $\neg\phi(\vec{a})$ and ϕ is atomic then $Witness_{\psi(\vec{a})}(0, \vec{a}) \Leftrightarrow \neg\phi(\vec{a})$, otherwise we will apply De Morgan's laws or the rules for quantifiers to ϕ.

Finally consider a sequence Δ of open or purely existential formulas, $\{\psi_1, \ldots, \psi_n\}$, with free variables $\vec{a}$. The witness of Δ corresponds to a sequence of witnesses for the n formulas in Δ:

$$Witness_{\bigvee\Delta}(w, \vec{a}) \;\Leftrightarrow\; Seq(w) \wedge Len(w) = n \wedge \bigvee_{i=1}^{n} Witness_{\psi_i}(\beta(i, w), \vec{a}).$$

Alternatively we will write w_{ψ_i} for $\beta(i, w)$ $(i = 1, \ldots, n)$.

The above definition does not provide a method of finding the witness w of a given formula ψ, but sets the conditions under which w can be considered as the witness of ψ. Such a method will be offered by Buss' witnessing theorem. This has been modified to fit our Tait-style sequent calculus (theorem 3.3). We assume that before applying the witnessing theorem to a given I-normal derivation δ of Σ^b_1-formulas $\forall$-inversion is applied to every occurrence of universal formulas.

THEOREM 3.3. *Let Δ be a sequence of purely existential formulas provable in $S^1_2(\Box^p_1)$ with an I-normal derivation δ. Then there is a function symbol g denoting a polynomial time computable function so that*

$$S^1_2(\Box^p_1) \vdash Witness_{\bigvee\Delta}(g(\vec{a}), \vec{a}).$$

PROOF. The proof is by induction on δ. We will consider only a few significant cases.

1. $(\wedge)$-inference.

$$\frac{\Delta, A_0 \qquad \Delta, A_1}{\Delta, A_0 \wedge A_1}$$

By induction hypothesis there is a $\Box^p_1$-function $g_j(\vec{a})$ that witnesses Δ, A_j, $(j = 0, 1)$ so that $S^1_2(\Box^p_1) \vdash Witness_{\bigvee\Delta\vee A_j}(g_j(\vec{a}), \vec{a})$. The witness for the conclusion $(\Delta, A_0 \wedge A_1)$ is a polynomial time computable function $g(\vec{a})$ that assigns a witness to Δ by choosing between the witnesses assigned to it in the premises. So

$$S^1_2(\Box^p_1) \vdash Witness_{\bigvee\Delta\vee A_0\wedge A_1}(g(\vec{a}), \vec{a}),$$

where $g(\vec{a}) := choice_\Delta(g_0(\vec{a}), g_1(\vec{a}))$, which means 'check $Witness_{\bigvee\Delta}(g_0(\vec{a})_\Delta, \vec{a})$. If $g_0(\vec{a})$ witnesses Δ then $g(\vec{a})$ is $g_0(\vec{a})$. Otherwise $g(\vec{a}) := g_1(\vec{a})$'.

2. $(\exists)$-inference.

$$\frac{\Delta, \theta(\vec{a}, t(\vec{a}))}{\Delta, \exists y\, \theta(\vec{a}, y)},$$

where θ is open.

By induction hypothesis the witness of the premise is a $\Box^p_1$-function $g'(\vec{a})$ so that $S^1_2(\Box^p_1) \vdash Witness_{\bigvee\Delta\vee\theta}(g'(\vec{a}), \vec{a}, t(\vec{a}))$. Then the witness for the conclusion is a $\Box^p_1$-function $g(\vec{a})$ that codes a sequence of witnesses whose last element, $\beta(n+1, g(\vec{a})) = \langle t(\vec{a}), 0\rangle$, is the witness for $\exists y\theta(\vec{a}, y)$ and, for $i = 0, \ldots, n$, $\beta(i, g(\vec{a})) := \beta(i, g'(\vec{a}))$. Thus

$$S^1_2(\Box^p_1) \vdash Witness_{\bigvee\Delta\vee\exists y\theta}(g(\vec{a}), \vec{a}).$$

3. $(PIND)$-inference:
The principal formula of induction can be either open or (purely) Σ^b_1. We consider the second case only.

$$\frac{\Delta, \varphi(\vec{a}, 0) \qquad \Delta, \neg\varphi(\vec{a}, \lfloor\frac{1}{2}b\rfloor), \varphi(\vec{a}, b)}{\Delta, \varphi(\vec{a}, t(\vec{a}))},$$

where $\varphi(\vec{a}, x)$ is $\exists y\, \theta(\vec{a}, y, x)$, $\theta(\vec{a}, y, x)$ is $y \leq s(\vec{a}) \wedge \theta'(\vec{a}, y, x)$ and θ' is open. Notice that $\neg\varphi(\vec{a}, \lfloor\frac{1}{2}b\rfloor)$ is $\neg\theta(\vec{a}, z, \lfloor\frac{1}{2}b\rfloor)$, where z is a new parameter not occurring in Δ, φ. So the right premise is $\Delta, \neg\theta(\vec{a}, z, \lfloor\frac{1}{2}b\rfloor), \exists y\, \theta(\vec{a}, y, b)$.

By induction hypothesis there are $\Box^p_1$-functions $g_0(\vec{a})$ and $g_1(\vec{a}, z, b)$ so that

$$S^1_2(\Box^p_1) \vdash Witness_{\bigvee\Delta\vee\varphi}(g_0(\vec{a}), \vec{a}, 0) \tag{1}$$

and

$$S^1_2(\Box^p_1) \vdash Witness_{\bigvee\Delta\vee\neg\theta\vee\varphi}(g_1(\vec{a}, z, b), \vec{a}, z, b). \tag{2}$$

Define function g^* by recursion on notation as follows:

$$\begin{aligned} g^*(\vec{a}, 0) &= g_0(\vec{a})_\varphi \\ g^*(\vec{a}, n) &= g_1(\vec{a}, g^*(\vec{a}, n \dot{-} 1), t\lceil n)_\varphi. \end{aligned}$$

Clearly g^* is polynomial time computable, since for every x $g^*(\vec{a}, x) \leq s(\vec{a})$.

Consider now the side formulas Δ. If $g_0(\vec{a})$ is a witness of Δ, then keep it, otherwise look for the smallest length $k \leq |t(\vec{a})|$ so that $Witness_\Delta(g_1(\vec{a}, g^*(\vec{a}, k \dot{-} 1), t(\vec{a})\lceil k)_\Delta, \vec{a}, t(\vec{a})\lceil k)$. In other words, we define a function h as the choice between g_0 and g_1

$$h(\vec{a}) = \begin{cases} g_0(\vec{a})_\Delta & \text{if } Witness_\Delta(g_0(\vec{a})_\Delta, \vec{a}, 0) \\ g_1(\vec{a}, g^*(\vec{a}, k \dot{-} 1), t(\vec{a})\lceil k)_\Delta & \text{otherwise} \end{cases},$$

where k is found by sharply bounded search. Since g_0 and g_1 are polynomial time computable functions so it will be f.

Consider then (1). If $\neg Witness_\varphi(g_0(\vec{a})_\Delta, \vec{a})$ but $Witness_\varphi(g_0(\vec{a})_\varphi, \vec{a}, 0)$, then we can assign any witness to Δ in particular $g_1(\vec{a}, g^*(\vec{a}, 0), 0)_\Delta$.

Let $\hat{g}(\vec{a}, y)$ be $\langle g_1(\vec{a}, g^*(\vec{a}, y \dot{-} 1), t(\vec{a})\lceil y)_\Delta, g^*(\vec{a}, y)\rangle$. Then

$$S^1_2(\Box^p_1) \vdash Witness_{\bigvee\Delta\vee\varphi}(\hat{g}(\vec{a}, 0), \vec{a}, 0) \tag{3}$$

and

$$\begin{aligned} S^1_2(\Box^p_1) \vdash\ & \neg Witness_{\bigvee\Delta\vee\varphi}(\hat{g}(\vec{a}, m \dot{-} 1), \vec{a}, t(\vec{a})\lceil m \dot{-} 1), \\ & Witness_{\bigvee\Delta\vee\varphi}(\hat{g}(\vec{a}, m), \vec{a}, t(\vec{a})\lceil m), \end{aligned} \tag{4}$$

which easily follows from (2).

Apply then induction on (3) and (4) to obtain

$$S^1_2(\Box^p_1) \vdash Witness_{\bigvee\Delta\vee\varphi}(\hat{g}(\vec{a}, |t(\vec{a})|), \vec{a}, t(\vec{a})).$$

Finally $g(\vec{a}) = \langle h(\vec{a}), g^*(\vec{a}, t(\vec{a}))\rangle$.

□

As usual write Δ as $\Delta[\exists y_1\phi_1(\vec{a}, y_1), \ldots, \exists y_n\phi_n(\vec{a}, y_n)]$. Define $\Delta[g(\vec{a})]$ as $\Delta[\phi_1(\vec{a}, \beta(1, g(\vec{a})_{\exists y_1\phi_1}), \ldots, \phi_n(\vec{a}, \beta(1, g(\vec{a})_{\exists y_n\phi_n})]$. The following result shows that a derivation of $Witness_\Delta(g(\vec{a}), \vec{a})$ in $S^1_2(\Box^p_1)$ is in fact a derivation of $\Delta[g(\vec{a})]$ in the theory with open induction.

COROLLARY 3.4. *Let δ be an I-normal derivation of $Witness_\Delta(g(\vec{a}),\vec{a})$ in $S^1_2(\Box^p_1)$ where Δ is a sequence of purely existential formulas. Then there is an I-normal $(\mathbf{QF}(\Box^p_1) - PIND)$-derivation δ' of $\Delta[g(\vec{a})]$.*

To prove the above result we use the definition of witnessing formula showing, by induction on δ, that for each sequence of formulas Γ occurring in δ, $Witness_\Gamma(g(\vec{a}),\vec{a})$ is in fact $\Gamma[g(\vec{a})]$. But $\Gamma[g(\vec{a})]$ is a sequence of open formulas. Thus any Σ^b_1-induction inference of δ is now transformed into an open induction inference.

3.3. Herbrand analysis.

In [8] Sieg provides a clear separation between the reduction of the original derivation to a derivation with open induction only and proved in the Herbrand theory $(\mathbf{QF}(\Box^p_1) - \mathrm{PIND})$, and the application of $\exists$-inversion on a specific purely existential formula, in the $\exists$-inversion procedure. In this way the 'obvious' steps of computation (mainly, the $\exists$-inversion steps) are distinguished from the non-trivial ones (the reduction of the complexity of induction via suitable recursion). Here I combine Sieg's results in a single procedure, as shown by theorem 3.5, to facilitate the comparison with Buss' witnessing theorem. It is indeed the $\exists$-inversion procedure, and not just the original $\exists$-inversion theorem of Sieg, that corresponds to Buss' witnessing theorem [8].

Consider a sequence containing purely existential formulas Δ. As usual we will write $\Delta[\exists y_1\,\phi_1(\vec{a},y_1),\ldots,\exists y_n\,\phi_n(\vec{a},y_n)]$, with ϕ_i open $(i=1,\ldots,n)$, to stress all the occurrences of purely existential formulas in Δ. We will write $\Delta[t(\vec{a})]$, to indicate the sequence of formulas $\Delta[\phi_1(\vec{a},t(\vec{a})_1),\ldots,\phi_n(\vec{a},t(\vec{a})_n)]$, where $t(\vec{a})_i$ is the i-th projection of $t(\vec{a})$.

THEOREM 3.5. *Let δ be an I-normal $S^1_2(\Box^p_1)$ derivation of Δ, where Δ is a sequence of purely existential formulas. Then there is a term $t^{HA}(\vec{a})$ denoting a polynomial time computable function so that*

$$(\mathbf{QF}(\Box^p_1) - PIND) \vdash \Delta[t^{HA}(\vec{a})].$$

PROOF.

1. $(\wedge)$-inference.

$$\frac{\Delta,\ A_0 \qquad \Delta,\ A_1}{\Delta,\ A_0 \wedge A_1}$$

By induction hypothesis there are terms $l_0(\vec{a})$ and $l_1(\vec{a})$ denoting polynomial time computable functions so that

$$(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash \Delta[l_0(\vec{a})],\ A_0$$

and

$$(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash \Delta[l_1(\vec{a})],\ A_1.$$

Define $t^{HA}(\vec{a})$ as the choice between $l_0(\vec{a})$ and $l_1(\vec{a})$, $t^{HA}(\vec{a}) := choice_\Delta(l_0(\vec{a}),l_1(\vec{a}))$, where $choice_\Delta(l_0(\vec{a}),l_1(\vec{a}))$ means 'check $\bigvee_{i=1}^n \phi_i(\vec{a},l_0(\vec{a})_i)$. If it is true then assign $l_0(\vec{a})$ to $t^{HA}(\vec{a})$, otherwise put $t^{HA}(\vec{a}) := l_1(\vec{a})$'. Thus

$$(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash \Delta[t^{HA}(\vec{a})],\ A_0 \wedge A_1.$$

2. $(\exists)$-inference.

$$\frac{\Delta,\ \theta(\vec{a},t(\vec{a}))}{\Delta,\ \exists y\,\theta(\vec{a},y)},$$

where θ is open.

By induction hypothesis there is a term denoting a $\Box^p_1$-function $l(\vec{a})$ representing a sequence of n-terms so that

$$(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash \Delta[l(\vec{a})],\ \theta(\vec{a},t(\vec{a})).$$

Define $t^{HA}(\vec{a})_i := l(\vec{a})_i$ $(i=0,\ldots,n)$, and $t^{HA}(\vec{a})_{n+1} := t(\vec{a})$. Thus

$$(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash \Delta[t^{HA}(\vec{a})],\theta(\vec{a},t^{HA}(\vec{a})_{n+1}).$$

3. $(PIND)$-inference:
The principal formula of induction can be either open or (purely) Σ^b_1. We consider the second case only.

$$\frac{\Delta,\ \varphi(\vec{a},0) \qquad \Delta,\ \neg\varphi(\vec{a},\lfloor\tfrac{1}{2}b\rfloor),\ \varphi(\vec{a},b)}{\Delta,\ \varphi(\vec{a},t(\vec{a}))},$$

where $\varphi(\vec{a},x)$ is $\exists y\,\theta(\vec{a},y,x)$, $\theta(\vec{a},y,x)$ is $y \le s(\vec{a}) \wedge \theta'(\vec{a},y,x)$ and θ' is open.

By induction hypothesis there are terms $l_0(\vec{a})$ and $l_1(\vec{a},z,b)$ denoting polynomial time computable functions so that

$$(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash \Delta[l_0(\vec{a})],\ \theta(\vec{a},l_0(\vec{a})_{n+1},0) \tag{5}$$

and

$$\begin{aligned}(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash\ & \Delta[l_1(\vec{a},z,b)],\ \neg\theta(\vec{a},z,\lfloor\tfrac{1}{2}b\rfloor),\\ & \theta(\vec{a},l_1(\vec{a},z,b)_{n+1},b),\end{aligned} \tag{6}$$

Define function f by recursion on notation as follows:

$$\begin{aligned} f(\vec{a},0) &= l_0(\vec{a})_{n+1}\\ f(\vec{a},x) &= l_1(\vec{a},f(\vec{a},\lfloor\tfrac{1}{2}x\rfloor),x)_{n+1}.\end{aligned}$$

Clearly f is polynomial time computable, since for every x $f(\vec{a},x) \le s(\vec{a})$.

[8] The formulation and proof of theorem 3.5 was obtained, with the helpful comments and suggestions of Pitassi and Sieg, during the Spring 1996 Dimacs workshop. Very useful further suggestions were given to me by both Avigad and Sieg.

So (5) and (6) can be transformed into the following inferences:

$$(\mathbf{QF}(\Box_1^p) - \text{PIND}) \vdash \Delta[l_0(\vec{a})],\ \theta(\vec{a}, f(\vec{a}, 0), 0) \tag{7}$$

and

$$\begin{aligned}(\mathbf{QF}(\Box_1^p) - \text{PIND}) \quad \vdash \quad & \Delta[l_1(\vec{a}, f(\vec{a}, \lfloor \tfrac{1}{2} b \rfloor), b)], \\ & \neg\theta(\vec{a}, f(\vec{a}, \lfloor \tfrac{1}{2} b \rfloor), \lfloor \tfrac{1}{2} b \rfloor),\ \theta(\vec{a}, f(\vec{a}, b), b).\end{aligned} \tag{8}$$

Consider now Δ. Define a term $l(\vec{a})$ as follows (for better readability we will omit the parameter variables $\vec{a}$ of t):

$$l(\vec{a}) := \mathit{choice}_\Delta(\, l_0(\vec{a}),\ l_1(\vec{a}, f(\vec{a}, t\lceil h(\vec{a}, |t|) \dot{-} 1), t\lceil h(\vec{a}, |t|)) \,),$$

where h is a function defined by sharply bounded search that finds the least length $k \leq |t|$ so that $\bigvee_{i=1}^n \phi_i(\vec{a}, l_1(\vec{a}, f(\vec{a}, t\lceil k \dot{-} 1), t(\vec{a})\lceil k))$. Suppose that $\neg \bigvee_{i=1}^n \phi_i(\vec{a}, l_0(\vec{a}))$, then there is a minimum length k, $0 \leq k \leq |t|$ so that $l(\vec{a}) := l_1(\vec{a}, f(\vec{a}, t\lceil k \dot{-} 1), t\lceil k)$.

Consider then (7). If $\neg \bigvee_{i=1}^n \phi_i(\vec{a}, l_0(\vec{a}))$ then either some open formula in Δ is true or $\theta(\vec{a}, f(\vec{a}, 0), 0)$ is true. Therefore we can prove the following sequent

$$(\mathbf{QF}(\Box_1^p) - \text{PIND}) \vdash \Delta[l_1(\vec{a}, f(\vec{a}, 0), 0)] \vee \theta(\vec{a}, f(\vec{a}, 0), 0). \tag{9}$$

From (8), with b substituted by $t\lceil m + 1$, we can easily prove

$$\begin{aligned}(\mathbf{QF}(\Box_1^p) - \text{PIND}) \quad \vdash \quad & \neg(\Delta[l_1(\vec{a}, f(\vec{a}, t\lceil m \dot{-} 1), t\lceil m)] \vee \\ & \theta(\vec{a}, f(\vec{a}, t\lceil m), t\lceil m)), \\ \Delta[l_1(\vec{a}, f(\vec{a}, t\lceil m), t\lceil m + 1)] \quad \vee \quad & \theta(\vec{a}, f(\vec{a}, t\lceil m + 1), t\lceil m + 1).\end{aligned} \tag{10}$$

Apply open induction on (9) and (10) to obtain

$$(\mathbf{QF}(\Box_1^p) - \text{PIND}) \vdash \Delta[l_1(\vec{a}, f(\vec{a}, \lfloor \tfrac{1}{2} t \rfloor), t)],\ \theta(\vec{a}, f(\vec{a}, t), t).$$

Define now $t^{HA}(\vec{a})$ in terms of $l(\vec{a})$ and $f(\vec{a})$

$$t^{HA}(\vec{a})_i := \begin{cases} l(\vec{a})_i & \text{if } i = 0, \ldots, n \\ f(\vec{a}, t) & \text{if } i = n + 1 \end{cases}.$$

Thus

$$(\mathbf{QF}(\Box_1^p) - \text{PIND}) \vdash \Delta[t^{HA}(\vec{a})],\ \theta(\vec{a}, t^{HA}(\vec{a})_{n+1}, t).$$

□

4. Comparison.

In this section I compare (my version of the) witnessing method and Herbrand analysis as presented in the previous section.

Consider a sequence of purely existential formulas $\Delta[\exists y_1 \psi_1, \ldots, \exists y_n \psi_n]$. Herbrand analysis extracts a term $t^{HA}(\vec{a})$ so that each projection $t^{HA}(\vec{a})_i$, $(i = 1, \ldots, n)$, is the Skolem term for occurrences of y_i in ψ_i. We call such term $t_i^{HA}(\vec{a})$. The witnessing method instead extracts a witness $g(\vec{a})$ which represents a sequence of values so that $\beta(1, g(\vec{a})_{\exists y_i \psi_i})$ witnesses all occurrences of y_i in ψ_i. We call such term $t_i^W(\vec{a})$. Notice that if Δ contains open formulas $\varphi_1, \ldots, \varphi_k$ then the sequence denoted by $g(\vec{a})$ has length $k + n$.

We want to compare the terms extracted by the two methods. In other words we want to compare the *computational complexities* of (the algorithms associated to) these terms.

Theorem 4.1 proves that $g(\vec{a})$ is more complex than $t^{HA}(\vec{a})$ according to the definition of complexity that we will introduce shortly. We will then show that the computational complexity of each $t_i^{HA}(\vec{a})$, $i = 1, \ldots, n$, is at most equal to the computational complexity of each $t_i^W(\vec{a})$ (corollary 4.2 and corollary 4.3).

We want to propose here what we consider a "natural" definition of computational complexity for the terms extracted by Buss' and Sieg's methods.

Given a term $t(\vec{x})$, we define $C_{t(\vec{x})}(\vec{a})$ inductively according to the following steps. Note that $C_{t(\vec{x})}(\vec{a})$ gives a reasonable measure of the computational complexity of t on input $\vec{a}$.

1. If g is one of the basic functions, then $C_{g(\vec{x})}(\vec{a}) = 1$.
2. $C_{f(g(\vec{x}))}(\vec{a}) = C_{g(\vec{x})}(\vec{a}) + C_{f(\vec{x})}(g(\vec{a}))$.
   To compute $f(\vec{a}, g(\vec{a}))$ we first compute g with input $\vec{a}$ and then we apply f to the result.
3. Let Δ be the sequence of purely existential formulas $\{\exists y_1 \psi_1(\vec{a}, y_1), \ldots, \exists y_n \psi_n(\vec{a}, y_n)\}$, with $n \geq 1$. Then

$$\begin{aligned} C_{\mathit{choice}_\Delta(g_0(\vec{x}), g_1(\vec{x}))}(\vec{a}) &= C_{g_0(\vec{x})}(\vec{a}) + C_{g_1(\vec{x})}(\vec{a}) + \\ & \quad C_{\mathit{Witness}_\Delta(y, \vec{x})}(g_0(\vec{a}), \vec{a}) \\ &= C_{g_0(\vec{x})}(\vec{a}) + C_{g_1(\vec{x})}(\vec{a}) + \\ & \quad \sum_{i=1}^{n} C_{\psi_i(\vec{x}, y_i)}(\vec{a}, \beta(i, g_0(\vec{a}))), \end{aligned}$$

where $C_{\psi_i(\vec{x}, y_i)}(\vec{a}, \beta(i, g_0(\vec{a})))$ denotes the time required to determine whether $\psi_i(\vec{a}, \beta(i, g_0(\vec{a})))$ is true. Note that if Δ contains also open formulas $\varphi_1, \ldots, \varphi_k$, then the complexity of checking whether $g_0(\vec{a})$

is the witness of Δ will be

$$\sum_{j=1}^{k} C_{\varphi_j}(\vec{a}) + \sum_{i=1}^{n} C_{\psi_i(\vec{x},y_i)}(\vec{a}, \beta(i, g_0(\vec{a}))).$$

4. Let ϕ be an open formula.

$$\begin{aligned} C_{choice_\phi(t_0(\vec{x}),t_1(\vec{x}))}(\vec{a}) &= C_{t_0(\vec{x})}(\vec{a}) + C_{t_1(\vec{x})}(\vec{a}) + \\ & \quad C_{\phi(\vec{x},y)}(\vec{a}, t_0(\vec{a})), \end{aligned}$$

where $C_{\phi(\vec{x},y)}(\vec{a}, t_0(\vec{a}))$ denotes the time required to determine whether $\phi(\vec{a}, t_0(\vec{a}))$ is true.

5. Given functions g and h define f by *limited recursion on notation* as follows

$$\begin{aligned} f(\vec{a}, 0) &= g(\vec{a}) \\ f(\vec{a}, y) &= h(\vec{a}, f(\vec{a}, \lfloor \tfrac{1}{2} y \rfloor), y) \end{aligned}$$

Let $f(\vec{a}, y) \leq s(\vec{a})$, for every y. Then

$$\begin{aligned} C_{f(\vec{x},t(\vec{x}))}(\vec{a}) &= C_{g(\vec{x})}(\vec{a}) + C_{t(\vec{x})}(\vec{a}) + \\ & \quad \sum_{i=0}^{|t(\vec{a})|} C_{h(\vec{x},z,b)}(\vec{a}, f(\vec{a}, t(\vec{a})\lceil i \dot{-} 1), t(\vec{a})\lceil i). \end{aligned}$$

In other words, to compute $f(\vec{a}, t(\vec{a}))$ we must compute the values of f at 0 up to $t(\vec{a})$ by doubling each time the previous value (i.e. from $t(\vec{a})\lceil 0$ up to $t(\vec{a})\lceil |t(\vec{a})|$).

6. Let θ be an open formula. Define function h by *sharply bounded search* as follows

$$h(\vec{a}, m+1) = \begin{cases} h(\vec{a}, m) & \text{if } \theta(\vec{a}, t(\vec{a})\lceil h(\vec{a}, m)) \\ h(\vec{a}, m) + 1 & \text{otherwise} \end{cases}$$

Then

$$C_{h(\vec{x},|t(\vec{x})|)}(\vec{a}) = C_{|t(\vec{x})|}(\vec{a}) + \sum_{i=1}^{|t(\vec{a})|} C_{\theta(\vec{x},y)}(\vec{a}, t(\vec{a})\lceil i \dot{-} 1).$$

We want to find the first $n \leq |t(\vec{a})|$ so that $\theta(\vec{a}, t(\vec{a})\lceil h(\vec{a}, n))$. To do that we need to check for $i = 0, \ldots, n \dot{-} 1$ if $\theta(\vec{a}, t(\vec{a})\lceil h(\vec{a}, i))$. There are at most $|t(\vec{a})|$ of such checks to perform.

7. Let $t_i^W(\vec{a}) := \beta(i, g(\vec{a})) = r_i(\vec{a})$. Then

$$C_{t_i^W(\vec{x})}(\vec{a}) = C_{g(\vec{x})}(\vec{a}) + C_{r_i(\vec{x})}(\vec{a}).$$

8. For any $\vec{a}$, $|t(\vec{a})| \leq C_{t(\vec{x})}(\vec{a})$.

9. $C_{t(\vec{x})}(\vec{a}) \geq 1$.

In a sense $C_{t(\vec{x})}(\vec{a})$ gives a *natural* upper bound on the number of steps required to compute t on input $\vec{a}$.

For each term t we also define a function $B_t(k)$ which bounds the length of the computation of t over inputs of length at most k; that is, for every $\vec{a}$, we have $C_t(\vec{a}) \leq B_t(|\vec{a}|)$. B_t is defined inductively, as follows:

1. $B_{f(\vec{x})}(k) = \sup_{\{\vec{a} \mid |\vec{a}| \leq k\}} C_f(\vec{a})$, for initial functions f.

2. $B_{f(g(\vec{x}))}(k) = B_{g(x)}(k) + B_{f(\vec{x})}(B_{g(\vec{x})}(k))$.

3. $B_{choice_\Delta(g_0(\vec{x}),g_1(\vec{x}))}(k) = B_{g_0(\vec{x})}(k) + B_{g_1(\vec{x})}(k) + \sum_{i=1}^{n} sup_{\{(\vec{a},g_0(\vec{a})) \mid |a| \leq k \wedge |g_0(\vec{a})| \leq B_{g_0(\vec{x})}(k)\}} C_{\phi_i(\vec{x},y_i)}(\vec{a}, \beta(i, g_0(\vec{a})))$.

4. $B_{choice_\phi(t_0(\vec{x}),t_1(\vec{x}))}(k) = B_{t_0(\vec{x})}(k) + B_{t_1(\vec{x})}(k) + sup_{\{(\vec{a},t_0(\vec{a})) \mid |a| \leq k \wedge |t_0(\vec{a})| \leq B_{t_0(\vec{x})}(k)\}} C_{\phi(\vec{x},y)}(\vec{a}, t_0(\vec{a}))$.

5. Recursion.

$$B_{f(\vec{x},t(\vec{x}))}(k) = B_{g(\vec{x})}(k) + B_{t(\vec{x})}(k) + B_{s(\vec{x})}(k) + |t(\vec{a})| \cdot sup_{\{(\vec{a},s(\vec{a}),t(\vec{a})) \mid |\vec{a}| \leq k \wedge |s(\vec{a})| \leq B_{s(\vec{x})}(k) \wedge |t(\vec{a})| \leq B_{t(\vec{x})}(k)\}} C_{h(\vec{x},z,b)}(\vec{a}, s(\vec{a}), t(\vec{a}))$$

6. (Sharply) bounded search.

$$B_{h(\vec{x},|t(\vec{x})|)}(k) = B_{t(\vec{x})}(k) + |t(\vec{a})| \cdot sup_{\{(\vec{a},t(\vec{a})) \mid |\vec{a}| \leq k \wedge |t(\vec{a})| \leq B_{t(\vec{x})}(k)\}} C_{h(\vec{x},b)}(\vec{a}, |t(\vec{a})|)$$

Note that B_t is nondecreasing.

THEOREM 4.1. *Let δ be an I-normal $S_2^1(\Box_1^p)$-derivation of a sequence of purely existential formulas Δ. Then there are term $t^{HA}(\vec{a})$ and function symbol g denoting polynomial time computable functions so that*

1. $(\mathbf{QF}(\Box_1^p) - PIND) \vdash \Delta[t^{HA}(\vec{a})]$,

2. $(\mathbf{QF}(\Box_1^p) - PIND) \vdash \Delta[g(\vec{a})]$,

3. *For every v, $B_{t^{HA}(\vec{x})}(v) \leq B_{g(\vec{x})}(v)$.*

PROOF. The proof of 1 and 2 is given by theorem 3.5, theorem 3.3 and corollary 3.4 respectively. We prove here 3. The proof is by induction on derivation δ. The problematic inferences are those with two premises. For simplicity we will consider only the case of cut-inferences. The others behave similarly. Consider the following inference

$$\frac{\Delta, A_0 \qquad \Delta, A_1}{\Delta}$$

is the witness of Δ will be

$$\sum_{j=1}^{k} C_{\varphi_j}(\vec{a}) + \sum_{i=1}^{n} C_{\psi_i(\vec{x},y_i)}(\vec{a}, \beta(i, g_0(\vec{a}))).$$

4. Let ϕ be an open formula.

$$\begin{aligned} C_{choice_\phi(t_0(\vec{x}),t_1(\vec{x}))}(\vec{a}) &= C_{t_0(\vec{x})}(\vec{a}) + C_{t_1(\vec{x})}(\vec{a}) + \\ & \quad C_{\phi(\vec{x},y)}(\vec{a}, t_0(\vec{a})), \end{aligned}$$

where $C_{\phi(\vec{x},y)}(\vec{a}, t_0(\vec{a}))$ denotes the time required to determine whether $\phi(\vec{a}, t_0(\vec{a}))$ is true.

5. Given functions g and h define f by *limited recursion on notation* as follows

$$\begin{aligned} f(\vec{a}, 0) &= g(\vec{a}) \\ f(\vec{a}, y) &= h(\vec{a}, f(\vec{a}, \lfloor \tfrac{1}{2} y \rfloor), y) \end{aligned}$$

Let $f(\vec{a}, y) \leq s(\vec{a})$, for every y. Then

$$\begin{aligned} C_{f(\vec{x},t(\vec{x}))}(\vec{a}) &= C_{g(\vec{x})}(\vec{a}) + C_{t(\vec{x})}(\vec{a}) + \\ & \quad \sum_{i=0}^{|t(\vec{a})|} C_{h(\vec{x},z,b)}(\vec{a}, f(\vec{a}, t(\vec{a})\lceil i \dot{-} 1), t(\vec{a})\lceil i). \end{aligned}$$

In other words, to compute $f(\vec{a}, t(\vec{a}))$ we must compute the values of f at 0 up to $t(\vec{a})$ by doubling each time the previous value (i.e. from $t(\vec{a})\lceil 0$ up to $t(\vec{a})\lceil |t(\vec{a})|$).

6. Let θ be an open formula. Define function h by *sharply bounded search* as follows

$$h(\vec{a}, m+1) = \begin{cases} h(\vec{a}, m) & \text{if } \theta(\vec{a}, t(\vec{a})\lceil h(\vec{a}, m)) \\ h(\vec{a}, m) + 1 & \text{otherwise} \end{cases}$$

Then

$$C_{h(\vec{x},|t(\vec{x})|)}(\vec{a}) = C_{|t(\vec{x})|}(\vec{a}) + \sum_{i=1}^{|t(\vec{a})|} C_{\theta(\vec{x},y)}(\vec{a}, t(\vec{a})\lceil i \dot{-} 1).$$

We want to find the first $n \leq |t(\vec{a})|$ so that $\theta(\vec{a}, t(\vec{a})\lceil h(\vec{a}, n))$. To do that we need to check for $i = 0, \ldots, n \dot{-} 1$ if $\theta(\vec{a}, t(\vec{a})\lceil h(\vec{a}, i))$. There are at most $|t(\vec{a})|$ of such checks to perform.

7. Let $t_i^W(\vec{a}) := \beta(i, g(\vec{a})) = r_i(\vec{a})$. Then

$$C_{t_i^W(\vec{x})}(\vec{a}) = C_{g(\vec{x})}(\vec{a}) + C_{r_i(\vec{x})}(\vec{a}).$$

8. For any $\vec{a}$, $|t(\vec{a})| \leq C_{t(\vec{x})}(\vec{a})$.

9. $C_{t(\vec{x})}(\vec{a}) \geq 1$.

In a sense $C_{t(\vec{x})}(\vec{a})$ gives a *natural* upper bound on the number of steps required to compute t on input $\vec{a}$.

For each term t we also define a function $B_t(k)$ which bounds the length of the computation of t over inputs of length at most k; that is, for every $\vec{a}$, we have $C_t(\vec{a}) \leq B_t(|\vec{a}|)$. B_t is defined inductively, as follows:

1. $B_{f(\vec{x})}(k) = \sup_{\{\vec{a} \mid |\vec{a}| \leq k\}} C_f(\vec{a})$, for initial functions f.

2. $B_{f(g(\vec{x}))}(k) = B_{g(x)}(k) + B_{f(\vec{x})}(B_{g(\vec{x})}(k))$.

3. $B_{choice_\Delta(g_0(\vec{x}),g_1(\vec{x}))}(k) = B_{g_0(\vec{x})}(k) + B_{g_1(\vec{x})}(k) +$
$\sum_{i=1}^{n} sup_{\{(\vec{a},g_0(\vec{a})) \mid |a| \leq k \wedge |g_0(\vec{a})| \leq B_{g_0(\vec{x})}(k)\}} C_{\phi_i(\vec{x},y_i)}(\vec{a}, \beta(i, g_0(\vec{a})))$.

4. $B_{choice_\phi(t_0(\vec{x}),t_1(\vec{x}))}(k) = B_{t_0(\vec{x})}(k) + B_{t_1(\vec{x})}(k) +$
$sup_{\{(\vec{a},t_0(\vec{a})) \mid |a| \leq k \wedge |t_0(\vec{a})| \leq B_{t_0(\vec{x})}(k)\}} C_{\phi(\vec{x},y)}(\vec{a}, t_0(\vec{a})))$.

5. Recursion.

$$B_{f(\vec{x},t(\vec{x}))}(k) = B_{g(\vec{x})}(k) + B_{t(\vec{x})}(k) + B_{s(\vec{x})}(k) + |t(\vec{a})| \cdot$$
$$sup_{\{(\vec{a},s(\vec{a}),t(\vec{a})) \mid |\vec{a}| \leq k \wedge |s(\vec{a})| \leq B_{s(\vec{x})}(k) \wedge |t(\vec{a})| \leq B_{t(\vec{x})}(k)\}} C_{h(\vec{x},z,b)}(\vec{a}, s(\vec{a}), t(\vec{a})).$$

6. (Sharply) bounded search.

$$B_{h(\vec{x},|t(\vec{x})|)}(k) = B_{t(\vec{x})}(k) + |t(\vec{a})| \cdot$$
$$sup_{\{(\vec{a},t(\vec{a})) \mid |\vec{a}| \leq k \wedge |t(\vec{a})| \leq B_{t(\vec{x})}(k)\}} C_{h(\vec{x},b)}(\vec{a}, |t(\vec{a})|).$$

Note that B_t is nondecreasing.

THEOREM 4.1. *Let δ be an I-normal $S_2^1(\Box_1^p)$-derivation of a sequence of purely existential formulas Δ. Then there are term $t^{HA}(\vec{a})$ and function symbol g denoting polynomial time computable functions so that*

1. $(\mathbf{QF}(\Box_1^p) - PIND) \vdash \Delta[t^{HA}(\vec{a})]$,

2. $(\mathbf{QF}(\Box_1^p) - PIND) \vdash \Delta[g(\vec{a})]$,

3. *For every v, $B_{t^{HA}(\vec{x})}(v) \leq B_{g(\vec{x})}(v)$.*

PROOF. The proof of 1 and 2 is given by theorem 3.5, theorem 3.3 and corollary 3.4 respectively. We prove here 3. The proof is by induction on derivation δ. The problematic inferences are those with two premises. For simplicity we will consider only the case of cut-inferences. The others behave similarly. Consider the following inference

$$\frac{\Delta, A_0 \qquad \Delta, A_1}{\Delta}$$

Suppose that Herbrand analysis assigns terms $t_0(\vec{a})$ and $t_1(\vec{a})$ to Δ in the premises. On the other hand, the witnessing method assigns $g_0(\vec{a})$ and $g_1(\vec{a})$ to Δ in the premises.

Now the choice between $t_0(\vec{a})$ and $t_1(\vec{a})$ depends on whether at least one of the formulas in Δ is true with Skolem term extracted from $t_0(\vec{a})$. If yes, then $t^{HA}(\vec{a})$ is $t_0(\vec{a})$, otherwise it is $t_1(\vec{a})$. So

$$B_{t^{HA}(\vec{x})}(v) = B_{t_0(\vec{x})}(v) + B_{t_1(\vec{x})}(v) + \sum_{i=1}^{n} sup_{\{(\vec{a},t_0(\vec{a}))||\vec{a}|\leq v \wedge |t_0(\vec{a})|\leq B_{t_0(\vec{x})}(v)\}}(C_{\psi_i}(\vec{a}, t_0(\vec{a}))).$$

In the case of the witnessing method, instead, we choose between $g_0(\vec{a})$ and $g_1(\vec{a})$ according to whether $g_0(\vec{a})$ witnesses at least one of the formulas in Δ. $g(\vec{a})$ is the result of such choice. Since Δ may contain also open formulas $\varphi_1, \ldots, \varphi_k$, then the choice of the witness depends also on such formulas. Thus

$$B_{g(\vec{x})}(v) = B_{g_0(\vec{x})}(v) + B_{g_1(\vec{x})}(v) + \sum_{i=1}^{k+n} sup_{\{(\vec{a},g_0(\vec{a}))||\vec{a}|\leq v \wedge |g_0(\vec{a})|\leq B_{g_0(\vec{x})}(v)\}}(C_{\phi_i}(\vec{a}, g_0(\vec{a}))),$$

where ϕ_i is either φ_j $(j = 1, \ldots, k)$ or ψ_m $(m = 1, \ldots, n)$.

It is easy to see that the number of checks made by the witnessing theorem is higher than the number of checks made by Herbrand analysis if there are open formulas occurring in Δ. □

The above result is important if some very complex open formulas occur in Δ. Clearly if the complexity of such formulas is instead very small then the fact that we must perform more checks does not cost too much and may be irrelevant.

Let Δ be $\Delta[\exists y_1 \phi_1, \ldots, \exists y_n \phi_n]$. Write $t^W(\vec{a})$ to indicate the part of $g(\vec{a})$ that witnesses the existential variables $y_1, \ldots, y_n$. If we limit our attention to the complexity of $t^W(\vec{a})$, instead of $g(\vec{a})$, still the term extracted by Herbrand analysis is less complex than the one extracted by the witnessing method as corollary 4.2 shows. The reason is that to choose between the witnesses assigned in the premises in the case of two-premise inferences we must consider all the side formulas of such inferences and not simply occurrences of existential formulas. Nevertheless there is no reason why we should consider all the side formulas and not just limit our attention to occurrences of existential formulas. If we choose to consider only occurrences of existential formulas then the terms extracted by the two methods have the same computational complexity as corollary 4.3 shows.

COROLLARY 4.2. *Let δ be an I-normal $S^1_2(\Box^p_1)$-derivation of a sequence of purely existential formulas Δ. Then there are terms $t^{HA}(\vec{a})$ and $t^W(\vec{a})$ denoting polynomial time computable functions so that*

1. $(\mathbf{QF}(\Box^p_1) - PIND) \vdash \Delta[t^{HA}(\vec{a})]$,
2. $(\mathbf{QF}(\Box^p_1) - PIND) \vdash \Delta[t^W(\vec{a})]$,
3. *For every v, $B_{t^{HA}(\vec{x})}(v) \leq B_{t^W(\vec{x})}(v)$.*

COROLLARY 4.3. *Let δ be an I-normal $S^1_2(\Box^p_1)$-derivation of a sequence of purely existential formulas Δ. Then there are terms $t^{HA}(\vec{a})$ and $t^W(\vec{a})$ denoting polynomial time computable functions so that*

1. $(\mathbf{QF}(\Box^p_1) - PIND) \vdash \Delta[t^{HA}(\vec{a})]$,
2. $(\mathbf{QF}(\Box^p_1) - PIND) \vdash \Delta[t^W(\vec{a})]$,
3. *For every v, $B_{t^{HA}(\vec{x})}(v) = B_{t^W(\vec{x})}(v)$.*

5. Alternative strategies.

In section 3 we showed a particular strategy to extract terms in (our modified versions of) Buss' and Sieg's methods. Here we want to explore some other possible strategies and compare them.

- We will consider an alternative approach to the solution of the case of induction inferences.
- We will look at an alternative way of treating formulas in a sequent.

5.1. Induction inferences.

In the proofs of theorems 3.3 and 3.5 the witness, or Skolem term respectively, assigned in the conclusion of an induction inference is built by searching for the first point at which the side formulas are true. Here we will instead look for the first point at which the induction formula is false (which clearly renders the side formulas true).

For simplicity we will show how this new strategy is applied by just one of the two metods. We choose Herbrand analysis.

Consider the following induction inference

$$\frac{\Delta, \varphi(\vec{a}, 0) \qquad \Delta, \neg\varphi(\vec{a}, \lfloor \tfrac{1}{2} b \rfloor), \varphi(\vec{a}, b)}{\Delta, \varphi(\vec{a}, t(\vec{a}))},$$

where $\varphi(\vec{a}, x)$ is $\exists y\, \theta(\vec{a}, y, x)$, $\theta(\vec{a}, y, x)$ is $y \leq s(\vec{a}) \wedge \theta'(\vec{a}, y, x)$ and θ' is open. To find the Skolem term $t^{HA}(\vec{a})$ for Δ we look for the least length $\leq |t(\vec{a})|$ at which the induction formula fails. For better readability we will write t istead of $t(\vec{a})$.

By induction hypothesis there are $S^1_2(\Box^p_1)$-terms $l_0(\vec{a})$, $l_1(\vec{a}, z, b)$,

$$(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash \Delta[l_0(\vec{a})],\ \theta(\vec{a}, l_0(\vec{a})_{n+1}, 0) \tag{11}$$

and

$$\begin{aligned}(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash\ & \Delta[l_1(\vec{a}, z, b)],\ \neg\theta(\vec{a}, z, \lfloor \tfrac{1}{2} b \rfloor), \\ & \theta(\vec{a}, l_1(\vec{a}, z, b)_{n+1}, b).\end{aligned} \tag{12}$$

Define function f by recursion on notation as follows:

$$\begin{aligned} f(\vec{a}, 0) &= l_0(\vec{a})_{n+1} \\ f(\vec{a}, x) &= l_1(\vec{a}, f(\vec{a}, \lfloor \tfrac{1}{2} x \rfloor), x)_{n+1}. \end{aligned}$$

Clearly f is polynomial time computable, since for every x $f(\vec{a}, x) \leq s(\vec{a})$. So (11) and (12) can be transformed into the following inferences:

$$(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash \Delta[l_0(\vec{a})],\ \theta(\vec{a}, f(\vec{a}, 0), 0) \tag{13}$$

and, for b substituted by $t\lceil m+1$

$$\begin{aligned}(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash\ & \Delta[l_1(\vec{a}, f(\vec{a}, t\lceil m), t\lceil m+1)], \\ & \neg\theta(\vec{a}, f(\vec{a}, t\lceil m), t\lceil m), \\ & \theta(\vec{a}, f(\vec{a}, t\lceil m+1), t\lceil m).\end{aligned} \tag{14}$$

(13) and (14) are the premises of an open induction. We can therefore prove the following

$$\begin{aligned}(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash\ & \neg\theta(\vec{a}, f(\vec{a}, 0), 0) \vee \theta(\vec{a}, f(\vec{a}, |t|), t) \vee \\ & \exists y < |t|\, \neg(\, \neg\theta(\vec{a}, f(\vec{a}, y \dot{-} 1), t\lceil(y \dot{-} 1)) \vee \\ & \theta(\vec{a}, f(\vec{a}, y), t\lceil y)\,).\end{aligned}$$

Thus, by *sharply bounded search* we can eliminate the sharply bounded existential quantifier by introducing a polynomial function h so that

$$\begin{aligned}(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash\ & \neg\theta(\vec{a}, f(\vec{a}, 0), 0) \vee \theta(\vec{a}, f(\vec{a}, |t|), t) \vee \\ & \neg(\neg\theta(\vec{a}, f(\vec{a}, h(\vec{a}, |t|) \dot{-} 1), t(\vec{a})\lceil h(\vec{a}, |t|) \dot{-} 1) \vee \\ & \theta(\vec{a}, f(\vec{a}, h(\vec{a}, |t|)), t\lceil h(\vec{a}, |t|))).\end{aligned} \tag{15}$$

Function $h(\vec{a}, |t|)$ finds the smallest $k \leq |t|$ so that $\neg\theta(\vec{a}, f(\vec{a}, k), t\lceil k)$.
Cut (15) with (13) and (14), with b substituted by $tgessi\lceil h(\vec{a}, |t|)$, to obtain

$$(\mathbf{QF}(\Box^p_1) - \mathrm{PIND}) \vdash \Delta[t^{HA}(\vec{a})],\ \theta(\vec{a}, t^{HA}(\vec{a})_{n+1}, t),$$

where for all $i = 1, \ldots, n$

$$t^{HA}(\vec{a})_i := choice_\Delta(\, l_0(\vec{a})_i, l_1(\vec{a}, f(\vec{a}, h(\vec{a}, |t|) \dot{-} 1), t\lceil h(\vec{a}, |t|))_i\,),$$

and

$$t^{HA}(\vec{a})_{n+1} := f(\vec{a}, h(\vec{a}, |t|)).$$

5.2. Comparison with the extraction strategy adopted in the proof of theorem 3.5.
Let us now compare the term extracted in the proof of theorem 3.5 in the case of induction inferences and the alternative suggested above.
To compute the term $t^{HA}(\vec{a})$ extracted in the proof of theorem 3.5 we compute f first and then l_0 and check whether there is an i $(i = 1, \ldots, n)$ so that $\psi_i(\vec{a}, l_0(\vec{a})_i)$ is true. If yes we can stop our computation. Otherwise we find the first k $(0 \leq k \leq n)$ so that $\bigvee_{i=1}^n \psi_i(\vec{a}, l_1(\vec{a}, f(\vec{a}, t\lceil k \dot{-} 1), t\lceil k))_i)$ is true. We will have at most $|t|$ many checks to perform.
To compute term $t^{HA}(\vec{a})$ extracted by the alternative procedure showed in section 5.1 we must compute f first and then l_0 and check whether there is an i $(i = 1, \ldots, n)$ so that $\psi_i(\vec{a}, l_0(\vec{a})_i)$ is true. If yes we can stop our computation. Otherwise we find the first k $(0 \leq k \leq n)$ so that $\neg\theta(\vec{a}, f(\vec{a}, t\lceil k), t\lceil k)$ is true. We will have at most $|t|$ many of such checks to perform. Eventually we will compute $l_0(\vec{a})$ and check whether $\bigvee \psi_i(\vec{a}, l_0(\vec{a})_i)$ is true. If yes then we stop otherwise we compute $l_1(\vec{a}, f(\vec{a}, t\lceil k), t\lceil k+1)$.
Thus in both cases we compute f and l_0 and we check whether $\bigvee \psi_i(\vec{a}, l_0(\vec{a})_i)$ is true. Then in the first case we compute $|t|$-many times l_1 and check the existential formulas in Δ with projections of l_1 as Skolem terms. In the second case, instead we compute l_1 only once. And we find k $(0 \leq k \leq n)$ by performing the checks on just one formula (the induction formula.

Suppose now that the complexity of the formulas in Δ and of the principal formula of the induction is the same.

The difference between the two strategies seems to depend on the type of checks performed. The first strategy requires at most $(|t|+1) \cdot n$ many of such checks, where n is the number of existential formulas occurring in Δ, $|t|$ indicates the maximum number of checks to perform with l_1 and 1 is for the first check with l_0.
The second strategy, instead checks only one formula. So the maximum number of checks is $|t| + n$, n for the first check on Δ and l_0 and $|t|$ is the maximum number of checks on the principal formula of the induction inference.
Notice that if $|t| + 1 = n$ then the first strategy performs at most n^2 checks, while the maximum number of checks required by the second one is $\leq 2 \cdot n$.

5.3. Assignment of terms to sequences of formulas.
Consider a sequence of purely existential formulas $\Delta[\exists y_1 \psi_1, \ldots, \exists y_n \psi_n]$. Δ may contain also open formulas $\varphi_1, \ldots, \varphi_k$. The terms assigned to the existential variables $y_1, \ldots, y_n$ occurring in Δ may be extracted by looking at Δ as a unique formula, or by considering the individual formulas in Δ separately. The first strategy suggested is the one used to prove theorems 3.3 and 3.5 and in section 5.1. Here we will consider the implications of treating each formula occurring in a sequence independently from the others. As in section 5.1, we will present this alternative extraction procedure as applied

by Herbrand analysis. Same considerations can be used for the witnessing method.

In proving theorem 3.5 we assign to Δ a Skolem term $t^{HA}(\vec{a})$, built by induction on the derivation of Δ. In particular, if Δ is the conclusion of an inference with two premises, $t^{HA}(\vec{a})$ is the result of a choice between the terms $l_0(\vec{a})$ and $l_1(\vec{a})$ assigned in the premises. To be more precise $t^{HA}(\vec{a})$ will be $l_0(\vec{a})$ if there is *at least one* formula $\exists y_i\, \psi_i$ $(i = 0, \ldots, n)$ in Δ so that $\psi_i(\vec{a}, (l_0)_i)$. Otherwise $t^{HA}(\vec{a}) := l_1(\vec{a})$. Here we suggest that for each $\exists y_i \psi_i$ $(i = 1, \ldots, n)$ in Δ the Skolem term $t_i^{HA}(\vec{a})$ assigned to y_i will be chosen according to whether $\psi_i(\vec{a}, l_0(\vec{a})_i)$ only is true.

The first notable difference between the two strategies is that the order of choices is different and the terms assigned to each y_i may be different because each formula $\exists y_i \psi_i$ is considered separately. In other words, suppose that $\psi_i(\vec{a}, l_0(\vec{a})_i)$ is true. Then in the first case we will assign $l_0(\vec{a})_i$ to y_i and $l_0(\vec{a})_{i+1}$ to y_{i+1}. While the second strategy will assign $l_0(\vec{a})_i$ to y_i and $l_0(\vec{a})_{i+1}$ to y_{i+1} *only* if also $\psi_{i+1}(\vec{a}, l_0(\vec{a})_{i+1})$ is true, otherwise it will assign $l_1(\vec{a})_{i+1}$ to y_{i+1}.

The second important difference between the two strategies is that in the first case the Skolem term assigned to y_i is chosen after checking $\bigvee_{j=1}^{n} \psi_j(\vec{a}, l_0(\vec{a})_j)$. The second strategy, instead, only checks $\psi_i(\vec{a}, l_0(\vec{a})_i)$. In other words we have to check at most n formulas in the first case, while we always check only one in the second case.

Let us now combine the second extraction strategy here introduced with the ones previously considered to treat the induction case. It is easy to see that the extraction strategy introduced in section 5.1, which checks the induction formula only, is more efficient than the one used to prove theorem 3.5, which finds the first point where at least one of the side formulas is true.

In section 5.2 we already showed that the strategy introduced in section 5.1 requires less checks than the one used to prove theorem 3.5. Moreover here the latter seems to require an additional work: the construction of a Skolem term for each existential variable in the side formulas. This operation is not necessary since we focus our attention only on individual formulas.

6. Conclusion.

The goal of this paper was to show that Buss' and Sieg's methods, in the context of S_2^1, behave exactly in the same way.

In my analysis I moved away from the specific calculi (Gentzen's sequent calculus and Tait's version of the sequent calculus) in which both methods were presented. I showed that Buss' bootstrapping is nothing else than an

implicit extension of the language of S_2^1. In other words, if I can build polynomial time computable functions from the basic functions whose symbols are allowed in the language of S_2^1, it seems reasonable to add to this language symbols for all the new functions. This very simple operation is very important because I am not forced to check every time a new witness is introduced whether it is still (denoting) a polynomial time computable function.

Another important result reached by this paper is the proof that the witnessing method does not need Parikh's theorem. This avoids an unecessary extension of the original derivation.

After these initial steps I was able to prove that (my versions of) the witnessing method and Herbrand analysis are extracting terms in exactly the same way. Thus the algorithms represented by such terms have the same computational complexity.

Finally I explored different extraction strategies that can be used by both methods and showed that the computational complexity of the terms extracted strongly depends on the particular extraction strategies adopted.

References

[1] Samuel R. Buss: Bounded Arithmetic, Bibliopolis, 1986.
[2] S. A. Cook: *The Complexity of Theorem Proving Procedures.*, Proc. 3rd ACM Symp. on Theory of Computing, pp. 151-158, Shaker Heights, Ohio 1971.
[3] Barbara Kauffmann:*Application of Proof Theory to Computational Complexity Theory: Comparison of Different Methods.*, Ph.D. thesis, Manuscript.
[4] Jan Krajíček: Bounded Arithmetic, Propostional Logic, and Complexity Theory, Encyclopedia of Mathematics and its Applications, Vol. 60, Cambridge University Press, 1995.
[5] Fernando J.I. Ferreira: Polynomial Time Computable Arithmetic and Conservative Extensions, Pennsylvania State University, University Park, PA, Ph.D. dissertation, 1988.
[6] R. Parikh: *Existence and Feasibility in Arithmetic*, Journal ogf Symbolic Logic 36 (1971), pp. 494-508.
[7] Wilfried Sieg: *Fragments of Arithmetic*, Annals of Pure and Applied Logic 28 (1985), pp. 33-71, North Holland.
[8] Wilfried Sieg: *Herbrand Analyses*, Archive for Mathematical Logic 30 (1991), pp. 409-441, Springer-Verlag.

PHILOSOPHY DEPARTMENT, CARNEGIE MELLON UNIVERSITY, PITTSBURGH, PA 15213
E-mail address: bk1k@andrew.cmu.edu

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

Towards lower bounds for bounded-depth Frege proofs with modular connectives

Alexis Maciel and Toniann Pitassi

ABSTRACT. We show that for every prime power p^k, quasipolynomial-size bounded-depth Frege proofs with mod p^k counting connectives can be simulated by quasipolynomial-size proofs of depth 3 consisting of a threshold connective at the output, mod p^k connectives on level two, and AND connectives of small fan-in on level one. We argue that this result is a plausible first step towards proving lower bounds for bounded-depth Frege proofs with modular connectives, an outstanding open problem. We also discuss possible interesting consequences for propositional theorem proving.

1. Introduction

An outstanding problem in logic and complexity theory is to prove superpolynomial lower bounds for classical propositional proof systems, known as Frege systems. This problem appears to be quite difficult, although in the last 10 years, substantial progress has been made on restricted versions of the problem. For example, it is now known that Resolution proofs as well as bounded-depth Frege proofs of the propositional pigeonhole principle require exponential size, and these results hold even if we add the mod p axiom schema for any prime p [**Hak85, Ajt88, Ajt90, BP, Rii**].

It has been noted many times that various proof systems correspond to various Boolean circuit classes, and it has been the case that the proof system lower bound has only been obtained once lower bounds for the corresponding circuit class have been firmly established. For example, bounded-depth Frege systems correspond to bounded-depth circuits, and the lower bounds for bounded-depth Frege systems use and generalize much of the technical machinery behind the AC^0 lower bounds for the parity function. In a similar manner, bounded-depth Frege systems with mod p connectives correspond to $ACC^0[p]$, and Frege systems correspond to NC^1.

Since there are strong lower bounds for $ACC^0[p]$ due to Razborov [**Raz87**] and Smolensky [**Smo87**], the next step in propositional proof complexity is to obtain

---

1991 *Mathematics Subject Classification.* Primary 03F20; Secondary 68Q15.

Part of this work was performed in the departments of Mathematics and Computer Science at the University of Pittsburgh.

The first author was supported in part by NSF Grant CCR-9457782 and CCR-9522084.

The second author was supported in part by NSF Grant CCR-9457782, US-Israel BSF Grant 95-00238, and Grant INT-9600919/ME-103 from NSF and MŠMT (Czech Republic).

exponential lower bounds for the corresponding class of bounded-depth Frege systems with mod p connectives. However, at present we cannot even show superpolynomial lower bounds for bounded-depth Frege systems with modular connectives, even under a reasonable complexity-theoretic assumption like P $\neq$ NP. (Obviously NP $\neq$ coNP implies superpolynomial lower bounds for any propositional proof system, so by reasonable, we mean a weaker assumption than this.) A lot of effort has already gone towards solving this problem, and so far it has resulted in new and elegant algebraic proof systems, such as Nullstellensatz proofs ([**BIK+a**, **BIK+b**]) and Gröbner proofs [**CEI96**]. One promising approach for proving lower bounds for $\mathrm{ACC}^0[p]$ proofs is to prove lower bounds for Gröbner refutations. In fact, it can be shown that lower bounds for $\mathrm{ACC}^0[p]$ proofs imply lower bounds for Gröbner refutations over $\mathbf{Z}_p$. This important step was made very recently by Razborov [**Raza**] who showed that any Gröbner refutation of the propositional pigeonhole principle requires linear degree. (The Gröbner proof system has since been renamed the *polynomial calculus*.)

Unfortunately, lower bounds for the polynomial calculus do not appear to be enough to obtain lower bounds for $\mathrm{ACC}^0[p]$ proofs. The essential problem is that good lower bounds for Gröbner refutations seem to give good lower bounds for a very flat bounded-depth proof system having only mod p connectives at the top of each formula, rather than mod p connectives throughout the bounded-depth formula.

Surprising results from circuit complexity tell us that this may not be a problem. Allender [**All89**] (see also [**AH94**] and [**Yao90**], [**BT94**], [**GKR+95**] and [**Reg93**] for related results) has shown that quasipolynomial-size $\mathrm{ACC}^0[p]$ circuits can be simulated by quasipolynomial-size, depth-3 circuits. The depth-3 circuits are of a special form: the output gate is a threshold gate, the middle layer of gates are mod p gates, and the bottom level consists of AND gates of small fan-in. Alternatively, this result can be viewed as showing that any $\mathrm{ACC}^0[p]$ circuit can be computed (with high probability for every 0/1 input) by a small-degree probabilistic polynomial over $\mathbf{Z}_p$. Thus, in the circuit world, a constant number of levels of mod p gates, intertwined with AND and OR gates, can be collapsed to a single small-degree probabilistic polynomial. The heart of the argument is that the OR function can be very well approximated by a small-degree probabilistic polynomial over $\mathbf{Z}_p$.

In this paper, we show that these collapsing results carry over to the world of proof systems. We show that any $\mathrm{ACC}^0[p]$ proof can be transformed into a quasipolynomial-size, depth-3 proof, with only one level of mod p connectives. Our result can be viewed as showing that the collapsing theorem for $\mathrm{ACC}^0[p]$ is highly constructive, since it can be formalized in the same proof system as the circuit class itself. In formal terms, the collapsing theorem is formalizable in a very restricted version of S^1_2.

The remainder of this paper is organized as follows. In section 2 we define our proof system. In Section 3 we give the general overview of the main theorem. In Section 4 we define the probabilistic polynomials used in the construction of the main theorem. In Sections 5, 6, and 7 we prove our main result, in the case of $\mathrm{ACC}^0[2]$ proofs. In Section 8 we generalize our result to $\mathrm{ACC}^0[p^k]$ proofs. In Section 9 we discuss possible extensions to ACC^0 proofs. Finally in Section 10 we conclude with a discussion of how this theorem suggests a method for proving lower bounds for $\mathrm{ACC}^0[p]$ proofs, as well as a method for obtaining a deterministic algorithm for propositional theorem proving.

2. A logic system with threshold and modular connectives

In this section, we describe a logic system with mod 2, negation, AND and OR connectives as well as threshold connectives. (The generalization of our results to mod p connectives, for p prime, will be discussed in Section 8.) Our system is an extension of the system PTK introduced by Buss and Clote [**BC96**, Section 10]. In their system, only threshold and negation connectives are allowed.

DEFINITION 2.1. Formula *depth* and *size* are defined inductively by:

1. A propositional variable x_i, $i \in \mathbf{N}$, is a formula of depth 0 and size 1.
2. If A is a formula then $\neg A$ is a formula of depth $1 + \mathrm{depth}(A)$ and size $1 + \mathrm{size}(A)$.
3. If $A_1, \ldots, A_n$ are formulas, $n \geq 0$, then $\wedge(A_1, \ldots, A_n)$ is a formula of depth $1 + \max\{\mathrm{depth}(A_i) : 1 \leq i \leq n\}$ and size $n + 1 + \sum_{1 \leq i \leq n} \mathrm{size}(A_i)$. We interpret $\wedge(A_1, \ldots, A_n)$ to be true if and only if all of the A_i's are true.
4. If $A_1, \ldots, A_n$ are formulas, $n \geq 0$, then $\vee(A_1, \ldots, A_n)$ is a formula of depth $1 + \max\{\mathrm{depth}(A_i) : 1 \leq i \leq n\}$ and size $n + 1 + \sum_{1 \leq i \leq n} \mathrm{size}(A_i)$. We interpret $\vee(A_1, \ldots, A_n)$ to be true if and only if one of the A_i's is true.
5. If $A_1, \ldots, A_n$ are formulas, $j = 0, 1$, $n \geq 0$, then $\oplus_j(A_1, \ldots, A_n)$ is a formula of depth $1 + \max\{\mathrm{depth}(A_i) : 1 \leq i \leq n\}$ and size $n + 1 + \sum_{1 \leq i \leq n} \mathrm{size}(A_i)$. We interpret $\oplus_j(A_1, \ldots, A_n)$ to be true if and only if the number of true A_i's is equal to j mod 2.
6. If $A_1, \ldots, A_n$ are formulas, $n, k \geq 0$, then $\mathrm{Th}_k(A_1, \ldots, A_n)$ is a formula of depth $1 + \max\{\mathrm{depth}(A_i) : 1 \leq i \leq n\}$ and size $(n+k) + 1 + \sum_{1 \leq i \leq k} \mathrm{size}(A_i)$. We interpret $\mathrm{Th}_k(A_1, \ldots, A_n)$ to be true if and only if the number of true A_i's is greater than or equal to k. In the following sections, we will use the more descriptive notation $\sum_{i=1}^{m} A_i \geq k$ to represent $\mathrm{Th}_k(A_1, \ldots, A_n)$.

In the above definitions, $\wedge(A_1, \ldots, A_n)$ denotes the logical AND of the multiset consisting of $A_1, \ldots A_n$, and similarly for $\vee$, $\oplus_j$ and Th_k. Thus commutativity of the connectives is implicit. A *cedent* is any sequence $A_1, \ldots, A_n$ of formulas separated by commas. Cedents will usually be designated by capital Greek letters such as Γ and Δ. A *sequent* is given by $\Gamma \rightarrow \Delta$, where Γ, Δ are arbitrary cedents. The size of a cedent $A_1, \ldots, A_n$ is $\sum_{1 \leq i \leq n} \mathrm{size}(A_i)$ and its depth is $\max_{1 \leq i \leq n}(\mathrm{depth}(A_i))$. The size of a sequent $\Gamma \rightarrow \Delta$ is $\mathrm{size}(\Gamma) + \mathrm{size}(\Delta)$ and its depth is $\max(\mathrm{depth}(\Gamma), \mathrm{depth}(\Delta))$. The intended interpretation of the sequent $\Gamma \rightarrow \Delta$ is that the conjunction of the formulas in Γ implies the disjunction of the formulas in Δ.

A proof of a sequent S in our logic system is a sequence of sequents, $S_1, ..., S_q$, such that each sequent S_i is either an initial sequent, or follows from previous sequents by one of the rules of inference, and the final sequent, S_q, is S. The size of the proof is $\sum_{1 \leq i \leq q} \mathrm{size}(S_i)$ and its depth is $\max_{1 \leq i \leq q}(\mathrm{depth}(S_i))$.

An *initial sequent* is of the following form:

1. $A \rightarrow A$ where A is any formula
2. $\rightarrow \wedge() \; ; \; \vee() \rightarrow$
3. $\oplus_1() \rightarrow \; ; \; \rightarrow \oplus_0()$
4. $\mathrm{Th}_k() \rightarrow$ for $k \geq 1$; $\rightarrow \mathrm{Th}_0(A_1, \ldots, A_n)$ for $n \geq 0$

The rules of inference are given by Table 1. Note that the logical rules are defined for $n \geq 1$ and $k \geq 1$.

structural rules

weak left: $\dfrac{\Gamma, \Delta \to \Gamma'}{\Gamma, A, \Delta \to \Gamma'}$ weak right: $\dfrac{\Gamma \to \Gamma', \Delta'}{\Gamma \to \Gamma', A, \Delta'}$

contract left: $\dfrac{\Gamma, A, A, \Delta \to \Gamma'}{\Gamma, A, \Delta \to \Gamma'}$ contract right: $\dfrac{\Gamma \to \Gamma', A, A, \Delta'}{\Gamma \to \Gamma', A, \Delta'}$

permute left: $\dfrac{\Gamma, A, B, \Delta \to \Gamma'}{\Gamma, B, A, \Delta \to \Gamma'}$ permute right: $\dfrac{\Gamma \to \Gamma', A, B, \Delta'}{\Gamma \to \Gamma', B, A, \Delta'}$

cut rule

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

logical rules

$\neg$-left: $\dfrac{\Gamma \to A, \Delta}{\neg A, \Gamma \to \Delta}$ $\neg$-right: $\dfrac{A, \Gamma \to \Delta}{\Gamma \to \neg A, \Delta}$

$\wedge$-left: $$\frac{A_1, \wedge(A_2, \ldots, A_n), \Gamma \to \Delta}{\wedge(A_1, \ldots, A_n), \Gamma \to \Delta}$$

$\wedge$-right: $$\frac{\Gamma \to A_1, \Delta \qquad \Gamma \to \wedge(A_2, \ldots, A_n), \Delta}{\Gamma \to \wedge(A_1, \ldots, A_n), \Delta}$$

$\vee$-left: $$\frac{A_1, \Gamma \to \Delta \qquad \vee(A_2, \ldots, A_n), \Gamma \to \Delta}{\vee(A_1, \ldots, A_n), \Gamma \to \Delta}$$

$\vee$-right: $$\frac{\Gamma \to A_1, \vee(A_2, \ldots, A_n), \Delta}{\Gamma \to \vee(A_1, \ldots, A_n), \Delta}$$

$\oplus$-left: $$\frac{A_1, \oplus_{i-1}(A_2, \ldots, A_n), \Gamma \to \Delta \qquad \oplus_i(A_2, \ldots, A_n), \Gamma \to A_1, \Delta}{\oplus_i(A_1, \ldots, A_n), \Gamma \to \Delta}$$

$\oplus$-right: $$\frac{A_1, \Gamma \to \oplus_{i-1}(A_2, \ldots, A_n), \Delta \qquad \Gamma \to A_1, \oplus_i(A_2, \ldots, A_n), \Delta}{\Gamma \to \oplus_i(A_1, \ldots, A_n), \Delta}$$

Th_k-left: $$\frac{\mathrm{Th}_k(A_2, \ldots, A_n), \Gamma \to \Delta \qquad A_1, \mathrm{Th}_{k-1}(A_2, \ldots, A_n), \Gamma \to \Delta}{\mathrm{Th}_k(A_1, \ldots, A_n), \Gamma \to \Delta}$$

Th_k-right: $$\frac{\Gamma \to A_1, \mathrm{Th}_k(A_2, \ldots, A_n), \Delta \qquad \Gamma \to \mathrm{Th}_{k-1}(A_2, \ldots, A_n), \Delta}{\Gamma \to \mathrm{Th}_k(A_1, \ldots, A_n), \Delta}$$

TABLE 1. Rules of inference

THEOREM 2.2. *Our proof system is sound and complete.*

PROOF. (Proof sketch.) A *truth assignment* is a mapping $\nu : \{x_i : i \in \mathbf{N}\} \to \{0, 1\}$. By induction on formula depth, it is clear how to extend ν to assign a truth value to every formula and sequent of our logic system. A sequent is *valid* if it is true for every truth assignment. By induction on the number of inferences

in proofs, it is straightforward to show that every theorem of our system is valid. Therefore, our system is sound.

Our proof of completeness follows that of Buss and Clote [BC96]. The central idea is as follows. We will say that an inference rule of the form "S_1 and S_2 derive S_3" has the *inversion property* if for every truth assignment ν, $\nu(S_3) = 1$ implies $\nu(S_2) = 1$ and also $\nu(S_2) = 1$. In other words, soundness holds in both directions. It is not hard to check that all of the logical proof rules of our system satisfy the inversion property.

In order to prove completeness, suppose that $\Gamma \to \Delta$ is valid. We will construct a cut-free proof of $\Gamma \to \Delta$ in a top-down fashion, by applying our logical rules so as to "break up" formulas in Γ and Δ into smaller formulas. As long as there is a formula in either Γ or Δ that is not atomic, this is possible since we have logical rules which allow us to break down every connective occurring on either the left or the right side of a sequent. When we can proceed no further, we are left with leaf sequents in which every formula is atomic. The final step is to show that if $\Gamma \to \Delta$ is a leaf sequent, then it is either an initial sequent, or it can be obtained by applying weakening to an initial sequent. This final step follows directly from the inversion property: Since the original sequent was a tautology, by the inversion property, all leaf sequents must also be tautologies, and therefore can be shown to be of the right form. □

DEFINITION 2.3. Let $F = \{(\Gamma_n \to \Delta_n) : n \in N\}$ be a family of sequents in which all formulas involve only the connectives $\neg$, $\wedge$, $\vee$ and $\oplus$. Then $\{R_n : n \in \mathbf{N}\}$ is a family of $\mathrm{ACC}^0[2]$ proofs for F if there exist constants c and d such that the following conditions hold: (1) Each R_n is a valid proof of $(\Gamma_n \to \Delta_n)$ in our system, and furthermore involves only the connectives $\neg$, $\wedge$, $\vee$ and $\oplus$; (2) For all i, the depth of R_n is at most d; and (3) For all n, the size of R_n is at most $(\mathrm{size}(\Gamma_n \to \Delta_n))^c$.

We say that a formula f can be *arranged into d levels* if the connectives of f can be arranged into d groups $L_1, \ldots, L_d$ called levels such that all the inputs of every connective at some level are either propositional variables or connectives from the previous levels. Note that f can be arranged into d levels if and only if f has depth at most d. Moreover, if f has depth less than d, then some of the levels may be empty.

DEFINITION 2.4. Let $\Gamma \to \Delta$ be a sequent in which all formulas involve only the connectives $\neg$, $\wedge$, $\vee$ and $\oplus$. Then R is a size-s *flat* proof of $\Gamma \to \Delta$ if R is of size at most s and every formula in R can be arranged into three levels such that level 3 contains only $\wedge$, $\vee$ and threshold connectives, level 2 contains only $\oplus$ connectives, and level 1 contains only $\wedge$ connectives of fan-in $\log s$. If $F = \{(\Gamma_n \to \Delta_n) : n \in \mathbf{N}\}$ is a family of sequents in which all formulas involve only the connectives $\neg$, $\wedge$, $\vee$ and $\oplus$, then $\{R_n : n \in \mathbf{N}\}$ is a family of size-$s(n)$ *flat* proofs for F if, for every n, R_n is a size-$s(n)$ flat proof of $\Gamma_n \to \Delta_n$.

In the sections that follow, polynomials over $\mathbf{Z}_2$ in the propositional variables $x_1, \ldots, x_n$ will play an important role. In fact, most of the formulas in this article will be statements about such polynomials. To this end, we will adopt the following conventions. First, the symbols $\oplus$ and $\bigoplus$ will be used to denote addition of polynomials over $\mathbf{Z}_2$. They are to be distinguished from $\oplus_0$ and $\oplus_1$ which represent the parity connectives. Second, when written as part of a formula, a polynomial u will be interpreted as the natural formula expressing the fact that $u \equiv 1 \pmod 2$.

For example, $x_1 \oplus x_2x_3$ is interpreted as $\oplus_1(x_1, \wedge(x_2, x_3))$, and $1 \oplus x_1 \oplus x_2x_3$, as $\oplus_0(x_1, \wedge(x_2, x_3))$. Note that we always assume that polynomials are written in standard form, i.e., as sums of monomials.

Throughout this article, we will establish that particular sequents can be derived in our logic system by quasipolynomial-size flat proofs. This will not be done by writing detailed proofs of these sequents. Instead, we will outline how the proof should go, indicating the main steps, and leave to the reader the tedious but straightforward task of filling in the details. These missing details will usually amount to the proof of some simple fact about formulas with threshold and parity connectives. We will also leave to the reader the simple task of verifying the claimed bounds on the complexity of the proofs.

3. Overview of the simulation

Let $(\Gamma_1 \to \Delta_1), (\Gamma_2 \to \Delta_2), \ldots, (\Gamma_L \to \Delta_L)$ be a depth-d, size-s $\mathrm{ACC}^0[2]$ proof in the propositional variables $x_1, \ldots, x_n$. We assume that $s \in 2^{(\log n)^{O(d)}}$. This proof will be fixed throughout Sections 3, 4 and 5. Allender [All89] (see also [AH94]) has shown that

THEOREM 3.1 (Allender [All89]). *Every* $\mathrm{ACC}^0[2]$ *circuit can be simulated by a depth-three, quasipolynomial-size circuit consisting of an unweighted threshold gate at the output,* MOD_2 *gates on level two, and* AND *gates of polylog fan-in on level one.*

By viewing the propositional variables as inputs, every formula A in the $\mathrm{ACC}^0[2]$ proof can be viewed as an $\mathrm{ACC}^0[2]$ circuit. Define $\mathrm{tr}(A)$, the *translation* of A, to be a depth-three circuit that simulates A, according to the theorem. If $\Gamma = (A_1, \ldots, A_m)$, then let $\mathrm{tr}(\Gamma) = (\mathrm{tr}(A_1), \ldots, \mathrm{tr}(A_m))$. Since for every formula A, $\mathrm{tr}(A)$ and A compute the same function, it follows that $\mathrm{tr}(\Gamma_i) \to \mathrm{tr}(\Delta_i)$ is valid if and only if $\Gamma_i \to \Delta_i$ is valid.

Now replace every $\Gamma_i \to \Delta_i$ in the proof by its translation $\mathrm{tr}(\Gamma_i) \to \mathrm{tr}(\Delta_i)$. The mere fact that $\mathrm{tr}(\Gamma_i) \to \mathrm{tr}(\Delta_i)$ is valid exactly when $\Gamma_i \to \Delta_i$ is valid does not imply that $(\mathrm{tr}(\Gamma_1) \to \mathrm{tr}(\Delta_1)), \ldots, (\mathrm{tr}(\Gamma_L) \to \mathrm{tr}(\Delta_L))$ is a proof. However, we will show that this sequence constitutes the skeleton of a proof that simulates the original $\mathrm{ACC}^0[2]$ proof. We will show that whenever a rule is used to infer $\Gamma_i \to \Delta_i$ from $\Gamma_j \to \Delta_j$ and $\Gamma_k \to \Delta_k$, $i > j \geq k$, then $\mathrm{tr}(\Gamma_i) \to \mathrm{tr}(\Delta_i)$ can be derived from $\mathrm{tr}(\Gamma_j) \to \mathrm{tr}(\Delta_j)$ and $\mathrm{tr}(\Gamma_k) \to \mathrm{tr}(\Delta_k)$. Then, by putting together all these derivations, and by appending a derivation of $\Gamma_L \to \Delta_L$ from $\mathrm{tr}(\Gamma_L) \to \mathrm{tr}(\Delta_L)$, we will obtain a proof that simulates the original $\mathrm{ACC}^0[2]$ proof. As for the complexity of this new proof, we will show that all the formulas it contains are of depth three, quasipolynomial size, and of the form threshold of MOD_2's of AND's of polylog fan-in, except for those in the derivation of $\Gamma_L \to \Delta_L$ from $\mathrm{tr}(\Gamma_L) \to \mathrm{tr}(\Delta_L)$. In this last derivation, subformulas of $\Gamma_L \to \Delta_L$ will also appear. Therefore, in the case where $\Gamma_L \to \Delta_L$ is simply a DNF formula $\to C_1, \ldots, C_m$, the simulating proof is a quasipolynomial-size flat proof.

There are many proofs of Theorem 3.1 and, consequently, many possible definitions for $\mathrm{tr}(A)$. But all these definitions follow the same pattern. First, a low-degree *probabilistic polynomial* Q_* is associated with every connective $*$ occurring in our $\mathrm{ACC}^0[2]$ proof. Precise definitions will be given in the next section; for the moment, we will only say that Q_* computes the output of the connective $*$ with high probability. Then, the polynomials associated with all the connectives occurring in a formula A are composed, according to the structure of A, to yield a probabilistic polynomial P_A that computes the value of A with high probability. Finally, the formula $\mathrm{tr}(A)$ is defined as the majority, over all the possible random choices, of the probabilistic polynomial P_A. This formula has the same value as A since A is true precisely when P_A is 1 with high probability.

In the following sections, we will define the polynomials Q_* associated with the various connectives, the polynomials P_A and the formulas $\mathrm{tr}(A)$. We will then establish that several key properties of these polynomials and of $\mathrm{tr}(A)$ have depth-three, quasipolynomial-size proofs of the required form. Finally, these properties will be used to show that every inference rule used in an $\mathrm{ACC}^0[2]$ proof can be simulated by a flat proof in our logic system.

4. Probabilistic polynomials

A *probabilistic polynomial* is a polynomial in two sets of variables: ordinary variables and *probabilistic variables*. Such a polynomial P, with ordinary variables $u = u_1, \ldots, u_m$ and probabilistic variables $v = v_1, \ldots, v_r$, computes a function f with error ε if, for every value of u, $\mathrm{Prob}_v[P(u, v) \neq f(u)] < \varepsilon$. The probabilistic variables are assigned uniformly and independently chosen values from $\{0, 1\}$.

Let $\varepsilon = 1/2^{\lceil 2+\log s \rceil}$. That is, ε is the inverse of the smallest power of 2 greater than or equal to $4s$. For every $m \in [s]$ and every connective $*$ of fan-in m occurring in our $\mathrm{ACC}^0[2]$ proof, we define a probabilistic polynomial Q^m_* over $\mathbf{Z}_2$ that computes the connective with error ε.

Given a connective of fan-in m, let $u = (u_1, \ldots, u_m)$ denote its terms. The polynomial associated with the $\neg$ connective is simply $Q_\neg(u_1) = u_1 \oplus 1$. For the mod 2 connectives, we have $Q^m_{\oplus_1}(u) = \bigoplus_{i=1}^m u_i$ and $Q^m_{\oplus_0}(u) = (\bigoplus_{i=1}^m u_i) \oplus 1$. Recall that all these are polynomials over $\mathbf{Z}_2$.

Consider now an $\vee$ of fan-in m. The existence of a low-degree probabilistic polynomial $Q^m_\vee$ that computes the $\vee$ of m variables with error ε is the key step in the proof of Theorem 3.1. Many constructions are possible, each yielding a different proof of this circuit simulation. The definition that seems to be the most appropriate for our purposes, because of the complexity of the associated proofs, is in terms of universal hashing.

Let $S = \{i : u_i = 1\}$. We want $Q^m_\vee$ to determine, with high probability, if $|S| = 0$ or if $|S| \geq 1$. Now either $|S| = 0$ or $\frac{1}{8}2^k \leq |S| < \frac{1}{4}2^k$, for some $k \in \{3, \ldots, \lceil \log m \rceil + 3\}$. For every such k, let $\mathcal{H}_k$ be a small universal family of hash functions with domain D containing $[m]$ and range $[2^k]$. That is, $\mathcal{H}_k \subseteq \{h : D \to [2^k]\}$, $[m] \subseteq D$, $|\mathcal{H}_k|$ is quasipolynomial in m, and, for every $i \neq j$ in D, the probability that $h(i) = h(j) = r$ is $1/2^{2k}$, when h and r are randomly chosen from $\mathcal{H}_k$ and $[2^k]$. Such families $\mathcal{H}_k$ exist: for example, see [Wig94] for a family of size $2^{2\lceil \log m \rceil}$. For our purposes, the particular choice of $\mathcal{H}_k$ is not important, as long as it is a small universal family of hash functions from D to $[2^k]$. For technical reasons, we will also require that $|\mathcal{H}_k|$ be a power of 2 no smaller than 4.

Such a family $\mathcal{H}_k$ can be viewed as a matrix H_k of dimensions $|\mathcal{H}_k| \times m$ with entries in $[2^k]$. Associate a function from $\mathcal{H}_k$ to each row of H_k, in some arbitrary way, and define $H_k(h, i)$ to be $h(i)$. It is then not difficult to see that the universal property of $\mathcal{H}_k$ can be equivalently stated as follows: given any $i \neq j$ in $[m]$, the

number of rows in which columns i and j contain the same element is equal to $|\mathcal{H}_k|/2^k$.

The idea behind the definition of $Q_\vee^m$ is as follows. For every k, let $R_k = \bigoplus_{i:h(i)=r} u_i$, where the pair h, r is randomly and uniformly chosen from $\mathcal{H}_k \times [2^k]$. If $|S| = 0$, then $R_k = 0$. On the other hand, by using the universal property of the family of hash functions $\mathcal{H}_k$, we will show in the next section that if $\frac{1}{8}2^k \leq |S| < \frac{1}{4}2^k$, then $R_k = 1$ with probability at least $1/16$. This will imply that if $|S| \geq 1$, then, with probability at least $1/16$, $R_k = 1$ for some $k \in \{3, \ldots, \lceil \log m \rceil + 3\}$. This probability will then be amplified to $1-\varepsilon$ by taking $C = \lceil \log \frac{1}{\varepsilon} / \log \frac{16}{15} \rceil$ independent copies.

The polynomial $Q_\vee^m$ will "encode" the above in the following way. Every pair h, r with $h \in \mathcal{H}_k$ and $r \in [2^k]$ can be encoded as a binary string of length $\log|\mathcal{H}_k|+k$. The sequence of probabilistic variables w of $Q_\vee^m$ will be interpreted as a sequence of encodings for pairs $h_{l,k}, r_{l,k}$, $l = 1, \ldots, C$ and $k = 3, \ldots, \lceil \log m \rceil + 3$, where $h_{l,k} \in \mathcal{H}_k$ and $r_{l,k} \in [2^k]$. For every $i \in [m]$ and every l, k, let $F_{l,k}^i$ be a degree $\log|\mathcal{H}_k| + k$ polynomial over $\mathbf{Z}_2$ in the variables encoding $h_{l,k}, r_{l,k}$ such that $F_{l,k}^i$ takes the value 1 if $h_{l,k}(i) = r_{l,k}$, and the value 0, otherwise. Note that when the probabilistic variables are set to specific values, then $F_{l,k}^i$ is simply a constant. In that case, both as a polynomial and as a formula, $\bigoplus_{i=1}^m u_i F_{l,k}^i$ is identical to $R_{l,k} = \bigoplus_{i:h_{l,k}(i)=r_{l,k}} u_i$, the residue modulo 2 of the number of $i \in S$ such that $h_{l,k}(i) = r_{l,k}$. Recall that we always assume that polynomials are written in standard form, i.e., as sums of monomials.

DEFINITION 4.1. The polynomial $Q_\vee^m$ is defined by

$$Q_\vee^m(u, w) = 1 \oplus \prod_{l=1}^{C} \prod_{k=3}^{\lceil \log m \rceil + 3} \left(1 \oplus \bigoplus_i u_i F_{l,k}^i\right).$$

Note that $Q_\vee^m$ has polylog(n) degree and uses polylog(n) probabilistic variables.

If $|S| = 0$, then clearly $Q_\vee^m(u, w) = 0$. Otherwise, if $|S| \geq 1$, then $Q_\vee^m(u, w) = 1$ with probability $1 - \varepsilon$. Therefore, $Q_\vee^m$ does compute $\vee(u)$ with error ε; that is, for every value of u,

$$\mathrm{Prob}_w[Q_\vee^m(u, w) = 1] \begin{cases} \geq 1 - \varepsilon & \text{if } \vee(u) = 1 \\ = 0 & \text{if } \vee(u) = 0 \end{cases}$$

A key step in the proof of our main result, the depth-three simulation of ACC$^0[2]$ proofs, will be to formalize the proof of this property of $Q_\vee^m$, in our logic system, using a depth-three, quasipolynomial-size proof of the form threshold of MOD$_2$'s of AND's of polylog fan-in, i.e., a flat proof.

The polynomial $Q_\wedge^m$ associated with an $\wedge$ of fan-in m is defined from $Q_\vee^m$ as

$$Q_\wedge^m(u, w) = 1 \oplus Q_\vee^m(1 \oplus u_1, \ldots, 1 \oplus u_m, w).$$

In this case, we have that for every value of u,

$$\mathrm{Prob}_w[Q_\wedge^m(u, w) = 1] \begin{cases} = 1 & \text{if } \wedge(u) = 1 \\ \leq \varepsilon & \text{if } \wedge(u) = 0 \end{cases}$$

We now turn to the definition of the probabilistic polynomials P_A that compute, with high probability, the value of the formulas A that appear in our ACC$^0[2]$ proof. Let $y_1, \ldots, y_d$ be d disjoint sequences of probabilistic variables whose length is the maximum number of probabilistic variables required by any of the polynomials Q_*^m,

for $m \in [s]$. Let A be a formula of depth d appearing in the ACC$^0[2]$ proof. The polynomial P_A, with ordinary variables $x = (x_1, \ldots, x_n)$ and probabilistic variables $y_1, \ldots, y_d$, is obtained by replacing each occurrence of a connective $*$ of fan-in m at level i by the corresponding probabilistic polynomial Q_*^m, using probabilistic variables y_i.

DEFINITION 4.2. For $i = 1, \ldots, n$, $P_{x_i} = x_i$. Next, $P_{\wedge()} = 1$, $P_{\vee()} = 0$, $P_{\oplus_1()} = 0$, $P_{\oplus_0()} = 1$. Finally, if $m \geq 1$ and $A = *(A_1, \ldots, A_m)$ is a formula of depth $d \geq 1$, then P_A is the probabilistic polynomial in $x, y_1, \ldots, y_d$ defined by $P_A = Q_*^m(P_{A_1}, \ldots, P_{A_m}, y_d)$.

For example, for

$$A = \vee(\wedge(x_1, x_2), x_3, \vee(x_1, x_4)),$$

we have

$$P_A(x_1, x_2, x_3, x_4, y_1, y_2) = Q_\vee^3(Q_\wedge^2(x_1, x_2, y_1), x_3, Q_\vee^2(x_1, x_4, y_1), y_2).$$

Note that P_A has $(\log n)^{O(d)}$ degree and uses polylog(n) probabilistic variables.

By using the key property of $Q_\vee^m$, it is not difficult to show that for every value of x,

$$\mathrm{Prob}_{y_1,\ldots,y_d}[P_A(x, y_1, \ldots, y_d) = 1] \begin{cases} \geq 1 - \varepsilon s_A & \text{if } A(x) = 1 \\ \leq \varepsilon s_A & \text{if } A(x) = 0 \end{cases}$$

where s_A denotes the size of A. Based on this, we define the *translation* of A, denoted $\mathrm{tr}(A)$, to be the formula that expresses the fact that A is true by saying that the probability that $P_A = 1$ is very high.

DEFINITION 4.3. If A is a propositional variable x_i, $i = 1, \ldots, n$, or one of the formulas $\wedge()$, $\vee()$, $\oplus_1()$ or $\oplus_0()$, then $\mathrm{tr}(A) = A$. If A is any other formula of depth $d \geq 1$, then $\mathrm{tr}(A)$ is the formula

$$\sum_{\sigma_1,\ldots,\sigma_d} P_A(x, \sigma_1, \ldots, \sigma_d) \geq (1 - \varepsilon s_A) 2^{|y_1| + \cdots + |y_d|}$$

where $\sigma_1, \ldots, \sigma_d$ range over all possible values of the sequences of probabilistic variables $y_1, \ldots, y_d$.

Note that $\mathrm{tr}(A)$ is indeed a depth-three formula of size $2^{(\log n)^{O(d)}}$ and of the form threshold of MOD$_2$'s of AND's of fan-in $(\log n)^{O(d)}$. In addition, the threshold in the definition of $\mathrm{tr}(A)$ is an integer since ε is the inverse of a power of 2 and $|y_i| > \log(1/\varepsilon)$, for every i.

5. The key property of $Q_\vee^m$

In this section, we show that the key property of $Q_\vee^m$, namely, that for every value of $u = (u_1, \ldots, u_m)$,

$$\mathrm{Prob}_w[Q_\vee^m(u, w) = 1] \begin{cases} \geq 1 - \varepsilon & \text{if } \vee(u_1, \ldots, u_m) = 1 \\ = 0 & \text{if } \vee(u_1, \ldots, u_m) = 0 \end{cases}$$

has a quasipolynomial-size flat proof in our logic system. Note that in the later applications of this property, the u_i's will not be merely propositional variables. They will be polynomials of degree $(\log n)^{O(d)}$ over $\mathbf{Z}_2$ in the propositional variables $x_1, \ldots, x_n$. Therefore, special care will be necessary to ensure that the resulting proofs are flat proofs.

This property can be proved in many ways. In every case, however, the main step is to establish that if $\frac{1}{8}2^k \leq |S| < \frac{1}{4}2^k$, where $S = \{i : u_i = 1\}$, then, with probability at least $1/16$, $R_k = \bigoplus_{i:h(i)=r} u_i$ is equal to 1. (The actual numbers in the bounds on $|S|$ and in the probability may vary slightly.) Recall that R_k is equal to the residue modulo 2 of the number of $i \in S$ such that $h(i) = r$. Therefore, one way to prove such a lower bound on the probability that $R_k = 1$ is to in fact prove that, with probability at least $1/16$, there is exactly one $i \in S$ such that $h(i) = r$. This in turn can be shown either by an argument similar to the BPP $\subseteq \Sigma_2^p$ argument (see [Sip83]) or by a simple application of the inclusion-exclusion principle, as suggested in [Reg93].

It turns out, however, that such a proof is not appropriate for our purposes. The reason is that it seems to require statements such as "the probability that $u_i = 1$ and, for all $j \neq i$, that $h(j) \neq r$ or $u_j = 0$, is at least $1/2$." Since the u_i are in fact polynomials over $\mathbf{Z}_2$ that compute the connectives from the previous level, this would lead to a depth-four proof of the form threshold of AND's of MOD_2's of small AND's.

To work our way around this problem, we will give a new proof of the lower bound on the probability that $R_k = 1$. Let $C_{h,r}$ denote the number of $i \in S$ such that $h(i) = r$. The idea is to use inclusion-exclusion directly to evaluate the probability that $C_{h,r}$ is odd, instead of the probability that $C_{h,r} = 1$. Since this proof does not seem to have been previously published, we first state and prove the result without worrying about the complexity of the proof.

PROPOSITION 5.1. *If* $\frac{1}{8}2^k \leq |S| < \frac{1}{4}2^k$, *then*

$$\mathrm{Prob}_{h,r\in\mathcal{H}_k\times[2^k]}\left[\left(\bigoplus_{i:h(i)=r} u_i\right) = 1\right] \geq \frac{1}{16}.$$

PROOF. Using the notation introduced earlier, we want to show that $\mathrm{Prob}_{h,r\in\mathcal{H}_k\times[2^k]}[C_{h,r} \text{ is odd}] \geq \frac{1}{16}$. For every $i \in S$, let $B_i = \{h, r : (h(i) = r) \wedge (C_{h,r} \text{ is odd})\}$. Let $B = \cup_{i\in S} B_i$ and let $V = \{h, r : C_{h,r} \text{ is odd}\}$. Then, by the principle of inclusion-exclusion, we have that

$$|V| \geq |V - B| + \sum_{i\in S} |B_i| - \sum_{i,j\in S, i<j} |B_i \cap B_j|.$$

Dividing by $|\mathcal{H}_k \times [2^k]|$, we get that

$$\begin{aligned}\mathrm{Prob}_{h,r}[C_{h,r} \text{ is odd}] \;\geq\; & \sum_{i\in S} \mathrm{Prob}_{h,r}[(h(i) = r) \wedge (C_{h,r} \text{ is odd})] \\ & - \sum_{i,j\in S, i<j} \mathrm{Prob}_{h,r}[(h(i) = h(j) = r) \wedge (C_{h,r} \text{ is odd})].\end{aligned}$$

The probability in the second term is at most $\mathrm{Prob}_{h,r}[h(i) = h(j) = r]$, and this is bounded above by $1/2^{2k}$ because $\mathcal{H}_k$ is a universal family of hash functions. As for the first term, for every $i \in S$, we have that

$$\begin{aligned}&\mathrm{Prob}_{h,r}[(h(i) = r) \wedge (C_{h,r} \text{ is odd})] \\ &= \mathrm{Prob}_{h,r}[(h(i) = r) \wedge (|\{j \in S - \{i\} : h(j) = r\}| \text{ is even})] \\ &= \mathrm{Prob}_{h,r}[h(i) = r] - \mathrm{Prob}_{h,r}[(h(i) = r) \wedge (|\{j \in S - \{i\} : h(j) = r\}| \text{ is odd})] \\ &\geq \mathrm{Prob}_{h,r}[h(i) = r] - \mathrm{Prob}_{h,r}[(\exists j \in S - \{i\})(h(i) = h(j) = r)] \\ &\geq \mathrm{Prob}_{h,r}[h(i) = r] - \sum_{j\in S-i} \mathrm{Prob}_{h,r}[h(i) = h(j) = r].\end{aligned}$$

The first term is easily seen to be equal to $1/2^k$. Therefore, by using once again the universal property of $\mathcal{H}_k$, we get that

$$\begin{aligned}\mathrm{Prob}_{h,r}[C_{h,r} \text{ is odd}] \;&\geq\; \sum_{i\in S}\left(\frac{1}{2^k} - \sum_{j\in S-i}\frac{1}{2^{2k}}\right) - \sum_{i,j\in S, i<j}\frac{1}{2^{2k}} \\ &\geq\; |S|\frac{2^k - |S|}{2^{2k}} - \frac{\frac{1}{2}|S|^2}{2^{2k}} \\ &\geq\; \frac{1}{8}\left(1 - \frac{1}{4}\right) - \frac{1}{32} \\ &=\; \frac{1}{16}.\end{aligned}$$

□

We now proceed to show that this proof can be formalized by a flat proof. First, we prove a lemma typical of the kind of basic facts about formulas with threshold and parity connectives that will be used in the subsequent proofs.

LEMMA 5.2. *Let P and Q be arbitrary polynomials of degree $(\log n)^{O(d)}$ over $\mathbf{Z}_2$ in the propositional variables $x_1, \ldots, x_n$. Then the sequents $P, Q \to PQ$, $PQ \to P$ and $PQ \to Q$ have flat proofs of size $2^{(\log n)^{O(d)}}$.*

PROOF. Suppose that $P = \sum_{i=1}^{s} A_i + a$ and $Q = \sum_{j=1}^{t} B_j + b$, where the A_i's are distinct monomials in the variables $x_1, \ldots, x_n$, $a \in \mathbf{Z}_2$ and similarly for Q. Recall that in a sequent such as $P, Q \to PQ$, P stands for the formula $\oplus_{1-a}(A_1, \ldots, A_s)$, where A_i denotes the conjunction of all the factors in the monomial A_i. Similarly, Q stands for $\oplus_{1-b}(B_1, \ldots, B_t)$.

The polynomial $PQ = \sum_{i=1}^{s}\sum_{j=1}^{t} A_iB_j + \sum_{j=1}^{t} aB_j + \sum_{i=1}^{s} bA_i + ab$, on the other hand, does not stand for the formula

$$\oplus_{1-ab}(A_1B_1, \ldots, A_sB_t, aB_1, \ldots, aB_t, bA_1, \ldots, bA_s),$$

where A_iB_j denotes the conjunction of all the factors in A_i and B_j, aB_j denotes B_j repeated a times and similarly for bA_i. Instead, PQ must first be written in standard form, i.e., as a sum of distinct monomials. Nevertheless, it is easy to prove that

$$PQ \;\to\; \oplus_{1-ab}(A_1B_1, \ldots, A_sB_t, aB_1, \ldots, aB_t, bA_1, \ldots, bA_s)$$

and vice-versa.

Therefore, to prove that $P, Q \to PQ$, it is sufficient to show that

$$\begin{aligned}(5.1)\qquad &\oplus_{1-a}(A_1, \ldots, A_s),\ \oplus_{1-b}(B_1, \ldots, B_t) \\ &\to\; \oplus_{1-ab}(A_1B_1, \ldots, A_sB_t, aB_1, \ldots, aB_t, bA_1, \ldots, bA_s)\end{aligned}$$

and to prove $PQ \rightarrowtail P$, it is sufficient to show that

$$\oplus_{1-ab}(A_1B_1,\dots,A_sB_t,aB_1,\dots,aB_t,bA_1,\dots,bA_s) \rightarrow \oplus_{1-a}(A_1,\dots,A_s). \tag{5.2}$$

To prove both of these sequents, we will first show that for every c and d, the sequent

$$\oplus_c(A_1,\dots,A_s),\ \oplus_d(B_1,\dots,B_t) \rightarrow \oplus_{cd}(A_1B_1,\dots,A_sB_t) \tag{5.3}$$

has a quasipolynomial-size flat proof. We do this by induction on s.

In what follows, we will make frequent use of several basic facts about parity formulas, such as

$$\oplus_c(A_1,\dots,A_s),\ \oplus_d(B_1,\dots,B_t) \rightarrow \oplus_{c+d}(A_1,\dots,A_s,B_1,\dots,B_t),$$

$\rightarrow \oplus_0(A_i), \oplus_1(A_i)$ and $A_i \rightarrow \oplus_1(A_i)$.

For the base case, $s = 0$, we must prove that $\oplus_c(),\ \oplus_d(B_1,\dots,B_t) \rightarrow \oplus_{cd}()$. This sequent follows from $\oplus_c() \rightarrow$ if $c = 1$ and from $\rightarrow \oplus_{cd}()$ if $c = 0$.

Now assume that for every c' and d' we have a proof for

$$\oplus_{c'}(A_1,\dots,A_{s-1}),\ \oplus_{d'}(B_1,\dots,B_t) \rightarrow \oplus_{c'd'}(A_1B_1,\dots,A_{s-1}B_t).$$

A proof for Sequent (5.3) is outlined below.

1. $\oplus_c(A_1,\dots,A_s), A_s \rightarrow \oplus_{c-1}(A_1,\dots,A_{s-1})$
2. $\oplus_{c-1}(A_1,\dots,A_{s-1}), \oplus_d(B_1,\dots,B_t) \rightarrow \oplus_{(c-1)d}(A_1B_1,\dots,A_{s-1}B_t)$
3. $A_s, \oplus_d(B_1,\dots,B_t) \rightarrow \oplus_d(A_sB_1,\dots,A_sB_t)$
4. $\oplus_{(c-1)d}(A_1B_1,\dots,A_{s-1}B_t), \oplus_d(A_sB_1,\dots,A_sB_t) \rightarrow \oplus_{cd}(A_1B_1,\dots,A_sB_t)$
5. $\oplus_c(A_1,\dots,A_s), \oplus_d(B_1,\dots,B_t), A_s \rightarrow \oplus_{cd}(A_1B_1,\dots,A_sB_t)$

Lines 1 and 4 are simple basic facts. Line 2 is from the inductive hypothesis. The existence of a proof for Line 3 can be established by induction on t. Line 5 follows from the previous ones by three applications of the cut rule.

6. $\oplus_c(A_1,\dots,A_s) \rightarrow A_s, \oplus_c(A_1,\dots,A_{s-1})$
7. $\oplus_c(A_1,\dots,A_{s-1}), \oplus_d(B_1,\dots,B_t) \rightarrow \oplus_{cd}(A_1B_1,\dots,A_{s-1}B_t)$
8. $\rightarrow A_s, \oplus_0(A_sB_1,\dots,A_sB_t)$
9. $\oplus_{cd}(A_1B_1,\dots,A_{s-1}B_t), \oplus_0(A_sB_1,\dots,A_sB_t) \rightarrow \oplus_{cd}(A_1B_1,\dots,A_sB_t)$
10. $\oplus_c(A_1,\dots,A_s), \oplus_d(B_1,\dots,B_t) \rightarrow A_s, \oplus_{cd}(A_1B_1,\dots,A_sB_t)$

Lines 6 and 9 are simple basic facts. Line 7 is from the inductive hypothesis. The existence of a proof for Line 8 can be established by induction on t. Line 10 follows from Lines 6 to 9 by three applications of the cut rule. Sequent (5.3) now follows from Lines 5 and 10 by the cut rule.

We now prove Sequents (5.1) and (5.2) from Sequent (5.3). Sequent (5.1) is proved as follows.

1. $\oplus_{1-a}(A_1,\dots,A_s), \oplus_{1-b}(B_1,\dots,B_t) \rightarrow \oplus_{(1-a)(1-b)}(A_1B_1,\dots,A_sB_t)$
2. $\oplus_{1-a}(A_1,\dots,A_s), \oplus_{1-b}(B_1,\dots,B_t), \oplus_{(1-a)(1-b)}(A_1B_1,\dots,A_sB_t) \rightarrow \oplus_{1-ab}(A_1B_1,\dots,A_sB_t,aB_1,\dots,aB_t,bA_1,\dots,bA_s)$

Line 1 is from Sequent (5.3). Line 2 is a basic fact. Note that $(1-a)(1-b) + a(1-b) + b(1-a) = 1 - ab$. Sequent (5.1) follows from these two lines by the cut rule.

The idea behind the proof of Sequent (5.2) is to prove the sequent

$$\oplus_{-a}(A_1,\dots,A_s) \rightarrow \oplus_{-ab}(A_1B_1,\dots,A_sB_t,aB_1,\dots,aB_t,bA_1,\dots,bA_s).$$

The proof proceeds as follows.

1. $\rightarrow \oplus_{-b}(B_1,\dots,B_t), \oplus_{1-b}(B_1,\dots,B_t)$
2. $\oplus_{-a}(A_1,\dots,A_s), \oplus_{-b}(B_1,\dots,B_t) \rightarrow \oplus_{ab}(A_1B_1,\dots,A_sB_t)$
3. $\oplus_{-a}(A_1,\dots,A_s), \oplus_{-b}(B_1,\dots,B_t), \oplus_{ab}(A_1B_1,\dots,A_sB_t) \rightarrow \oplus_{-ab}(A_1B_1,\dots,A_sB_t,aB_1,\dots,aB_t,bA_1,\dots,bA_s)$
4. $\oplus_{-a}(A_1,\dots,A_s), \oplus_{-b}(B_1,\dots,B_t) \rightarrow \oplus_{-ab}(A_1B_1,\dots,A_sB_t,aB_1,\dots,aB_t,bA_1,\dots,bA_s)$

Lines 1 and 3 are simple basic facts. For Line 3, note that $ab+a(-b)+b(-a) = -ab$. Line 2 is from Sequent (5.3). Line 4 follows by one application of the cut rule.

5. $\oplus_{-a}(A_1,\dots,A_s), \oplus_{1-b}(B_1,\dots,B_t) \rightarrow \oplus_{-a(1-b)}(A_1B_1,\dots,A_sB_t)$
6. $\oplus_{-a}(A_1,\dots,A_s), \oplus_{1-b}(B_1,\dots,B_t), \oplus_{-a(1-b)}(A_1B_1,\dots,A_sB_t) \rightarrow \oplus_{-ab}(A_1B_1,\dots,A_sB_t,aB_1,\dots,aB_t,bA_1,\dots,bA_s)$
7. $\oplus_{-a}(A_1,\dots,A_s), \oplus_{1-b}(B_1,\dots,B_t) \rightarrow \oplus_{-ab}(A_1B_1,\dots,A_sB_t,aB_1,\dots,aB_t,bA_1,\dots,bA_s)$
8. $\oplus_{-a}(A_1,\dots,A_s) \rightarrow \oplus_{-ab}(A_1B_1,\dots,A_sB_t,aB_1,\dots,aB_t,bA_1,\dots,bA_s)$

Line 5 is from Sequent (5.3). Line 6 is a simple basic fact. Note that $-a(1-b) + a(1-b) + b(-a) = -ab$. Line 7 follows from Lines 5 and 6 by the cut rule. Line 8 follows from Lines 1, 4 and 7 by two applications of the cut rule.

9. $\rightarrow \oplus_{-a}(A_1,\dots,A_s), \oplus_{1-a}(A_1,\dots,A_s)$
10. $\oplus_{-ab}(A_1B_1,\dots,A_sB_t,aB_1,\dots,aB_t,bA_1,\dots,bA_s)$ $\oplus_{1-ab}(A_1B_1,\dots,A_sB_t,aB_1,\dots,aB_t,bA_1,\dots,bA_s) \rightarrow$

Lines 9 and 10 are simple basic facts. Sequent (5.2) follows from Lines 8, 9 and 10 by two applications of the cut rule. □

We now establish the inclusion-exclusion principle that is needed. Let k be an arbitrary number in $\{3,\dots,\lceil \log m \rceil + 3\}$. For every pair h, r in $\mathcal{H}_k \times [2^k]$, let $\varphi_{h,r}$ denote the polynomial $\bigoplus_{i:h(i)=r} u_i$. Then, for every i, let $\varphi^i_{h,r}$ denote $\bigoplus_{j:h(j)=r} u_iu_j$ and, for every $i \neq j$, let $\varphi^{i,j}_{h,r}$ denote $\bigoplus_{l:h(l)=r} u_iu_ju_l$. Notice that $\varphi^i_{h,r}$ and $\varphi^{i,j}_{h,r}$ are equivalent, respectively, to $u_i \wedge \varphi_{h,r}$ and $u_i \wedge u_j \wedge \varphi_{h,r}$.

LEMMA 5.3 (Inclusion-Exclusion). *For every number t, the sequent*

$$\sum_i \left(\sum_{h,r:h(i)=r} \varphi^i_{h,r} \right) \geq t \rightarrow \sum_{h,r} \varphi_{h,r} + \sum_{i<j} \left(\sum_{h,r:h(i)=h(j)=r} \varphi^{i,j}_{h,r} \right) \geq t$$

has a flat proof of size $2^{(\log n)^{O(d)}}$.

PROOF. If $t = 0$ or if $t > m|\mathcal{H}_k|2^k$, then the proof is simple and left to the reader. So suppose that $0 < t \leq m|\mathcal{H}_k|2^k$. The sequent to be proved is the same as

$$\sum_{h,r} \left(\sum_{i:h(i)=r} \varphi^i_{h,r} \right) \geq t \rightarrow \sum_{h,r} \left(\varphi_{h,r} + \sum_{i,j:i<j,h(i)=h(j)=r} \varphi^{i,j}_{h,r} \right) \geq t.$$

First, we show, for every pair h, r and for every number t_1, that

$$\sum_{i:h(i)=r} \varphi^i_{h,r} \geq t_1 \rightarrow \varphi_{h,r} + \sum_{i,j:i<j,h(i)=h(j)=r} \varphi^{i,j}_{h,r} \geq t_1. \tag{5.4}$$

If $t_1 = 0$, then there is nothing to prove. So suppose that $t_1 \geq 1$. Then

$$\sum_{i:h(i)=r} \varphi^i_{h,r} \geq t_1 \rightarrow \varphi_{h,r},$$

since, for every i, $\varphi^i_{h,r} \rightarrow \varphi_{h,r}$. In addition, since $\varphi^i_{h,r} \rightarrow u_i$, we also have that

$$\sum_{i:h(i)=r} \varphi^i_{h,r} \geq t_1 \rightarrow \sum_{i:h(i)=r} u_i \geq t_1.$$

It is not difficult to show that

$$\sum_{i:h(i)=r} u_i \geq t_1 \rightarrow \sum_{i,j:i<j,h(i)=h(j)=r} u_i u_j \geq \binom{t_1}{2}.$$

Now, since $\varphi_{h,r}, u_i u_j \rightarrow \varphi^{i,j}_{h,r}$,

$$\varphi_{h,r}, \sum_{i,j:i<j,h(i)=h(j)=r} u_i u_j \geq \binom{t_1}{2} \rightarrow \sum_{i,j:i<j,h(i)=h(j)=r} \varphi^{i,j}_{h,r} \geq \binom{t_1}{2}.$$

Therefore,

$$\sum_{i:h(i)=r} \varphi^i_{h,r} \geq t_1 \rightarrow \sum_{i,j:i<j,h(i)=h(j)=r} \varphi^{i,j}_{h,r} \geq \binom{t_1}{2}.$$

Note that $\binom{t_1}{2} + 1 \geq t_1$ because $t_1 \geq 1$. Therefore, (5.4) follows from

$$\varphi_{h,r}, \sum_{i,j:i<j,h(i)=h(j)=r} \varphi^{i,j}_{h,r} \geq \binom{t_1}{2} \rightarrow \varphi_{h,r} + \sum_{i,j:i<j,h(i)=h(j)=r} \varphi^{i,j}_{h,r} \geq \binom{t_1}{2} + 1.$$

The proof now concludes with an induction on the number of pairs h, r. Let

$$Y_{h,r} = \sum_{i:h(i)=r} \varphi^i_{h,r}$$

and

$$Z_{h,r} = \varphi_{h,r} + \sum_{i,j:i<j,h(i)=h(j)=r} \varphi^{i,j}_{h,r}.$$

Then (5.4) says that $Y_{h,r} \geq t_1 \rightarrow Z_{h,r} \geq t_1$, and we want to show that

$$\sum_{h,r} Y_{h,r} \geq t \rightarrow \sum_{h,r} Z_{h,r} \geq t.$$

Order the pairs in $\mathcal{H}_k \times [2^k]$ in some arbitrary way. We will show, by induction, that for every pair h', r' and for every number t_2, the sequent

$$\sum_{h,r\leq h',r'} Y_{h,r} \geq t_2 \rightarrow \sum_{h,r\leq h',r'} Z_{h,r} \geq t_2 \tag{5.5}$$

has a flat proof. The inductive basis, i.e., the case when h', r' is the first pair in $\mathcal{H}_k \times [2^k]$, is immediate from (5.4).

For the inductive step, let h', r' be one of the pairs and suppose that (5.5) holds for $h, r < h', r'$. We break the proof of (5.5) into cases according to the value of $Y_{h',r'}$. We will show, for every $t_1 = 1, \ldots, t-1$, that

$$Y_{h',r'} \geq t_1, \sum_{h,r\leq h',r'} Y_{h,r} \geq t \rightarrow Y_{h',r'} \geq t_1 + 1, \sum_{h,r\leq h',r'} Z_{h,r} \geq t. \tag{5.6}$$

This is equivalent in meaning to $(Y_{h',r'} = t_1) \Rightarrow$ (5.5). Then, since

$$\sum_{h,r\leq h',r'} Y_{h,r} \geq t \rightarrow Y_{h',r'} \geq 1, \sum_{h,r<h',r'} Y_{h,r} \geq t$$

and

$$Y_{h',r'} \geq t \rightarrow Z_{h',r'} \geq t,$$

we get that

$$\sum_{h,r\leq h',r'} Y_{h,r} \geq t \rightarrow Y_{h',r'} \geq 1, \sum_{h,r\leq h',r'} Z_{h,r} \geq t$$

and

$$Y_{h',r'} \geq t \rightarrow \sum_{h,r\leq h',r'} Z_{h,r} \geq t,$$

by using the inductive hypothesis and simple facts about threshold connectives. These two sequents correspond to the cases $Y_{h',r'} \leq 0$ and $Y_{h',r'} \geq t$. Therefore, by combining with (5.6), we get a proof of (5.5).

To complete the inductive step, there only remains to prove (5.6). The idea is simple. Suppose that $Y_{h',r'} = t_1$. Then $Z_{h',r'} \geq t_1$, by (5.4). On the other hand, $\sum_{h,r<h',r'} Y_{h,r} \geq t - t_1$, which implies that $\sum_{h,r<h',r'} Z_{h,r} \geq t - t_1$, by the inductive hypothesis. Combining the two we get $\sum_{h,r\leq h',r'} Z_{h,r} \geq t$. In sequent calculus, we have that

$$Y_{h',r'} \geq t_1 \rightarrow Z_{h',r'} \geq t_1$$

and that

$$\sum_{h,r\leq h',r'} Y_{h,r} \geq t \rightarrow Y_{h',r'} \geq t_1 + 1, \sum_{h,r<h',r'} Y_{h,r} \geq t - t_1.$$

The result follows by the inductive hypothesis and since

$$Z_{h',r'} \geq t_1, \sum_{h,r<h',r'} Z_{h,r} \geq t - t_1 \rightarrow \sum_{h,r\leq h',r'} Z_{h,r} \geq t.$$

□

We now show that Proposition 5.1 has a flat proof.

LEMMA 5.4. *The sequent*

$$\sum_i u_i \geq \frac{1}{8}2^k \rightarrow \sum_i u_i \geq \frac{1}{4}2^k, \sum_{h,r\in\mathcal{H}_k\times[2^k]} \varphi_{h,r} \geq \frac{N_k}{16},$$

where $N_k = |\mathcal{H}_k|2^k$, *has a flat proof of size* $2^{(\log n)^{O(d)}}$.

PROOF. Note that the threshold $N_k/16$ is an integer since $k \geq 3$ and $|\mathcal{H}_k|$ is a power of 2 no smaller than 4.

First, we show that for every i,

$$u_i \to \sum_j u_j \geq \frac{1}{4}2^k, \sum_{h,r:h(i)=r} \varphi^i_{h,r} \geq \frac{3}{4}|\mathcal{H}_k|. \tag{5.7}$$

Since, for every h, r such that $h(i) = r$,

$$u_i, \left(1 \oplus \bigoplus_{j:j\neq i, h(j)=r} u_j\right) \to \varphi^i_{h,r}$$

we have that

$$u_i, \sum_{h,r:h(i)=r} \left(1 \oplus \bigoplus_{j:j\neq i,h(j)=r} u_j\right) \geq \frac{3}{4}|\mathcal{H}_k| \to \sum_{h,r:h(i)=r} \varphi^i_{h,r} \geq \frac{3}{4}|\mathcal{H}_k|.$$

Therefore, it is sufficient to show that

$$\to \sum_j u_j \geq \frac{1}{4}2^k, \sum_{h,r:h(i)=r} \left(1 \oplus \bigoplus_{j:j\neq i,h(j)=r} u_j\right) \geq \frac{3}{4}|\mathcal{H}_k|.$$

Let $E^i_{h,r} = \bigoplus_{j:j\neq i,h(j)=r} u_j$. For every h, r,

$$\to E^i_{h,r}, 1 \oplus E^i_{h,r}.$$

Therefore,

$$\to \sum_{h,r:h(i)=r} E^i_{h,r} \geq \frac{1}{4}|\mathcal{H}_k|, \sum_{h,r:h(i)=r} (1 \oplus E^i_{h,r}) \geq \frac{3}{4}|\mathcal{H}_k|.$$

Since $E^i_{h,r} \to \sum_{j:j\neq i,h(j)=r} u_j \geq 1$, we have that

$$\sum_{h,r:h(i)=r} E^i_{h,r} \geq \frac{1}{4}|\mathcal{H}_k| \to \sum_{h,r:h(i)=r} \left(\sum_{j:j\neq i,h(j)=r} u_j\right) \geq \frac{1}{4}|\mathcal{H}_k|.$$

The formula on the right hand side of this sequent is the same as

$$\sum_{j:j\neq i} \left(\sum_{h,r:h(i)=h(j)=r} u_j\right) \geq \frac{1}{4}|\mathcal{H}_k|.$$

By the universal property of the family $\mathcal{H}_k$ of hash functions, the value of the inside sum is either 0 or $|\mathcal{H}_k|/2^k$. From this it follows that

$$\sum_{j:j\neq i} \left(\sum_{h,r:h(i)=h(j)=r} u_j\right) \geq \frac{1}{4}|\mathcal{H}_k| \to \sum_{j:j\neq i} u_j \geq \frac{1}{4}2^k.$$

Therefore,

$$\to \sum_j u_j \geq \frac{1}{4}2^k, \sum_{h,r:h(i)=r} (1 \oplus E^i_{h,r}) \geq \frac{3}{4}|\mathcal{H}_k|,$$

which completes the proof of (5.7).

From (5.7), we get that

$$\sum_i u_i \geq \frac{1}{8}2^k \to \sum_i u_i \geq \frac{1}{4}2^k, \sum_i \left(\sum_{h,r:h(i)=r} \varphi^i_{h,r}\right) \geq \frac{3}{32}N_k.$$

By the principle of inclusion-exclusion (Lemma 5.3), this implies that

$$\sum_i u_i \geq \frac{1}{8}2^k \to \sum_i u_i \geq \frac{1}{4}2^k, \sum_{h,r} \varphi_{h,r} + \sum_{i<j} \left(\sum_{h,r:h(i)=h(j)=r} \varphi^{i,j}_{h,r}\right) \geq \frac{3}{32}N_k.$$

Now,

$$\sum_{h,r} \varphi_{h,r} + \sum_{i<j} \left(\sum_{h,r:h(i)=h(j)=r} \varphi^{i,j}_{h,r}\right) \geq \frac{3}{32}N_k$$
$$\to \sum_{h,r} \varphi_{h,r} \geq \frac{N_k}{16}, \sum_{i<j} \left(\sum_{h,r:h(i)=h(j)=r} \varphi^{i,j}_{h,r}\right) \geq \frac{N_k}{32}.$$

Since $\varphi^{i,j}_{h,r} \to u_i u_j$,

$$\sum_{i<j} \left(\sum_{h,r:h(i)=h(j)=r} \varphi^{i,j}_{h,r}\right) \geq \frac{N_k}{32} \to \sum_{i<j} \left(\sum_{h,r:h(i)=h(j)=r} u_i u_j\right) \geq \frac{N_k}{32}.$$

Again by the universal property of $\mathcal{H}_k$, the value of the inside sum on the right hand side is at most $|\mathcal{H}_k|/2^k$. This implies that

$$\sum_{i<j} \left(\sum_{h,r:h(i)=h(j)=r} u_i u_j\right) \geq \frac{N_k}{32} \to \sum_{i<j} u_i u_j \geq \frac{2^{2k}}{32}.$$

It is not difficult to show that for every t,

$$\sum_{i<j} u_i u_j \geq \frac{t^2}{2} \to \sum_i u_i \geq t.$$

Therefore,

$$\sum_{i<j} u_i u_j \geq \frac{2^{2k}}{32} \to \sum_i u_i \geq \frac{2^k}{4}$$

which completes the proof of the lemma. □

Finally, we arrive at the key property of $Q^m_\vee$.

LEMMA 5.5. *Let $u = (u_1, \ldots, u_m)$ be a sequence of polynomials of degree $(\log n)^{O(d)}$ over $\mathbb{Z}_2$ in the propositional variables $x_1, \ldots, x_n$. The sequents*

$$\sum_\tau Q^m_\vee(u, \tau) \geq 1 \to \sum_i u_i \geq 1$$

and

$$\sum_i u_i \geq 1 \to \sum_\tau Q^m_\vee(u, \tau) \geq (1 - \varepsilon)2^{|\omega|},$$

where τ ranges over all possible values of the probabilistic variables w of $Q^m_\vee$, have flat proofs of size $2^{(\log n)^{O(d)}}$.

PROOF. First, from the sequents of Lemma 5.2, the following basic facts about polynomials over $\mathbf{Z}_2$ can be shown:

$$1 \oplus (1 \oplus P)(1 \oplus Q) \rightarrow P, Q \tag{5.8}$$

$$\sum_{i=1}^{M} P_i \geq K \rightarrow \sum_{i=1}^{M}\sum_{j=1}^{N}(1 \oplus (1 \oplus P_i)Q_j) \geq KN \tag{5.9}$$

$$\begin{aligned}&\textstyle\sum_{i=1}^{M}(1 \oplus P_i) \geq M - K,\ \sum_{i=1}^{N}(1 \oplus Q_i) \geq N - L\\ &\quad\rightarrow \textstyle\sum_{i=1}^{M}\sum_{j=1}^{N}(1 \oplus P_iQ_j) \geq MN - KL\end{aligned} \tag{5.10}$$

where P, Q, the P_i's and the Q_j's are polynomials over $\mathbf{Z}_2$ and K and L are numbers.

Consider the first sequent in the statement of the lemma. To prove this sequent, it is sufficient to show that for every τ, $Q^m_\vee(u,\tau) \rightarrow \bigvee_i u_i$. Consider an arbitrary τ. As was done earlier, let $R_{l,k}$ denote the polynomial $\bigoplus_{i:h_{l,k}(i)=r_{l,k}} u_i$. Then

$$Q^m_\vee(u,\tau) = 1 \oplus \prod_{l=1}^{C}\prod_{k=3}^{L(m)}(1 \oplus R_{l,k}),$$

where $L(m) = \lceil \log m \rceil + 3$. By repeated applications of (5.8), we get that

$$Q^m_\vee(u,\tau) \rightarrow \sum_{l,k} R_{l,k} \geq 1.$$

The first sequent follows since, for every pair l,k, $R_{l,k} \rightarrow \sum_i u_i \geq 1$.

Consider now the second sequent. By Lemma 5.4, we have that for every l,k,

$$\sum_i u_i \geq \frac{1}{8}2^k \rightarrow \sum_i u_i \geq \frac{1}{4}2^k,\ \sum_{h_{l,k},r_{l,k}} R_{l,k} \geq \frac{|\mathcal{H}_k|2^k}{16},$$

where the last sum ranges over all possible values of $h_{l,k}, r_{l,k}$. From (5.9), we get that for every l,k,

$$\begin{aligned}&\sum_{h_{l,k},r_{l,k}} R_{l,k} \geq \frac{|\mathcal{H}_k|2^k}{16}\\ &\rightarrow \sum_{(h_{l,k'},r_{l,k'})_{3\leq k'\leq L(m)}}\left(1 \oplus (1 \oplus R_{l,k})\prod_{k''\neq k}(1 \oplus R_{l,k''})\right) \geq \frac{\prod_{k'=3}^{L(m)}|\mathcal{H}_{k'}|2^{k'}}{16}.\end{aligned}$$

Therefore, for every l,

$$\sum_i u_i \geq 1 \rightarrow \sum_{(h_{l,k},r_{l,k})_{3\leq k\leq L(m)}}\left(1 \oplus \prod_{k'=3}^{L(m)}(1 \oplus R_{l,k'})\right) \geq \frac{\prod_{k=3}^{L(m)}|\mathcal{H}_k|2^k}{16}.$$

Then, by repeated applications of (5.10), we get that

$$\sum_i u_i \geq 1 \rightarrow \sum_{(h_{l,k},r_{l,k})_{1\leq l\leq C,\ 3\leq k\leq L(m)}}\left(1 \oplus \prod_{l=1}^{C}\prod_{k'=3}^{L(m)}(1 \oplus R_{l,k'})\right) \geq (1-\varepsilon)\left(\prod_{k=3}^{L(m)}|\mathcal{H}_k|2^k\right)^C.$$

The result follows the right hand side is precisely $\sum_\tau Q^m_\vee(u,\tau) \geq (1-\varepsilon)2^{|w|}$. □

The key property of $Q^m_\wedge$ is easily derived from the key property of $Q^m_\vee$.

LEMMA 5.6. *Let $u = (u_1,\ldots,u_m)$ be a sequence of polynomials of degree $(\log n)^{O(d)}$ over $\mathbf{Z}_2$ in the propositional variables $x_1,\ldots,x_n$. The sequents*

$$\sum_i u_i \geq m \rightarrow \sum_\tau Q^m_\wedge(u,\tau) \geq 2^{|w|}$$

and

$$\sum_\tau Q^m_\wedge(u,\tau) \geq \varepsilon 2^{|w|} + 1 \rightarrow \sum_i u_i \geq m,$$

where τ ranges over all possible values of the probabilistic variables w of $Q^m_\wedge$, have flat proofs of size $2^{(\log n)^{O(d)}}$.

PROOF. We prove the second sequent. The proof of the first one is similar. By Lemma 5.5, we have that

$$\sum_i(1 \oplus u_i) \geq 1 \rightarrow \sum_\tau Q^m_\vee(1 \oplus u,\tau) \geq (1-\varepsilon)2^{|w|},$$

where $1 \oplus u$ denotes $1 \oplus u_1,\ldots,1 \oplus u_m$. For every i, we have that $\rightarrow u_i, 1 \oplus u_i$. This implies that

$$\rightarrow \sum_i u_i \geq m,\ \sum_i(1 \oplus u_i) \geq 1.$$

On the other hand, since, for every τ, $Q^m_\wedge(u,\tau) = 1 \oplus Q^m_\vee(1 \oplus u,\tau)$, we have that $Q^m_\wedge(u,\tau), Q^m_\vee(1 \oplus u,\tau) \rightarrow$ and that

$$\sum_\tau Q^m_\wedge(u,\tau) \geq \varepsilon 2^{|w|} + 1,\ \sum_\tau Q^m_\vee(1 \oplus u,\tau) \geq (1-\varepsilon)2^{|w|} \rightarrow .$$

The result follows. □

6. Properties of P_A and $\mathrm{tr}(A)$

In this section, we show that several properties of the probabilistic polynomials P_A and of the formulas $\mathrm{tr}(A)$ have flat proofs in our logic system. This will then be used in the next section to show that the rules of inference used in $\mathrm{ACC}^0[2]$ proofs can be simulated by flat proofs.

These properties all have very easy proofs if one is not concerned with the complexity of these proofs. However, some care must be taken in order to obtain flat proofs. For example, the property

$$\sum_\sigma P_A(\sigma) \geq \varepsilon s_A 2^{|y|} + 1 \rightarrow \mathrm{tr}(A),$$

can be proved from the fact that P_A computes A with error ε, a fact that is easily established by using the key property of $Q_\vee^m$ (Lemma 5.5). However, such a proof mentions explicitly the formula A and this, of course, has to be avoided. We will therefore have to prove this and the other properties in a more direct way.

In the following lemmas, by a small $\mathrm{ACC}^0[2]$ formula, we mean a formula over the connectives $\neg$, $\wedge$, $\vee$ and $\oplus$, of size at most $1/(4\varepsilon)$. Recall that in Section 3 we fixed an arbitrary $\mathrm{ACC}^0[2]$ proof of depth d and size s and that in Section 4 we defined ε to be $1/2^{\lceil 2+\log s\rceil}$. We then defined the polynomials P_A and the formulas $\mathrm{tr}(A)$ in terms of ε.

LEMMA 6.1. *For every small* $\mathrm{ACC}^0[2]$ *formula* A *and for every small* $\mathrm{ACC}^0[2]$ *formula of the form* $\oplus_1(A_1,\dots,A_m)$, *the sequents* $\mathrm{tr}(A), \mathrm{tr}(\neg A) \to$ *and*

$$\mathrm{tr}(\oplus_1(A_1,\dots,A_m)),\ \mathrm{tr}(\oplus_0(A_1,\dots,A_m)) \to$$

have flat proofs of size $2^{(\log n)^{O(d)}}$.

PROOF. Suppose that A is a depth-d formula. Associated with A and $\neg A$, we have probabilistic polynomials $P_A(y_1,\dots,y_d)$ and $P_{\neg A}(y_1,\dots,y_{d+1})$. (We are omitting the sequence x of propositional variables.) Let y denote $y_1,\dots,y_d$ and let z denote y_{d+1}, so that we may write $P_A(y)$ for $P_A(y_1,\dots,y_d)$ and $P_{\neg A}(y,z)$ for $P_{\neg A}(y_1,\dots,y_{d+1})$. By definition, the sequent $\mathrm{tr}(A), \mathrm{tr}(\neg A) \to$ says that

$$\sum_\sigma P_A(\sigma) \ge (1-\varepsilon s_A)2^{|y|},\ \sum_{\sigma,\tau} P_{\neg A}(\sigma,\tau) \ge (1-\varepsilon s_{\neg A})2^{|y|+|z|} \to .$$

Recall that $P_{\neg A}(\sigma,\tau) = P_A(\sigma)\oplus 1$. For every σ, we have that $P_A(\sigma), P_A(\sigma)\oplus 1 \to$. Therefore,

$$\mathrm{tr}(A),\ \sum_\sigma (P_A(\sigma)\oplus 1) \ge \varepsilon s_A 2^{|y|} + 1 \to .$$

Since $\sum_\tau (P_A(\sigma)\oplus 1)$ is always at most $2^{|z|}$, and since $\sum_\tau (P_A(\sigma)\oplus 1) \ge 1 \to P_A(\sigma)\oplus 1$, we have that

$$\sum_{\sigma,\tau}(P_A(\sigma)\oplus 1) \ge \varepsilon s_A 2^{|y|+|z|}+1 \to \sum_\sigma (P_A(\sigma)\oplus 1) \ge \varepsilon s_A 2^{|y|}+1.$$

Therefore,

$$\mathrm{tr}(A),\ \sum_{\sigma,\tau} P_{\neg A}(\sigma,\tau) \ge \varepsilon s_A 2^{|y|+|z|}+1 \to .$$

The result follows since $1-\varepsilon s_{\neg A} \ge \varepsilon s_A$.

The second sequent is proved similarly. □

LEMMA 6.2. *For every small* $\mathrm{ACC}^0[2]$ *formula* A *of the form* $\vee(A_1,\dots,A_m)$ *and for every* $i \in [m]$, *the sequents* $\mathrm{tr}(A_i) \to \mathrm{tr}(A)$ *and* $\mathrm{tr}(\bigvee_{j\ne i} A_j) \to \mathrm{tr}(A)$ *have flat proofs of size* $2^{(\log n)^{O(d)}}$.

PROOF. For the first sequent, we have to show that

$$\sum_\sigma P_{A_i}(\sigma) \ge (1-\varepsilon s_{A_i})2^{|y|} \to \sum_\sigma \left(\sum_\tau P_A(\sigma,\tau)\right) \ge (1-\varepsilon s_A)2^{|y|+|z|}.$$

Consider an arbitrary σ. Since $P_A(\sigma,z) = Q_\vee^m(P_{A_1}(\sigma),\dots,P_{A_m}(\sigma),z)$, from Lemma 5.5 we get that

$$P_{A_i}(\sigma) \to \sum_\tau P_A(\sigma,\tau) \ge (1-\varepsilon)2^{|z|}.$$

Therefore,

$$\sum_\sigma P_{A_i}(\sigma) \ge (1-\varepsilon s_{A_i})2^{|y|} \to \sum_\sigma \left(\sum_\tau P_A(\sigma,\tau)\right) \ge (1-\varepsilon s_{A_i})2^{|y|}(1-\varepsilon)2^{|z|},$$

which implies that $\mathrm{tr}(A_i) \to \mathrm{tr}(A)$, since $(1-\varepsilon s_{A_i})(1-\varepsilon) \ge 1-\varepsilon s_A$.

Now let $B = \bigvee_{j\ne i} A_j$. Then $P_B(y,z) = Q_\vee^{m-1}((P_{A_j}(y))_{j\ne i}, z)$. We want to show that

$$\sum_\sigma \left(\sum_\tau P_B(\sigma,\tau)\right) \ge (1-\varepsilon s_B)2^{|y|+|z|} \to \sum_\sigma \left(\sum_\tau P_A(\sigma,\tau)\right) \ge (1-\varepsilon s_A)2^{|y|+|z|}.$$

Consider an arbitrary σ. By Lemma 5.5, we have that

$$\sum_\tau P_B(\sigma,\tau) \ge 1 \to \sum_{j\in S} P_{A_j}(\sigma) \ge 1$$

and that

$$\sum_j P_{A_j}(\sigma) \ge 1 \to \sum_\tau P_A(\sigma,\tau) \ge (1-\varepsilon)2^{|z|}.$$

Therefore,

$$\sum_\tau P_B(\sigma,\tau) \ge 1 \to \sum_\tau P_A(\sigma,\tau) \ge (1-\varepsilon)2^{|z|}.$$

Since the maximum value of $\sum_\tau P_B(\sigma,\tau)$ is $2^{|z|}$, it can be shown that

$$\sum_\sigma \left(\sum_\tau P_B(\sigma,\tau)\right) \ge (1-\varepsilon s_B)2^{|y|+|z|} \to \sum_\sigma \left(\sum_\tau P_A(\sigma,\tau)\right) \ge (1-\varepsilon s_B)2^{|y|}(1-\varepsilon)2^{|z|}.$$

which implies that $\mathrm{tr}(B) \to \mathrm{tr}(A)$, since $(1-\varepsilon s_B)(1-\varepsilon) \ge 1-\varepsilon s_A$. □

LEMMA 6.3. *For every small* $\mathrm{ACC}^0[2]$ *formula of the form* $\oplus_i(A_1,\dots,A_m)$, *the sequent*

$$\mathrm{tr}(A_1), \mathrm{tr}(\oplus_{i-1}(A_2,\dots,A_m)) \to \mathrm{tr}(\oplus_i(A_1,\dots,A_m))$$

has a flat proof of size $2^{(\log n)^{O(d)}}$.

PROOF. Let $A = \oplus_i(A_1,\dots,A_m)$ and $B = \oplus_{i-1}(A_2,\dots,A_m)$. It is easy to see that for every pair σ,τ, $P_{A_1}(\sigma)$, $P_B(\sigma,\tau) \to P_A(\sigma,\tau)$. From this, we get that

$$\sum_{\sigma,\tau} P_{A_1}(\sigma) \ge (1-\varepsilon s_{A_1})2^{|y|+|z|},\ \sum_{\sigma,\tau} P_B(\sigma,\tau) \ge (1-\varepsilon s_B)2^{|y|+|z|} \to \sum_{\sigma,\tau} P_A(\sigma,\tau) \ge (1-\varepsilon s_{A_1}-\varepsilon s_B)2^{|y|+|z|}.$$

The result follows since $s_{A_1} + s_B \leq s_A$ and

$$\sum_{\sigma} P_{A_1}(\sigma) \geq (1 - \varepsilon s_{A_1}) 2^{|y|} \rightarrow \sum_{\sigma,\tau} P_{A_1}(\sigma) \geq (1 - \varepsilon s_{A_1}) 2^{|y|+|z|}.$$

□

LEMMA 6.4. *For every small* $\mathrm{ACC}^0[2]$ *formula* B *of depth* d, *the sequent*

$$\sum_{\sigma_1,\ldots,\sigma_d} P_B(\sigma_1,\ldots,\sigma_d) \geq \varepsilon s_B 2^{|y_1|+\cdots+|y_d|} + 1 \rightarrow \mathrm{tr}(B)$$

has a flat proof of size $2^{(\log n)^{O(d)}}$.

PROOF. We use induction on the *generalized* structure of B. Call A a *generalized* subformula of B if $A = *(B_i,\ldots,B_r)$ where $*(B_1,\ldots,B_r)$ is a subformula of B and $1 \leq i \leq r$. We show that for every generalized subformula A of B, the sequent

$$\sum_{\sigma_1,\ldots,\sigma_d} P_A(\sigma_1,\ldots,\sigma_d) \geq \varepsilon s_A 2^{|y_1|+\cdots+|y_d|} + 1 \rightarrow \mathrm{tr}(A),$$

where d is the depth of A, has a flat proof in our logic system.

Basis. Suppose that $A = *(u_1,\ldots,u_m)$, where all the u_i's are propositional variables.

Case $A = \vee(u_1,\ldots,u_m)$. Then $P_A(y_1) = Q_\vee^m(u_1,\ldots,u_m,y_1)$. The result follows directly from Lemma 5.5.

Case $A = \oplus_1(u_1,\ldots,u_m)$. Here P_A is deterministic, i.e., P_A does not depend on y_1. Therefore, $\sum_{\sigma_1} P_A \geq 1 \rightarrow P_A$ and $P_A \rightarrow \sum_{\sigma_1} P_A \geq 2^{|y_1|}$. The result follows easily.

The other cases are similar.

Inductive step. Now assume that $A = *(A_1,\ldots,A_m)$, where some of the A_i's are not merely propositional variables. Let y denote $y_1,\ldots,y_{d-1}$ and let z denote y_d. Then the sequent we must prove is

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1 \rightarrow \mathrm{tr}(A).$$

Case $A = \neg A_1$. Recall that in this case $P_A(y,z) = P_{A_1}(y) \oplus 1$. For every σ,τ, we have that $\rightarrow P_A(\sigma,\tau), P_{A_1}(\sigma)$. Therefore,

$$\rightarrow \mathrm{tr}(A), \sum_{\sigma}\left(\sum_{\tau} P_{A_1}(\sigma)\right) \geq \varepsilon s_A 2^{|y|+|z|} + 1$$

so that

$$\rightarrow \mathrm{tr}(A), \sum_{\sigma} P_{A_1}(\sigma) \geq \varepsilon s_A 2^{|y|} + 1,$$

since the inside sum is at most $2^{|z|}$. Then, by the inductive hypothesis, we get that $\rightarrow \mathrm{tr}(A), \mathrm{tr}(A_1)$. On the other hand,

$$\sum_{\sigma} (P_{A_1}(\sigma) \oplus 1) \geq \varepsilon s_{A_1} 2^{|y|} + 1, \ \mathrm{tr}(A_1) \rightarrow$$

so that

$$\sum_{\sigma}\left(\sum_{\tau} P_A(\sigma,\tau)\right) \geq \varepsilon s_A 2^{|y|+|z|} + 1, \ \mathrm{tr}(A_1) \rightarrow,$$

since, once again, the inside sum is at most $2^{|z|}$. The result follows.

Case $A = \vee(A_1,\ldots,A_m)$. For every σ, by Lemma 5.5, we have that

$$\sum_{\tau} P_A(\sigma,\tau) \geq 1 \rightarrow \sum_{i} P_{A_i}(\sigma) \geq 1.$$

Therefore,

$$\sum_{\sigma}\left(\sum_{\tau} P_A(\sigma,\tau)\right) \geq \varepsilon s_A 2^{|y|+|z|} + 1 \rightarrow \sum_{\sigma}\left(\sum_{i} P_{A_i}(\sigma)\right) \geq \varepsilon s_A 2^{|y|} + 1,$$

since the value of the inside sum on the left hand side is at most $2^{|z|}$. The formula on the right hand side is the same as

$$\sum_{i}\left(\sum_{\sigma} P_{A_i}(\sigma)\right) \geq \varepsilon s_A 2^{|y|} + 1.$$

For every i, the inductive hypothesis says that

$$\sum_{\sigma} P_{A_i}(\sigma) \geq \varepsilon s_{A_i} 2^{|y|} + 1 \rightarrow \mathrm{tr}(A_i).$$

Therefore, since $\mathrm{tr}(A_i) \rightarrow \mathrm{tr}(A)$ by Lemma 6.2,

$$\sum_{\sigma} P_{A_i}(\sigma) \geq \varepsilon s_{A_i} 2^{|y|} + 1 \rightarrow \mathrm{tr}(A).$$

From this, it is easy to show that

$$\sum_{i}\left(\sum_{\sigma} P_{A_i}(\sigma)\right) \geq \varepsilon s_A 2^{|y|} + 1 \rightarrow \mathrm{tr}(A).$$

The result follows.

Case $A = \wedge(A_1,\ldots,A_m)$. Recall that $P_A(y,z) = 1 \oplus Q_\vee^m(1 \oplus P_{A_1}(y),\ldots,1 \oplus P_{A_m}(y),z)$. For every σ,τ, we have both $P_A(\sigma,\tau), 1 \oplus P_A(\sigma,\tau) \rightarrow$ and its inverse. Therefore,

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1, \ \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq (1 - \varepsilon s_A) 2^{|y|+|z|} \rightarrow$$

and

$$\rightarrow \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq \varepsilon s_A 2^{|y|+|z|} + 1, \ \mathrm{tr}(A).$$

This implies that it is sufficient to show that

$$\sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq \varepsilon s_A 2^{|y|+|z|} + 1 \rightarrow \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq (1 - \varepsilon s_A) 2^{|y|+|z|}.$$

By an argument similar to the one used in the previous case ($A = \vee(A_1,\ldots,A_m)$), it can be shown that

$$\sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq \varepsilon s_A 2^{|y|+|z|} + 1 \rightarrow \sum_{i}\left(\sum_{\sigma}(1 \oplus P_{A_i}(\sigma))\right) \geq \varepsilon s_A 2^{|y|} + 1.$$

For every i, by the inductive hypothesis, we have that

$$\sum_{\sigma} P_{A_i}(\sigma) \geq \varepsilon s_{A_i} 2^{|y|} + 1 \rightarrow \sum_{\sigma} P_{A_i}(\sigma) \geq (1 - \varepsilon s_{A_i}) 2^{|y|}.$$

Therefore,

$$\sum_{\sigma}(1 \oplus P_{A_i}(\sigma)) \geq \varepsilon s_{A_i} 2^{|y|} + 1 \;\rightarrow\; \sum_{\sigma}(1 \oplus P_{A_i}(\sigma)) \geq (1 - \varepsilon s_{A_i}) 2^{|y|}.$$

As in the proof of the sequent $\mathrm{tr}(A_i) \rightarrow \mathrm{tr}(A)$ in Lemma 6.2, it can be shown that

$$\sum_{\sigma}(1 \oplus P_{A_i}(\sigma)) \geq (1 - \varepsilon s_{A_i}) 2^{|y|} \;\rightarrow\; \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq (1 - \varepsilon s_A) 2^{|y|+|z|}.$$

Therefore,

$$\sum_{\sigma}(1 \oplus P_{A_i}(\sigma)) \geq \varepsilon s_{A_i} 2^{|y|} + 1 \;\rightarrow\; \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq (1 - \varepsilon s_A) 2^{|y|+|z|}.$$

From this, we get that

$$\sum_{i}\left(\sum_{\sigma}(1 \oplus P_{A_i}(\sigma))\right) \geq \varepsilon s_A 2^{|y|} + 1 \;\rightarrow\; \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq (1 - \varepsilon s_A) 2^{|y|+|z|}.$$

The result follows.

Case $A = \oplus_1(A_1, \ldots, A_m)$. Here, we have that $P_A(y,z) = \bigoplus_{i=1}^{m} P_{A_i}(y)$. Let $B = \oplus_1(A_2, \ldots, A_m)$ and $B' = \oplus_0(A_2, \ldots, A_m)$. For every pair σ, τ, we have that $P_A(\sigma,\tau) \rightarrow P_{A_1}(\sigma), P_B(\sigma,\tau)$ and that $P_A(\sigma,\tau) \rightarrow P_{A_1}(\sigma) \oplus 1, P_{B'}(\sigma,\tau)$.

First consider $P_A(\sigma,\tau) \rightarrow P_{A_1}(\sigma) \oplus 1, P_{B'}(\sigma,\tau)$. This implies that

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1$$
$$\rightarrow \sum_{\sigma,\tau}(P_{A_1}(\sigma) \oplus 1) \geq \varepsilon s_{A_1} 2^{|y|+|z|} + 1,\ \sum_{\sigma,\tau} P_{B'}(\sigma,\tau) \geq \varepsilon s_{B'} 2^{|y|+|z|} + 1.$$

Therefore, since $\mathrm{tr}(A_1), \sum_{\sigma}(P_{A_1}(\sigma) \oplus 1) \geq \varepsilon s_{A_1} 2^{|y|} + 1 \rightarrow$, and by the inductive hypothesis,

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1,\ \mathrm{tr}(A_1) \;\rightarrow\; \mathrm{tr}(B').$$

By Lemma 6.3, $\mathrm{tr}(A_1), \mathrm{tr}(B') \rightarrow \mathrm{tr}(A)$. Therefore,

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1,\ \mathrm{tr}(A_1) \;\rightarrow\; \mathrm{tr}(A).$$

Now consider $P_A(\sigma,\tau) \rightarrow P_{A_1}(\sigma), P_B(\sigma,\tau)$. This implies that

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1$$
$$\rightarrow \sum_{\sigma,\tau} P_{A_1}(\sigma) \geq \varepsilon s_{A_1} 2^{|y|+|z|} + 1,\ \sum_{\sigma,\tau} P_B(\sigma,\tau) \geq \varepsilon s_B 2^{|y|+|z|} + 1.$$

Therefore, by the inductive hypothesis,

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1 \;\rightarrow\; \mathrm{tr}(A_1),\ \mathrm{tr}(B).$$

By an argument similar to the one used in the proof of Lemma 6.3, and by using the inductive hypothesis on A_1, it can be shown that $\mathrm{tr}(B) \rightarrow \mathrm{tr}(A_1), \mathrm{tr}(A)$. Therefore,

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1 \;\rightarrow\; \mathrm{tr}(A_1),\ \mathrm{tr}(A).$$

The result follows.

Case $A = \oplus_0(A_1, \ldots, A_m)$. Similar to the previous case. □

LEMMA 6.5. *For every small* $\mathrm{ACC}^0[2]$ *formula* A *and every small* $\mathrm{ACC}^0[2]$ *formula of the form* $\oplus_1(A_1, \ldots, A_m)$, *the sequents* $\rightarrow \mathrm{tr}(A), \mathrm{tr}(\neg A)$ *and*

$$\rightarrow \mathrm{tr}(\oplus_1(A_1, \ldots, A_m)),\ \mathrm{tr}(\oplus_0(A_1, \ldots, A_m))$$

have flat proofs of size $2^{(\log n)^{O(d)}}$.

PROOF. The proof of the first sequent is implicit in the proof of the previous lemma, case $A = \neg A_1$. The proof of the second sequent is similar. □

LEMMA 6.6. *For every small* $\mathrm{ACC}^0[2]$ *formula* A *of the form* $\vee(A_1, \ldots, A_m)$ *and for every* $i \in [m]$, *the sequent* $\mathrm{tr}(A) \rightarrow \mathrm{tr}(A_i), \mathrm{tr}(\bigvee_{j \neq i} A_j)$ *has a flat proof of size* $2^{(\log n)^{O(d)}}$.

PROOF. Let $B = \bigvee_{j \neq i} A_j$. Consider an arbitrary σ. By Lemma 5.5,

$$\sum_{\tau} P_A(\sigma,\tau) \geq 1 \;\rightarrow\; \sum_{j} P_{A_j}(\sigma) \geq 1$$

which implies that

$$\sum_{\tau} P_A(\sigma,\tau) \geq 1 \;\rightarrow\; P_{A_i}(\sigma),\ \sum_{j \neq i} P_{A_j}(\sigma) \geq 1.$$

In addition, again by Lemma 5.5,

$$\sum_{j \neq i} P_{A_j}(\sigma) \geq 1 \;\rightarrow\; \sum_{\tau} P_B(\sigma,\tau) \geq (1 - \varepsilon) 2^{|z|}.$$

Therefore, for every σ,

$$\sum_{\tau} P_A(\sigma,\tau) \geq 1 \;\rightarrow\; P_{A_i}(\sigma),\ \sum_{\tau} P_B(\sigma,\tau) \geq (1 - \varepsilon) 2^{|z|}.$$

From this, it can be shown that

$$\sum_{\sigma}\left(\sum_{\tau} P_A(\sigma,\tau)\right) \geq (1 - \varepsilon s_A) 2^{|y|+|z|}$$
$$\rightarrow \sum_{\sigma} P_{A_i}(\sigma) \geq \frac{1}{2}(1 - \varepsilon s_A) 2^{|y|},$$
$$\sum_{\sigma}\left(\sum_{\tau} P_B(\sigma,\tau)\right) \geq \frac{1}{2}(1 - \varepsilon s_A)(1 - \varepsilon) 2^{|y|+|z|},$$

since the inside sum on the left hand side is at most $2^{|z|}$. Note that $\frac{1}{2}(1 - \varepsilon s_A)(1 - \varepsilon) \geq 1/4 \geq \max\{\varepsilon s_{A_i}, \varepsilon s_B\}$. Therefore, by Lemma 6.4, $\mathrm{tr}(A) \rightarrow \mathrm{tr}(A_i), \mathrm{tr}(B)$. □

LEMMA 6.7. *For every small* $\mathrm{ACC}^0[2]$ *formula* A *of the form* $\wedge(A_1, \ldots, A_m)$ *and for every* $i \in [m]$, *the sequents* $\mathrm{tr}(A) \rightarrow \mathrm{tr}(A_i)$, $\mathrm{tr}(A) \rightarrow \mathrm{tr}(\bigwedge_{j \neq i} A_j)$ *and* $\mathrm{tr}(A_i), \mathrm{tr}(\bigwedge_{j \neq i} A_j) \rightarrow \mathrm{tr}(A)$ *have flat proofs of size* $2^{(\log n)^{O(d)}}$

PROOF. All three sequents are proved by using the proofs of the corresponding sequents for the $\vee$ connective (see Lemmas 6.2 and 6.6). We show how this is done for the first sequent; the arguments are similar for the other two.

We have to show that

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq (1 - \varepsilon s_A) 2^{|y|+|z|} \;\rightarrow\; \sum_{\sigma} P_{A_i}(\sigma) \geq (1 - \varepsilon s_{A_i}) 2^{|y|}.$$

Therefore,

$$\sum_{\sigma}(1 \oplus P_{A_i}(\sigma)) \geq \varepsilon s_{A_i} 2^{|y|} + 1 \;\rightarrow\; \sum_{\sigma}(1 \oplus P_{A_i}(\sigma)) \geq (1 - \varepsilon s_{A_i})2^{|y|}.$$

As in the proof of the sequent $\mathrm{tr}(A_i) \rightarrow \mathrm{tr}(A)$ in Lemma 6.2, it can be shown that

$$\sum_{\sigma}(1 \oplus P_{A_i}(\sigma)) \geq (1 - \varepsilon s_{A_i})2^{|y|} \;\rightarrow\; \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq (1 - \varepsilon s_A)2^{|y|+|z|}.$$

Therefore,

$$\sum_{\sigma}(1 \oplus P_{A_i}(\sigma)) \geq \varepsilon s_{A_i} 2^{|y|} + 1 \;\rightarrow\; \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq (1 - \varepsilon s_A)2^{|y|+|z|}.$$

From this, we get that

$$\sum_{i}\left(\sum_{\sigma}(1 \oplus P_{A_i}(\sigma))\right) \geq \varepsilon s_A 2^{|y|} + 1 \;\rightarrow\; \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq (1 - \varepsilon s_A)2^{|y|+|z|}.$$

The result follows.

Case $A = \oplus_1(A_1, \ldots, A_m)$. Here, we have that $P_A(y,z) = \bigoplus_{i=1}^{m} P_{A_i}(y)$. Let $B = \oplus_1(A_2, \ldots, A_m)$ and $B' = \oplus_0(A_2, \ldots, A_m)$. For every pair σ, τ, we have that $P_A(\sigma,\tau) \rightarrow P_{A_1}(\sigma), P_B(\sigma,\tau)$ and that $P_A(\sigma,\tau) \rightarrow P_{A_1}(\sigma) \oplus 1, P_{B'}(\sigma,\tau)$.

First consider $P_A(\sigma,\tau) \rightarrow P_{A_1}(\sigma) \oplus 1, P_{B'}(\sigma,\tau)$. This implies that

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1$$
$$\rightarrow \sum_{\sigma,\tau}(P_{A_1}(\sigma) \oplus 1) \geq \varepsilon s_{A_1} 2^{|y|+|z|} + 1, \sum_{\sigma,\tau} P_{B'}(\sigma,\tau) \geq \varepsilon s_{B'} 2^{|y|+|z|} + 1.$$

Therefore, since $\mathrm{tr}(A_1), \sum_\sigma(P_{A_1}(\sigma) \oplus 1) \geq \varepsilon s_{A_1} 2^{|y|} + 1 \rightarrow$, and by the inductive hypothesis,

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1,\ \mathrm{tr}(A_1) \;\rightarrow\; \mathrm{tr}(B').$$

By Lemma 6.3, $\mathrm{tr}(A_1), \mathrm{tr}(B') \rightarrow \mathrm{tr}(A)$. Therefore,

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1,\ \mathrm{tr}(A_1) \;\rightarrow\; \mathrm{tr}(A).$$

Now consider $P_A(\sigma,\tau) \rightarrow P_{A_1}(\sigma), P_B(\sigma,\tau)$. This implies that

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1$$
$$\rightarrow \sum_{\sigma,\tau} P_{A_1}(\sigma) \geq \varepsilon s_{A_1} 2^{|y|+|z|} + 1, \sum_{\sigma,\tau} P_B(\sigma,\tau) \geq \varepsilon s_B 2^{|y|+|z|} + 1.$$

Therefore, by the inductive hypothesis,

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1 \;\rightarrow\; \mathrm{tr}(A_1),\ \mathrm{tr}(B).$$

By an argument similar to the one used in the proof of Lemma 6.3, and by using the inductive hypothesis on A_1, it can be shown that $\mathrm{tr}(B) \rightarrow \mathrm{tr}(A_1), \mathrm{tr}(A)$. Therefore,

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq \varepsilon s_A 2^{|y|+|z|} + 1 \;\rightarrow\; \mathrm{tr}(A_1),\ \mathrm{tr}(A).$$

The result follows.

Case $A = \oplus_0(A_1, \ldots, A_m)$. Similar to the previous case. □

LEMMA 6.5. *For every small* $\mathrm{ACC}^0[2]$ *formula* A *and every small* $\mathrm{ACC}^0[2]$ *formula of the form* $\oplus_1(A_1, \ldots, A_m)$, *the sequents* $\rightarrow \mathrm{tr}(A), \mathrm{tr}(\neg A)$ *and*

$$\rightarrow \mathrm{tr}(\oplus_1(A_1, \ldots, A_m)),\ \mathrm{tr}(\oplus_0(A_1, \ldots, A_m))$$

have flat proofs of size $2^{(\log n)^{O(d)}}$.

PROOF. The proof of the first sequent is implicit in the proof of the previous lemma, case $A = \neg A_1$. The proof of the second sequent is similar. □

LEMMA 6.6. *For every small* $\mathrm{ACC}^0[2]$ *formula* A *of the form* $\vee(A_1, \ldots, A_m)$ *and for every* $i \in [m]$, *the sequent* $\mathrm{tr}(A) \rightarrow \mathrm{tr}(A_i)$, $\mathrm{tr}(\bigvee_{j \neq i} A_j)$ *has a flat proof of size* $2^{(\log n)^{O(d)}}$.

PROOF. Let $B = \bigvee_{j \neq i} A_j$. Consider an arbitrary σ. By Lemma 5.5,

$$\sum_{\tau} P_A(\sigma,\tau) \geq 1 \;\rightarrow\; \sum_{j} P_{A_j}(\sigma) \geq 1$$

which implies that

$$\sum_{\tau} P_A(\sigma,\tau) \geq 1 \;\rightarrow\; P_{A_i}(\sigma),\ \sum_{j \neq i} P_{A_j}(\sigma) \geq 1.$$

In addition, again by Lemma 5.5,

$$\sum_{j \neq i} P_{A_j}(\sigma) \geq 1 \;\rightarrow\; \sum_{\tau} P_B(\sigma,\tau) \geq (1 - \varepsilon)2^{|z|}.$$

Therefore, for every σ,

$$\sum_{\tau} P_A(\sigma,\tau) \geq 1 \;\rightarrow\; P_{A_i}(\sigma),\ \sum_{\tau} P_B(\sigma,\tau) \geq (1 - \varepsilon)2^{|z|}.$$

From this, it can be shown that

$$\sum_{\sigma}\left(\sum_{\tau} P_A(\sigma,\tau)\right) \geq (1 - \varepsilon s_A)2^{|y|+|z|}$$
$$\rightarrow \sum_{\sigma} P_{A_i}(\sigma) \geq \frac{1}{2}(1 - \varepsilon s_A)2^{|y|},$$
$$\sum_{\sigma}\left(\sum_{\tau} P_B(\sigma,\tau)\right) \geq \frac{1}{2}(1 - \varepsilon s_A)(1 - \varepsilon)2^{|y|+|z|},$$

since the inside sum on the left hand side is at most $2^{|z|}$. Note that $\frac{1}{2}(1 - \varepsilon s_A)(1 - \varepsilon) \geq 1/4 \geq \max\{\varepsilon s_{A_i}, \varepsilon s_B\}$. Therefore, by Lemma 6.4, $\mathrm{tr}(A) \rightarrow \mathrm{tr}(A_i), \mathrm{tr}(B)$. □

LEMMA 6.7. *For every small* $\mathrm{ACC}^0[2]$ *formula* A *of the form* $\wedge(A_1, \ldots, A_m)$ *and for every* $i \in [m]$, *the sequents* $\mathrm{tr}(A) \rightarrow \mathrm{tr}(A_i)$, $\mathrm{tr}(A) \rightarrow \mathrm{tr}(\bigwedge_{j \neq i} A_j)$ *and* $\mathrm{tr}(A_i), \mathrm{tr}(\bigwedge_{j \neq i} A_j) \rightarrow \mathrm{tr}(A)$ *have flat proofs of size* $2^{(\log n)^{O(d)}}$

PROOF. All three sequents are proved by using the proofs of the corresponding sequents for the $\vee$ connective (see Lemmas 6.2 and 6.6). We show how this is done for the first sequent; the arguments are similar for the other two.

We have to show that

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq (1 - \varepsilon s_A)2^{|y|+|z|} \;\rightarrow\; \sum_{\sigma} P_{A_i}(\sigma) \geq (1 - \varepsilon s_{A_i})2^{|y|}.$$

Since, for every σ, τ, $P_A(\sigma,\tau) = 1 \oplus Q^m_\vee(1 \oplus P_{A_1}\sigma(), \ldots, 1 \oplus P_{A_m}(\sigma), \tau)$, we have that

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq (1-\varepsilon s_A)2^{|y|+|z|},\ \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq \varepsilon s_A 2^{|y|+|z|}+1 \ \rightarrow$$

which implies that

$$\sum_{\sigma,\tau} P_A(\sigma,\tau) \geq (1-\varepsilon s_A)2^{|y|+|z|},\ \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq (1-\varepsilon s_A)2^{|y|+|z|} \ \rightarrow .$$

In addition,

$$\sum_{\sigma} P_{A_i}(\sigma) \geq \varepsilon s_{A_i} 2^{|y|}+1 \ \rightarrow\ \sum_{\sigma} P_{A_i}(\sigma) \geq (1-\varepsilon s_{A_i})2^{|y|},$$

by Lemma 6.4, and

$$\rightarrow \sum_{\sigma} P_{A_i}(\sigma) \geq \varepsilon s_{A_i} 2^{|y|}+1,\ \sum_{\sigma}(1 \oplus P_{A_i}(\sigma)) \geq (1-\varepsilon s_{A_i})2^{|y|}.$$

Therefore, it is sufficient to show that

$$\sum_{\sigma}(1 \oplus P_{A_i}(\sigma)) \geq (1-\varepsilon s_{A_i})2^{|y|}\ \rightarrow\ \sum_{\sigma,\tau}(1 \oplus P_A(\sigma,\tau)) \geq (1-\varepsilon s_A)2^{|y|+|z|}.$$

This in turn can be proved by the same argument that was used in Lemma 6.2 to prove that $\mathrm{tr}(A_i) \rightarrow \mathrm{tr}(A)$ when $A = \vee(A_1, \ldots, A_m)$. □

LEMMA 6.8. *For every small* $\mathrm{ACC}^0[2]$ *formula of the form* $\oplus_i(A_1, \ldots, A_m)$, *the sequents*

$$\mathrm{tr}(\oplus_i(A_2, \ldots, A_m)) \ \rightarrow\ \mathrm{tr}(A_1),\ \mathrm{tr}(\oplus_i(A_1, \ldots, A_m))$$

$$\mathrm{tr}(A_1),\ \mathrm{tr}(\oplus_i(A_1, \ldots, A_m)) \ \rightarrow\ \mathrm{tr}(\oplus_{i-1}(A_2, \ldots, A_m))$$

and

$$\mathrm{tr}(\oplus_i(A_1, \ldots, A_m)) \ \rightarrow\ \mathrm{tr}(A_1),\ \mathrm{tr}(\oplus_i(A_2, \ldots, A_m))$$

have flat proofs of size $2^{(\log n)^{O(d)}}$.

PROOF. By Lemma 6.3, we have that

$$\mathrm{tr}(A_1),\ \mathrm{tr}(\oplus_{i-1}(A_2, \ldots, A_m)) \rightarrow \mathrm{tr}(\oplus_i(A_1, \ldots, A_m)).$$

By an argument similar to the one used in the proof of that lemma, and by using Lemma 6.4 for A_1, it can be shown that

$$\mathrm{tr}(\oplus_i(A_2, \ldots, A_m)) \rightarrow \mathrm{tr}(A_1),\ \mathrm{tr}(\oplus_i(A_1, \ldots, A_m)).$$

The other two sequents follow from these two by Lemmas 6.1 and 6.5. □

7. Main result: the simulation of $\mathrm{ACC}^0[2]$ proofs

THEOREM 7.1. *If the family of sequents* $F = \{(\Gamma_n \rightarrow \Delta_n) : n \in \mathbb{N}\}$ *has a depth-d* $\mathrm{ACC}^0[2]$ *proof, then* $\mathrm{tr}(F) = \{(\mathrm{tr}(\Gamma_n) \rightarrow \mathrm{tr}(\Delta_n)) : n \in \mathbb{N}\}$ *has a flat proof of size* $2^{(\log n)^{O(d)}}$.

PROOF. It is sufficient to show that every rule of inference that can occur in an $\mathrm{ACC}^0[2]$ proof can be simulated by a flat proof in the sense that whenever $\Gamma \rightarrow \Delta$ can be inferred from $\Gamma' \rightarrow \Delta'$ and $\Gamma'' \rightarrow \Delta''$, then $\mathrm{tr}(\Gamma) \rightarrow \mathrm{tr}(\Delta)$ can be derived from $\mathrm{tr}(\Gamma') \rightarrow \mathrm{tr}(\Delta')$ and $\mathrm{tr}(\Gamma'') \rightarrow \mathrm{tr}(\Delta'')$ by using a flat proof. It is easy to verify that this can be done by using the sequents from the previous section. □

COROLLARY 7.2. *If the family of DNF formulas* $F = \{(\rightarrow C_{n1}, \ldots, C_{nm}) : n \in \mathbb{N}\}$ *has an* $\mathrm{ACC}^0[2]$ *proof, then* F *has a flat proof of size* $2^{(\log n)^{O(d)}}$.

PROOF. Given the previous theorem, there only remains to show that $\rightarrow C_{n1}, \ldots, C_{nm}$ can be derived from $\rightarrow \mathrm{tr}(C_{n1}), \ldots, \mathrm{tr}(C_{nm})$ by using a flat proof. Clearly, it suffices to show that for every formula A of the form $\wedge(u_1, \ldots, u_m)$, where all the u_i's are propositional variables, the sequent $\mathrm{tr}(A) \rightarrow A$ has a flat proof.

By Lemma 6.7, we have that for every i, $\mathrm{tr}(A) \rightarrow \mathrm{tr}(u_i)$. Since $\mathrm{tr}(u_i)$ is simply u_i, this is the same as $\mathrm{tr}(A) \rightarrow u_i$. The sequent $\mathrm{tr}(A) \rightarrow A$ can now be derived by repeated applications of the $\wedge$-right rule. □

8. Generalization to $\mathrm{ACC}^0[p^k]$ proofs

We now explain how the simulation of $\mathrm{ACC}^0[2]$ proofs can be generalized to $\mathrm{ACC}^0[p]$ proofs.

First, redefine our logic system by using the symbol $\oplus_j$ to denote a mod p connective instead of a mod 2 connective. For $j = 0, \ldots, p-1$, the formula $\oplus_j(A_1, \ldots, A_m)$ is interpreted to be true if and only if the number of true A_i's is equal to j mod p. To the initial sequents $\oplus_1() \rightarrow$ and $\rightarrow \oplus_0()$, add $\oplus_j() \rightarrow$ for $j = 2, \ldots, p-1$. The rules of inference remain the same. $\mathrm{ACC}^0[p]$ proofs can now be defined precisely as in Definition 2.3 and flat proofs now have mod p connectives on level two (see Definition 2.4).

The simulation of $\mathrm{ACC}^0[2]$ proofs made essential use of polynomials over $\mathbb{Z}_2$. Naturally, for $\mathrm{ACC}^0[p]$ proofs, polynomials over $\mathbb{Z}_p$ will be used instead. The symbols $\oplus$ and $\bigoplus$ will now denote addition of polynomials over $\mathbb{Z}_p$. Statements about polynomials over $\mathbb{Z}_p$ will be of the form $u \equiv a \pmod p$, where u is a polynomial over $\mathbb{Z}_p$ in the propositional variables and $a \in \mathbb{Z}_p$. Suppose that $u = a_0 + a_1M_1 + \cdots + a_NM_N$, where the M_i's are nonempty products of variables and the a_i's are coefficients in $\mathbb{Z}_p$. Then a statement such as $u \equiv a \pmod p$ will be realized by the formula $\oplus_{a-a_0}(M_1, \ldots, M_1, M_2, \ldots, M_2, \ldots, M_N, \ldots, M_N)$, where each M_i is repeated a_i times. Whenever a polynomial u appears on its own in a formula, then $u \equiv 1 \pmod p$ is assumed. In particular, $u \oplus 1$ is equivalent to $u \equiv 0 \pmod p$.

Probabilistic polynomials over $\mathbb{Z}_p$ are associated with the various connectives as follows. Let $\varepsilon = 1/2^{\lceil 1+\log p+\log s\rceil}$. For the $\neg$ connective, let $Q_\neg(u_1) = 1 \oplus (p-1)u_1$. For the mod p connectives, let

$$Q^m_{\oplus_j}(u) = 1 \oplus (p-1)\left((p-j) \oplus \bigoplus_{i=1}^{m} u_i\right)^{p-1}.$$

The fact that $Q^m_{\oplus_j}(u)$ computes $\oplus_j(u)$ follows from Fermat's Little Theorem: $a^{p-1} \equiv 1 \pmod p$ if and only if $a \not\equiv 0 \pmod p$.

Consider now an $\vee$ connective. Recall that the main idea behind the definition of $Q^m_\vee$ in the mod 2 case was the fact that if $\frac{1}{8}2^k \le |S| < \frac{1}{4}2^k$, then $R_k = \bigoplus_{i:h(i)=r} u_i$ is equal to 1 with probability at least 1/16. That is, the number of $i \in S$ such that $h(i) = r$ is odd with probability at least 1/16. In the mod p case, this is replaced by "the number of $i \in S$ such that $h(i) = r$ is not divisible by p." Accordingly, R_k is now $(\bigoplus_{i:h(i)=r} u_i)^{p-1}$ and $Q^m_\vee$ is defined by

$$Q^m_\vee(u,w) = 1 \oplus (p-1)\prod_{l=1}^{C}\prod_{k=3}^{\lceil \log m\rceil+3}\left(1 \oplus (p-1)\left(\bigoplus_i u_i F^i_{l,k}\right)^{p-1}\right).$$

Naturally, for the $\wedge$ connective we now have

$$Q^m_\wedge(u,w) = 1 \oplus (p-1)Q^m_\vee(1 \oplus (p-1)u_1, \ldots, 1 \oplus (p-1)u_m, w).$$

The lemmas of Sections 5 and 6, and their proofs, have to be adapted to the new definitions and the new context. Only minor changes are required and they usually involve basic properties of polynomials over $\mathbf{Z}_p$.

In Section 5, the key property of $Q^m_\vee$ now requires an additional hypothesis, namely, the fact that the inputs to the $\vee$ connective, the polynomials u_i, are such that $u_i \equiv 1 \pmod p$ or $u_i \equiv 0 \pmod p$. Consequently, in Lemmas 5.3 to 5.6, instead of "the sequent ... has a flat proof of size $2^{(\log n)^{O(d)}}$", we now have "the sequent ... can be derived from the sequents $\to u_i, u_i \oplus 1$, for $i = 1, \ldots, m$, by a flat proof of size $2^{(\log n)^{O(d)}}$". Recall that $u_i \oplus 1$ stands for $u_i \equiv 0 \pmod p$.

Some of the other changes are as follows. In the paragraph preceding Lemma 5.3, let $\varphi_{h,r} = (\bigoplus_{i:h(i)=r} u_i)^{p-1}$, $\varphi^i_{h,r} = u_i\varphi_{h,r}$ and $\varphi^{i,j}_{h,r} = u_iu_j\varphi_{h,r}$. In the proof of Lemma 5.4, $E^i_{h,r}$ should be defined as $1 \oplus (1 \oplus \bigoplus_{j:j\neq i,h(j)=r} u_j)^{p-1}$. Then, for every h, r,

$$\to E^i_{h,r}, \left(1 \oplus \bigoplus_{j:j\neq i,h(j)=r} u_j\right)^{p-1}.$$

The proof then proceeds as in the mod 2 case, with $1 \oplus E^i_{h,r}$ replaced by $(1 \oplus \bigoplus_{j:j\neq i,h(j)=r} u_j)^{p-1}$.

In Section 6, we first need an additional lemma.

LEMMA 8.1. *For every small* $\mathrm{ACC}^0[p]$ *formula* A, *the sequent*

$$\to P_A(\sigma_1, \ldots, \sigma_d), 1 \oplus P_A(\sigma_1, \ldots, \sigma_d)$$

has a flat proof of size $2^{(\log n)^{O(d)}}$.

This is easily proved by induction on the structure of A.

Lemma 6.1 should be for $\mathrm{tr}(\oplus_i(A_1, \ldots, A_m))$, $\mathrm{tr}(\oplus_j(A_1, \ldots, A_m)) \to$, $i, j \in \{0, \ldots, p-1\}$, $i \neq j$, instead of just $\mathrm{tr}(\oplus_1(A_1, \ldots, A_m))$, $\mathrm{tr}(\oplus_0(A_1, \ldots, A_m)) \to$. Similarly, Lemmas 6.3 and 6.8 should be for $i \in \{0, \ldots, p-1\}$. As for Lemma 6.5, the sequent $\to \mathrm{tr}(\oplus_1(A_1, \ldots, A_m))$, $\mathrm{tr}(\oplus_0(A_1, \ldots, A_m))$ should be replaced by

$$\to \mathrm{tr}(\oplus_0(A_1, \ldots, A_m)), \mathrm{tr}(\oplus_1(A_1, \ldots, A_m)), \ldots, \mathrm{tr}(\oplus_{p-1}(A_1, \ldots, A_m)).$$

The proof of this sequent uses the fact that $\varepsilon \le 1/(ps)$. In Lemma 6.4, cases have to be added for $\oplus_i(A_1, \ldots, A_m)$, $i \in \{2, \ldots, p-1\}$, but these are handled as in the mod 2 case.

Finally, we arrive at the main result, the simulation of $\mathrm{ACC}^0[p]$ proofs by quasipolynomial-size flat proofs.

THEOREM 8.2. *If the family of sequents* $F = \{(\Gamma_n \to \Delta_n) : n \in \mathbf{N}\}$ *has a depth-*d $\mathrm{ACC}^0[p]$ *proof, then* $\mathrm{tr}(F) = \{(\mathrm{tr}(\Gamma_n) \to \mathrm{tr}(\Delta_n)) : n \in \mathbf{N}\}$ *has a flat proof of size* $2^{(\log n)^{O(d)}}$.

This is proved exactly as Theorem 7.1.

It is possible to further generalize the simulation to $\mathrm{ACC}^0[p^k]$ proofs. These are defined exactly as $\mathrm{ACC}^0[p]$ proofs except that we now have connectives $\oplus_{j \bmod p^e}$, for $j = 0, \ldots, p^e - 1$ and $e = 1, \ldots, k$: the formula $\oplus_{j \bmod p^e}(A_1, \ldots, A_m)$ is true if and only if the number of true A_i's is equal to $j \bmod p^e$. The simulation then proceeds exactly as for $\mathrm{ACC}^0[p]$ proofs. In particular, polynomials over $\mathbf{Z}_p$ are still used and the simulating flat proofs still have only mod p connectives on level two (no mod p^e connectives). The only difference is in the handling of the mod p^e connectives.

The idea behind the translation of mod p^e connectives into low-degree polynomials over $\mathbf{Z}_p$ comes from a simple fact that was used, for example, by Beigel and Tarui [**BT94**] in their simulation of ACC^0 circuits (see Section 9 for more on their result): $u_1 + \cdots + u_m \equiv 0 \pmod{p^e}$ if and only if $\binom{u_1+\cdots+u_m}{p^l} \equiv 0 \pmod p$ for all $l \in \{0, \ldots, e-1\}$. Accordingly, let

$$Q^m_{\oplus_{j \bmod p^e}}(u) = \prod_{l=0}^{e-1}\left(1 \oplus (p-1)\left(\sum_{a=0}^{p^l}\left(\binom{p^e-j}{a}\sum_{S\subseteq[m],\,|S|=p^l-a}\prod_{i\in S}u_i\right)\right)^{p-1}\right).$$

It is not hard to see that the summation over a in this polynomial is congruent to 0 (mod p) if and only if $\binom{u_1+\cdots+u_m+p^e-j}{p^l} \equiv 0 \pmod p$.

The lemmas of Section 6 can be easily adapted to these new polynomials. One fact that must be used is that $\binom{p^e}{a} \equiv 0 \pmod p$ for every $a \in \{1, \ldots, p^e - 1\}$.

9. Possible extensions to ACC^0 proofs

Our depth-three simulation of $\mathrm{ACC}^0[p]$ proofs relies in an essential way on the corresponding simulation of $\mathrm{ACC}^0[p]$ circuits. As we have seen, most of the simulation consists in the formalization, in our logic system, using quasipolynomial-size flat proofs, of the key step in the circuit simulation, i.e., the computation of OR gates by low-degree probabilistic polynomials over $\mathbf{Z}_p$, and of several related basic facts. Fixed-depth simulations are also known for general ACC^0 circuits. For example, let SYM^+ denote the class of depth-two circuits with symmetric gates at the output and AND gates of polylog(n) fan-in on level one. Beigel and Tarui [**BT94**] have shown that ACC^0 circuits can be simulated by quasipolynomial-size SYM^+ circuits, which implies a simulation by depth-three threshold circuits. (See also [**GKR+95**].) Therefore, it is natural to ask if our simulation of $\mathrm{ACC}^0[p]$ proofs can be extended to ACC^0 proofs, perhaps by using the simulation results for ACC^0 circuits.

An important characteristic of the simulation of $\mathrm{ACC}^0[p]$ circuits, one that was not mentioned explicitly earlier, is that this simulation is "bottom-up". This means that the simulation of a circuit $A = *(A_1, \ldots, A_m)$ is done inductively by first simulating each subcircuit A_i and then combining with the $*$ gate to get the simulation of A. This characteristic plays an important role in our simulation because the rules of inference of the logic system involve the top-most connectives of the formulas. To illustrate this point, consider the $\vee$-right rule

$$\frac{\Gamma \to A_1, \vee(A_2, \ldots, A_m), \Delta}{\Gamma \to \vee(A_1, \ldots, A_m), \Delta}.$$

It is easy to see that this rule can be simulated by a flat proof if and only if the two sequents

$$\mathrm{tr}(A_1) \to \mathrm{tr}(\vee(A_1, \ldots, A_m))$$

and

$$\mathrm{tr}(\vee(A_2, \ldots, A_m)) \to \mathrm{tr}(\vee(A_1, \ldots, A_m))$$

have flat proofs themselves. Now the proof of these two sequents, which was done in Lemma 6.2, relies crucially on the fact that the polynomial $P_{\vee(A_1,\ldots,A_m)}$ computing the formula $\vee(A_1, \ldots, A_m)$ is defined in terms of the polynomials P_{A_i} computing the A_i's.

All of the known simulations of ACC^0 circuits, however, are "top-down". That is, the simulation of a circuit A of the form $B(G_1, \ldots, G_m)$, where the G_i's are the gates at level one, is done by first simulating B and then combining with the G_i's to get a simulation of A. As a consequence, it is not clear how a sequent such as $\mathrm{tr}(A_1) \to \mathrm{tr}(\vee(A_1, \ldots, A_m))$ could be proved without having to "translate back" to A_1, derive $\vee(A_1, \ldots, A_m)$ and then "translate" again to $\mathrm{tr}(\vee(A_1, \ldots, A_m))$. The reason is that if $A = *(A_1, \ldots, A_m) = B(G_1, \ldots, G_m)$, then, in a top-down simulation, $\mathrm{tr}(A)$ would be defined in terms of $\mathrm{tr}(B)$ and not in terms of the $\mathrm{tr}(A_i)$'s.

We are therefore led to the following question: is it possible to do a "bottom-up", fixed-depth simulation of ACC^0 circuits? This is an interesting question on its own, but a positive answer might also lead to a fixed-depth simulation of ACC^0 proofs. One way of obtaining such a "bottom-up" simulation would be to show that a circuit of the form MOD_r of SYM^+ can be simulated by a quasipolynomial-size SYM^+ circuit. Notice that this would imply that ACC^0 circuits with one level of arbitrary symmetric gates inserted anywhere could be simulated by quasipolynomial-size SYM^+ circuits. Such a result would be surprising; however, since no superpolynomial lower bound is known for SYM^+ circuits, this possibility cannot be ruled out. In addition, a bottom-up simulation of ACC^0 circuits—at least one that would proceed as indicated at the beginning of this section—would imply a simulation of ACC^0 circuits with an extra level of arbitrary symmetric gates at the input. The idea is to simply start with a stronger inductive basis. Therefore, the simulation of MOD_r of SYM^+ circuits by quasipolynomial-size SYM^+ circuits might not only be sufficient but also necessary for a bottom-up simulation of ACC^0 circuits. Whether this should be taken as an approach for obtaining such a simulation or as evidence in favor of its impossibility is open for debate.

10. Towards $\mathrm{ACC}^0[p]$-proof lower bounds

In this section, we outline an approach for obtaining exponential lower bounds for $\mathrm{ACC}^0[p]$ proofs.

In this article, we have shown that any quasipolynomial-size $\mathrm{ACC}^0[p]$ proof can be quasipolynomially simulated by a flat Frege proof. In fact, each formula can be viewed as a small-degree probabilistic polynomial. Thus, if we could succeed in obtaining lower bounds for our flat system, then $\mathrm{ACC}^0[p]$-proof lower bounds would follow.

A very successful recent method for obtaining lower bounds for very small depth propositional proof systems is the *interpolation method*. The general idea is as follows. Let $A(x,z) \wedge B(y,z)$ be a 3CNF formula, where $A(x,z)$ is a 3CNF formula involving the variables x and z, and $B(y,z)$ is a 3CNF formula involving the variables y and z. (Here, x, y, and z each denote a vector of propositional variables.) Let S be a particular propositional proof system. Then S has a *feasible interpolation theorem* if for every 3CNF formula F of the above split form, there exists a circuit, $C(z)$, such that: $C(a) = 1$ only if $A(x,a)$ is unsatisfiable, and $C(a) = 0$ only if $B(y,a)$ is unsatisfiable. Furthermore, the size of the circuit C must be polynomial in the size of the shortest S-refutation of F. Note that for any assignment a to z, at least one of $A(x,a)$ or $B(y,a)$ must be unsatisfiable because F is unsatisfiable. So the function computed by $C(z)$ exists, and we are interested in the case where the function is computable by polynomial-size circuits whenever F has a short refutation in S. There is also a monotone version of the above definition. Define $A(x,z) \wedge B(y,z)$ to be monotone if $A(x,z)$ involves only positive occurrences of z variables. In this case, S has a *feasible monotone interpolation theorem* if for every 3CNF formula of the monotone split form, there exists a monotone, polynomial-size circuit $C(z)$ computing the above (monotone) function.

Now suppose that we can show that S has a feasible interpolation theorem. Then, using a result due to Razborov [**Razb**] (and later generalized by Krajíček [**Kraa**]) this implies lower bounds for S, assuming a standard cryptographic conjecture. Furthermore, if we can show that S has a feasible monotone interpolation theorem, then using the method of [**BPR95**] (generalized by [**Kraa**]), together with exponential lower bounds on the size of monotone circuits, we get unconditional lower bounds for our proof system S. Thus, feasible interpolation theorems are quite important because they give rise to lower bounds.

This method has been quite successful in obtaining new lower bounds. For example, it has been used to obtain exponential lower bounds for Resolution and Cutting Planes [**BPR95, Krab, Pud, CH**], as well as for Nullstellensatz proofs [**PS**]. It has also been applied to get conditional lower bounds for bounded arithmetic [**Razb**], for Gröbner refutations [**PS**], and for generalizations of Cutting Planes [**IP, Krab**].

When does a proof system have an effective interpolation theorem, and how hard is it to show this? This is a very interesting question that does not have a satisfactory answer at present. For example, we do not know if $\mathrm{ACC}^0[p]$ proofs have effective interpolation theorems. An interesting observation, due to Impagliazzo, can be used to obtain interpolation theorems. Suppose that system S can be *deterministically simulated efficiently*. This means that there is an efficient procedure testing whether or not there is a polynomial-size S proof of a given formula F. The

existence of an efficient deterministic simulation not only suggests a good deterministic propositional theorem prover, but it also implies an effective interpolation theorem for S. This approach was used to obtain interpolation theorems for the Nullstellensatz system [**CEI96, PS**], as well as for the polynomial calculus [**PS**].

In light of the interpolation technology discussed above, it would be very interesting to obtain a subexponential-time deterministic simulation of flat proofs. First, this would give rise to a powerful and possibly efficient deterministic theorem prover. Second, it would imply superpolynomial lower bounds for $\mathrm{ACC}^0[p^k]$ proofs (assuming a standard cryptographic assumption).

As mentioned above, [**CEI96**] shows how to deterministically simulate polynomial calculus refutations using a variation of the Gröbner basis algorithm; our flat proofs can be viewed as a generalization of small-degree polynomial calculus refutations in the following way. A small-degree polynomial calculus refutation is a refutation in our system where each line is a small-degree polynomial over the main variables x. One derives new polynomials as small-degree linear combinations of previously generated polynomials. This linearity of the rules is exactly what enables the Gröbner basis algorithm to be used to efficiently find a small-degree polynomial calculus refutation, if one exists. In contrast, a flat refutation can be viewed as a *parametric* polynomial calculus refutation: each line is a small-degree polynomial but now over two types of variables, the main variables x and the symbolic parameters y. The intended interpretation is that $f(x, y)$ is true if for almost all values to the y variables, $f(x, y)$ evaluates to 1, and $f(x, y)$ is false if for almost all values of the y variables, $f(x, y)$ evaluates to 0. It is open whether or not there is a suitable generalization of the Gröbner basis algorithm which would give rise to a subexponential-time simulation of flat proofs.

Acknowledgments

We thank Kenneth Regan for useful conversations concerning the proof of Proposition 5.1.

References

[AH94] E. Allender and U. Hertrampf, *Depth reduction for circuits of unbounded fan-in*, Inform. and Comput. **112** (1994), no. 2, 217–238.

[Ajt88] M. Ajtai, *The complexity of the pigeonhole principle*, Proceedings of the 29th IEEE Symposium on Foundations of Computer Science, 1988, pp. 346–355.

[Ajt90] M. Ajtai, *Parity and the pigeonhole principle*, Feasible Mathematics (S.R. Buss and P.J. Scott, eds.), Birkhauser, 1990, pp. 1–24.

[All89] E. Allender, *A note on the power of threshold circuits*, Proceedings of the 30th IEEE Symposium on Foundations of Computer Science, 1989, pp. 580–584.

[BC96] S. Buss and P. Clote, *Cutting planes, connectivity, and threshold logic*, Arch. Math. Logic **35** (1996), no. 1, 33–62.

[BIK+a] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, and Pudlák P., *Lower bounds on Hilbert's nullstellensatz and propositional proofs*, To appear in *Proceedings of the London Mathematical Society*.

[BIK+b] S. Buss, R. Impagliazzo, J. Krajíček, P. Pudlák, A.A. Razborov, and J. Sgall, *Proof complexity in algebraic systems and bounded-depth Frege systems with modular counting*, Manuscript, 1996.

[BP] P. Beame and T. Pitassi, *An exponential separation between the matching principle and the pigeonhole principle*, To appear in *Annals of Pure and Applied Logic*.

[BPR95] M. Bonet, T. Pitassi, and R. Raz, *Lower bounds for cutting planes proofs with small coefficients*, Proceedings of the 27th ACM Symposium on Theory of Computing, 1995, Full version to appear in *Journal of Symbolic Logic*.

[BT94] R. Beigel and J. Tarui, *On ACC*, Comput. Complexity **4** (1994), 350–366.

[CEI96] M. Clegg, J. Edmonds, and R. Impagliazzo, *Using the Gröbner basis algorithm to find proofs of unsatisfiability*, Proceedings of the 28th ACM Symposium on Theory of Computing, 1996, pp. 174–183.

[CH] S.A. Cook and A. Haken, *An exponential lower bound for the size of monotone real circuits*, Manuscript, 1995.

[GKR+95] F. Green, J. Köbler, K. Regan, T. Schwentik, and J. Torán, *The power of the middle bit of a #P function*, J. Comput. System Sci. **50** (1995), 456–467.

[Hak85] A. Haken, *The intractability of resolution*, Theoret. Comput. Sci. **39** (1985), 297–308.

[IP] R. Impagliazzo and T. Pitassi, *Interpolation theorems for generalized cutting planes*, Manuscript, 1996.

[Kraa] J. Krajíček, *Bounded arithmetic, propositional logic and complexity theory*, Cambridge University Press.

[Krab] J. Krajíček, *Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic*, To appear in the *Journal of Symbolic Logic*.

[PS] P. Pudlák and J. Sgall, *Algebraic models of computation and interpolation for algebraic proof systems*, Manuscript, 1996.

[Pud] P. Pudlák, *Lower bounds for resolution and cutting planes proofs and monotone computation*, To appear in *Journal of Symbolic Logic*.

[Raza] A.A. Razborov, *Lower bounds for the polynomial calculus*, Manuscript, 1996.

[Razb] A.A. Razborov, *Unprovability of lower bounds on the circuit size in certain fragments of bounded arithmetic*, Izvestiya of the R.A.N. **59**, no. 1, 201–224.

[Raz87] A.A. Razborov, *Lower bounds for the size of circuits of bounded depth with basis* $\{\wedge, \oplus\}$, Mathematical Notes of the Academy of Sciences of the USSR **41** (1987), no. 4, 333–338.

[Reg93] K. Regan, *Efficient reductions from NP to parity using error-correcting codes*, Tech. Report 93–24, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, U.S.A., 1993.

[Rii] S. Riis, *Count(q) does not imply Count(p)*, Manuscript, 1995.

[Sip83] M. Sipser, *A complexity theoretic approach to randomness*, Proceedings of the 15th ACM Symposium on Theory of Computing, 1983, pp. 330–335.

[Smo87] R. Smolensky, *Algebraic methods in the theory of lower bounds for boolean circuit complexity*, Proceedings of the 19th ACM Symposium on Theory of Computing, 1987, pp. 77–82.

[Wig94] A. Wigderson, *Lectures on the fusion method and derandomization*, Tech. Report SOCS-95.2, School of Computer Science, McGill University, Montréal, Québec, Canada, 1994.

[Yao90] A.C.-C. Yao, *On ACC and threshold circuits*, Proceedings of the 31st IEEE Symposium on Foundations of Computer Science, 1990, pp. 619–627.

Department of Computer Science and UMIACS, University of Maryland, College Park, MD 20742, U.S.A.

Current address: Department of Mathematics and Computer Science, Clarkson University, Potsdam, NY 13699-5815, U.S.A.

E-mail address: alexis@cs.umd.edu

Department of Computer Science, University of Arizona, Tucson, AZ 85721, U.S.A.

E-mail address: toni@cs.arizona.edu

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

A Quantifier-Free Theory Based on a String Algebra for NC^1

François Pitt

ABSTRACT. In this paper, we define a string algebra L_1 which contains exactly the functions in NC^1 and give many examples of natural L_1 definitions for simple NC^1 functions (note that throughout this document, we will use NC^1 to mean U_{E^*}-*uniform* $NC^1 = ALOGTIME$). Based on L_1, we define a quantifier-free theory T_1 and show that for any formula A of T_1, $[\![A]\!]^m$, a family of propositional formulas which translate A, all have polysize $\mathcal{F}$-proofs whenever A is provable in T_1. Note that this work is still very much in progress.

1. Introduction

In 1964, Cobham [14] introduced his now famous recursion-theoretic characterization of the class of polynomial time computable functions FP. Based on this characterization, Steve Cook introduced his equational theory PV [15], and proved two theorems about PV supporting the claim that it captures polytime reasoning: first, any theorem of PV can be translated into a family of propositional formulas which have polysize $e\mathcal{F}$-proofs; second, PV can prove the soundness of $e\mathcal{F}$.

More recently, Steven Bellantoni and Steve Cook [3] gave a simpler characterization of FP (which uses the idea of safe versus normal parameters to dispense with the explicit size bounds in Cobham's characterization) and based on that characterization, they gave another equational theory that captures polytime reasoning [4].

1.1. Algebraic characterizations of NC^1. Based on Cobham's characterization of FP, Peter Clote came up with two closely related characterizations of FNC^1 [10, 12] by restricting the form of the recursion principle. His second characterization, N_0', uses a few simple base functions and the schemes of *Concatenation Recursion on Notation (CRN)* and *k-Bounded Recursion on Notation (k-BRN)*, and is structurally similar to Cobham's characterization. Unfortunately, some simple NC^1 functions have complex definitions in ALV', a fact which seems to be caused by the use of the inherently sequential scheme of k-BRN to define functions computed

1991 *Mathematics Subject Classification.* Primary 68Q15, 03F30.
This research was supported by the National Science and Engineering Research Council of Canada (NSERC), under the supervision of Steve Cook.

by parallel circuits—especially since k-BRN is based on Barrington's sophisticated characterization of NC^1 in terms of bounded-width branching programs [2].

To remedy this, Steve Cook suggested looking at schemes of recursion similar to Bill Allen's *Divide-and-Conquer Recursion (DCR)* [1], which he uses to characterize NC^1 and NC. I studied the algebra obtained by replacing k-BRN in Clote's N_0' by *k-Bounded Tree Recursion on Notation (k-BTRN)*, and showed it to be equivalent to N_0'. Then, I decided to try to define an algebra of functions based on Bloch's characterization of NC^1 [5], which uses Bellantoni and Cook's idea of safe versus normal parameters to dispense with explicit bounds in the recursion schemes. This proved very fruitful: it is possible to define an elegant algebra (which I call L_1 and present in Section 2 below) without using safe or normal parameters, simply by restricting explicitly the nesting depth of recursion. L_1 has simple definitions for many simple NC^1 functions, and the correspondence between NC^1 circuits and functions in L_1 is clear because all functions in L_1 are length-determined string functions (*i.e.*, functions from $\{0,1\}^n$ to $\{0,1\}^m$, where m depends only on n).

1.2. Theories of NC^1 reasoning. By analogy with Cook's PV, Clote came up in 1990 with an equational theory ALV based on his first algebra N_0 and showed that any theorem of ALV can be translated into a family of propositional formulas which have short $\mathcal{F}$-proofs [11]. Based on his second algebra N_0', Clote defined an equational theory ALV' and showed that its translated theorems also have polysize $\mathcal{F}$-proofs [12]. Unfortunately, he could not show that either of his systems can prove the soundness of $\mathcal{F}$.

Other researchers have looked at first-order theories for $ALOGTIME$ reasoning, most notably Gaisi Takeuti and Peter Clote [18, 13], but their theories are based on Clote's ALV' and do not use any form of tree recursion or tree induction.

Based on the algebra L_1, I have defined a quantifier-free theory T_1 in the style of Cook's PV and Clote's ALV', which I describe in Section 3 below (note that the axioms and rules of inference of T_1 are not yet completely fixed: further work will probably bring about small changes in the particular way that T_1 is formalized). My hope is that T_1 will prove to be more natural than other theories at capturing NC^1 reasoning because of its use of tree recursion and tree induction. So far, I have been able to show that any theorem of T_1 can be translated into a family of propositional tautologies which have short $\mathcal{F}$-proofs (Section 4), and the proof of this fact is considerably simpler than Clote's corresponding claim for ALV' because the functions in T_1 are those of L_1 which are length-determined. It remains to show that T_1 can prove the soundness of $\mathcal{F}$, a problem which I discuss in the Conclusion.

2. The string algebra L_1

2.1. Basic definitions. The basic objects of the algebra are strings over the alphabet $\{0,1\}$. The set of all such strings can be defined inductively: ε (the empty string) is a string, and if x is a string, then so are $x0$ and $x1$. Together with a wish for simplicity, this inductive definition motivates our choice of base functions.

Following, we define the base functions and the basic operators which we will use to construct new functions. We use $|x|$ to denote the length of x (*i.e.*, the number of symbols (bits) in the string x), $\vec{x}_k$ to denote a k-tuple of variables, and $\vec{x}$ to denote an arbitrary tuple of variables.

BASE: The set of *base functions* consists of (in order of increasing arity):

$$\varepsilon = \text{the empty string (a constant)},$$
$$x \cdot 0 = x \text{ with a '0' appended to the right},$$
$$x \cdot 1 = x \text{ with a '1' appended to the right},$$
$$\blacktriangleright x = \text{the } \lceil |x|/2 \rceil \text{ rightmost bits of } x \text{ ("right half")},$$
$$x \triangleleft y = x \text{ with } |y| \text{ bits removed from the right ("right chop")},$$
$$x\,?\,(y,z_0,z_1) = \begin{cases} y & \text{if } x = \varepsilon, \\ z_0 & \text{if } x = w0 \text{ for some } w, \\ z_1 & \text{if } x = w1 \text{ for some } w, \end{cases} \quad (\text{"conditional"})$$
$$\mathcal{I}_k^n(x_1,\ldots,x_n) = x_k \quad \text{for any } 1 \le k \le n \quad (\text{"identity" or "projection"}).$$

NOTE 1: In the definition of $x\,?\,(y,z_0,z_1)$, it is assumed that $|z_0| = |z_1|$. If that is not the case, then the value returned will be padded on the left with as many 0's as are necessary to make z_0 and z_1 the same length (the length of y does not change).

NOTE 2: Our "right half" function was called "back half (Bh)" by Bill Allen [1] and Stephen Bloch [6]. We introduce the new nomenclature because we feel that it is more representative of the action of the function, and the new notation to serve as a graphical reminder of that action (picture the black triangle cutting into the left part of x). Similarly, our "right chop" function was called "chop" by Steve Cook [16] and "most significant part (Msp)" by Bill Allen and Stephen Bloch. Our new notation should serve as a useful graphical mnemonic for the function's purpose and action (picture the bits of y cutting into the bits of x from the right—in the direction pointed to by the function symbol).

COMP: f is defined from g and $h_1,\ldots,h_k$ by *composition* if

$$f(\vec{x}) = g(h_1(\vec{x}),\ldots,h_k(\vec{x})).$$

CRN: f is defined from g and h by *concatenation recursion on notation* on x if h satisfies $h(i,x,\vec{y}) = 0,1$ for $i = 0,1$ and

$$f(\varepsilon,\vec{y}) = g(\vec{y}),$$
$$f(xi,\vec{y}) = f(x,\vec{y}) \cdot h(i,x,\vec{y}) \quad \text{for } i = 0,1.$$

TRN: f is defined from g, h, $\vec{l}$, and $\vec{r}$ by *tree recursion on notation* on x if

$$f(x,\vec{y}) = \begin{cases} g(x,\vec{y}) & \text{for } x = \varepsilon, 0, 1, \\ h\big(x,\vec{y},f(x\blacktriangleleft,\vec{l}(x\blacktriangleleft,\vec{y})),f(\blacktriangleright x,\vec{r}(\blacktriangleright x,\vec{y}))\big) & \text{otherwise,} \end{cases}$$

where $x\blacktriangleleft = x \triangleleft \blacktriangleright x$ (the $\lfloor |x|/2 \rfloor$ leftmost bits of x).

DEFINITION 2.1.

- L_0 is the closure of BASE under COMP and CRN.
- TRN$[L_0]$ is the set of all functions defined by TRN from functions in L_0.
- L_1 is the closure of $L_0 \cup$ TRN$[L_0]$ under COMP and CRN.

REMARK 2.2. In a previous version of this paper, we used a weaker but more natural form of TRN without the parameter functions $\vec{l}$ and $\vec{r}$. Relying on Bloch's result [5], we were able to show that the weaker algebra based on this form of TRN

also captured all of FNC^1. Thus, the parameter functions are not essential in the definition of TRN; they merely make it easier to define certain functions, to show that the algebra L_1 corresponds to FNC^1, and to prove certain properties of our theory T_1.

Notation. The next few subsections contain mainly function definitions, where the following notational conventions will be used.

- $\vec{x}$ represents an arbitrary tuple of variables while $\vec{x}_k$ represents a k-tuple of variables. For any constant string c, $\vec{c}_k$ represents the tuple consisting of k copies of c. (Also, we use 0 and 1 to stand for $\varepsilon \cdot 0$ and $\varepsilon \cdot 1$, respectively.)
- Unary functions have higher precedence than binary functions and binary functions have higher precedence than functions of higher arity. Concatenation has higher precedence than any other binary function when represented by juxtaposition; it has lower precedence than any other binary function when represented by "$\cdot$".
- i and j represent arbitrary fixed single bits, whereas k, ℓ, m, and n represent arbitrary fixed non-zero natural numbers. When 2^k is used, k ranges over all natural numbers (including zero), and when k (or 2^k) is used in conjunction with ℓ, we adopt the convention that $\ell \leq k$ (or $\ell \leq 2^k$, respectively) by default.
- Whenever the expression $k[+1]$ occurs in an equation, it is understood to mean that the equation holds when either k or $k+1$ is substituted for every occurrence of the expression.
- The notation $k \times x$ stands for $\overbrace{x \cdots x}^{k}$ (*i.e.*, x concatenated with itself k times; we will define concatenation shortly). We let $0 \times x = \varepsilon$ and use $\widehat{k}$ as an abbreviation for $k \times 1$, *i.e.*, the unary string representing k.

2.2. Functions in L_0. In this section, we will define a great number of functions in L_0 and show that many generalizations of CRN and TRN can be simulated in L_0. We are motivated by two goals: to define the machinery necessary to prove that L_1 contains all of FNC^1, and to show that many useful functions have simple definitions in our algebra. The very first function we will define is concatenation, using CRN:

$$\begin{aligned} y \cdot \varepsilon &= \mathcal{I}_1^1(y), \\ y \cdot xi &= (y \cdot x) \cdot \mathcal{I}_1^3(i, x, y) \quad \text{for } i = 0, 1. \end{aligned}$$

Next, we will define a few simple variations on some of the BASE functions. (From now on, we will use an informal notation in our definitions because it is easier to read. The reader should have no trouble supplying the more formal definitions should the need arise.) These functions include the following. The rightmost bit of x: $x' = x \,?\, (\varepsilon, 0, 1)$; x with its rightmost bit removed: $x\!< = x \lhd 1$; the $\lfloor |x|/2 \rfloor$ leftmost bits of x: $x\!\blacktriangleleft = x \lhd \blacktriangleright x$.

A function which reverses the bits of x can be defined by first using CRN to define a function $\mathsf{reverse}(x, y)$, which returns the $|y|$ rightmost bits of x reversed:

$$\begin{aligned} \mathsf{reverse}(x, \varepsilon) &= \varepsilon, \\ \mathsf{reverse}(x, yi) &= \mathsf{reverse}(x, y) \cdot (x \lhd y)' \quad \text{for } i = 0, 1. \end{aligned}$$

Then, $\mathsf{rev}(x) = \mathsf{reverse}(x, x)$ returns the reverse of x. Using this function, we can now define symmetric counterparts to some of the earlier functions:

$$y \rhd x = \mathsf{rev}(\mathsf{rev}(x) \lhd \mathsf{rev}(y)), \qquad \blacktriangleright x = \mathsf{rev}(\mathsf{rev}(x)\!<), \qquad `x = \mathsf{rev}(x)'.$$

Now, let us introduce a generalization of CRN: a function f is defined by *reverse CRN* on x from g and h if h satisfies $h(i, x, \vec{y}) = 0, 1$ for $i = 0, 1$ and

$$\begin{aligned} f(\varepsilon, \vec{y}) &= g(\vec{y}), \\ f(ix, \vec{y}) &= h(i, x, \vec{y}) \cdot f(x, \vec{y}) \quad \text{for } i = 0, 1. \end{aligned}$$

If f is defined by reverse CRN on x from g and h, then we can use CRN to define

$$\begin{aligned} \mathsf{aux\_}f(\varepsilon, \vec{y}) &= \mathsf{rev}(g(\vec{y})), \\ \mathsf{aux\_}f(xi, \vec{y}) &= \mathsf{aux\_}f(x, \vec{y}) \cdot h(i, x, \vec{y}) \quad \text{for } i = 0, 1, \end{aligned}$$

and $f(x, \vec{y}) = \mathsf{rev}(\mathsf{aux\_}f(\mathsf{rev}(x), \vec{y}))$, using COMP. In what follows, we will use the notational conventions outlined before this section and we will no longer write "for $i = 0, 1$" when using CRN to define new functions.

It is useful to have a conditional which tests for the length of a string: $x ?^{\mathsf{l}} (y, z) = x \,?\, (y, z, z)$; or for the truth-value of a string (with the conventions that $1 = \mathtt{true}$, 0 (or ε) $= \mathtt{false}$, and the truth-value of a string of length greater than one is determined by the string's rightmost bit): $x ?^{\mathsf{b}} (y, z) = x \,?\, (z, z, y)$.

The next functions we will introduce are the Boolean operators. Let us start with two different signum functions. One that returns true if its argument is empty and false otherwise: $\approx^{\mathsf{l}} x = x ?^{\mathsf{l}} (1, 0)$; and one that simply returns the truth-value of its argument: $\approx^{\mathsf{b}} x = x ?^{\mathsf{b}} (1, 0)$. Now, we can define the standard connectives together with some useful functions for comparing bits.

$$\begin{array}{lll} \neg^{\mathsf{b}} x = x ?^{\mathsf{b}} (0, 1) & x \wedge^{\mathsf{b}} y = x ?^{\mathsf{b}} (\approx^{\mathsf{b}} y, 0) & x \vee^{\mathsf{b}} y = x ?^{\mathsf{b}} (1, \approx^{\mathsf{b}} y) \\ x \rightarrow^{\mathsf{b}} y = x ?^{\mathsf{b}} (\approx^{\mathsf{b}} y, 1) & x \leftrightarrow^{\mathsf{b}} y = x ?^{\mathsf{b}} (\approx^{\mathsf{b}} y, \neg^{\mathsf{b}} y) & x \oplus^{\mathsf{b}} y = x ?^{\mathsf{b}} (\neg^{\mathsf{b}} y, \approx^{\mathsf{b}} y) \\ x \geq^{\mathsf{b}} y = y \rightarrow^{\mathsf{b}} x & x \leq^{\mathsf{b}} y = x \rightarrow^{\mathsf{b}} y & x =^{\mathsf{b}} y = x \leftrightarrow^{\mathsf{b}} y \\ x >^{\mathsf{b}} y = \neg^{\mathsf{b}} (x \leq^{\mathsf{b}} y) & x <^{\mathsf{b}} y = \neg^{\mathsf{b}} (x \geq^{\mathsf{b}} y) & x \neq^{\mathsf{b}} y = \neg^{\mathsf{b}} (x =^{\mathsf{b}} y) \end{array}$$

Now, we will define useful functions for manipulating strings. First, a simple function which returns a string of the same length as its input, but consisting entirely of 0's or 1's:

$$_j\varepsilon = \varepsilon, \qquad _j(xi) = {}_jx \cdot j.$$

Next, we define functions to perform simple bit manipulations on strings (extract single bits or substrings, pad to a certain length).

- "Left bit": $\mathsf{lb}(x, y) = y ?^{\mathsf{l}} (\varepsilon, `(\blacktriangleright y \rhd x))$ returns bit number $|y|$ of x from the left; "right bit": $\mathsf{rb}(x, y) = y ?^{\mathsf{l}} (\varepsilon, (x \lhd y\!<)')$ returns bit number $|y|$ of x from the right (both are equal to ε if $y = \varepsilon$ or $|y| > |x|$). For convenience, we also define $\mathsf{lb}^{\mathsf{b}}(x, y) = \approx^{\mathsf{b}} \mathsf{lb}(x, y)$ and $\mathsf{rb}^{\mathsf{b}}(x, y) = \approx^{\mathsf{b}} \mathsf{rb}(x, y)$ which return 0 or 1 for all arguments.
- "Left cut": $\mathsf{lc}(x, y) = x \lhd (y \rhd x)$ returns the $|y|$ leftmost bits of x; "right cut": $\mathsf{rc}(x, y) = (x \lhd y) \rhd x$ returns the $|y|$ rightmost bits of x (both return ε if $y = \varepsilon$ and x if $|y| \geq |x|$).
- "Left pad": $\mathsf{lp}_j(x, y) = {}_j(y \lhd x) \cdot x$ returns x padded on the left with j's so that $|\mathsf{lp}_j(x, y)| = |y|$; "right pad": $\mathsf{rp}_j(x, y) = x \cdot {}_j(x \rhd y)$ returns x padded on the right with j's so that $|\mathsf{rp}_j(x, y)| = |y|$ (both return x if $|y| \leq |x|$).

Following which, we can define a number of functions to compare the lengths of strings.

$$x \geq^{\mathsf{l}} y = \approx^{\mathsf{l}}(x \rhd y) \qquad x \leq^{\mathsf{l}} y = \approx^{\mathsf{l}}(x \lhd y) \qquad x =^{\mathsf{l}} y = (x \geq^{\mathsf{l}} y) \wedge^{\mathsf{b}} (x \leq^{\mathsf{l}} y)$$
$$x >^{\mathsf{l}} y = \neg^{\mathsf{b}}(x \leq^{\mathsf{l}} y) \qquad x <^{\mathsf{l}} y = \neg^{\mathsf{b}}(x \geq^{\mathsf{l}} y) \qquad x \neq^{\mathsf{l}} y = \neg^{\mathsf{b}}(x =^{\mathsf{l}} y)$$
$$\mathsf{max}^{\mathsf{l}}(x, y) = x \geq^{\mathsf{l}} y \,?^{\mathsf{b}}\, (x, y) \qquad \mathsf{max}^{\mathsf{l}}_1(x) = x$$
$$\mathsf{max}^{\mathsf{l}}_{2k[+1]}(\vec{x}_k, \vec{y}_{k[+1]}) = \mathsf{max}^{\mathsf{l}}\big(\mathsf{max}^{\mathsf{l}}_k(\vec{x}_k), \mathsf{max}^{\mathsf{l}}_{k[+1]}(\vec{y}_{k[+1]})\big)$$

We can also define functions to manipulate the lengths of strings, namely $\mathsf{div}^{\mathsf{l}}_{2^k}(x)$ which returns a string whose length is $\lfloor |x|/2^k \rfloor$ and a corresponding $\mathsf{mod}^{\mathsf{l}}_{2^k}(x)$ function.

$$\mathsf{div}^{\mathsf{l}}_1(x) = {}_1x \qquad \mathsf{div}^{\mathsf{l}}_{2k}(x) = \mathsf{div}^{\mathsf{l}}_k(x\blacktriangleleft)$$
$$\mathsf{mod}^{\mathsf{l}}_{2^k}(x) = \big(2^k \times \mathsf{div}^{\mathsf{l}}_{2^k}(x)\big) \rhd {}_1x$$

(For convenience and future use, we will also define $\mathsf{lmod}^{\mathsf{l}}_{2^k}(x) = \mathsf{lc}(x, \mathsf{mod}^{\mathsf{l}}_{2^k}(x))$ and $\mathsf{rmod}^{\mathsf{l}}_{2^k}(x) = \mathsf{rc}(x, \mathsf{mod}^{\mathsf{l}}_{2^k}(x))$.) Finally, we have all the functions we need to define a tuple function $(\langle \vec{x}_k \rangle_k)$ and corresponding projection functions $(\pi^k_\ell(x))$. To form tuples, we simply concatenate the arguments together after padding them on the left so that they all have the same length. The projection functions are then defined easily using the lc and rc functions as well as $\rhd$ and $\lhd$. One small complication arises because we can only divide the length of a string by a power of 2, so we need to form tuples that always have a power of 2 elements even when there are fewer of them that are actually input values. The definitions follow and are inspired by similar definitions in Bloch's paper [5]. (The tuple function is defined in terms of an auxiliary function tuple which has an extra parameter specifying the length to which each value should be padded.)

$$\mathsf{tuple}_1(x, z) = \mathsf{lp}_0(x, z)$$
$$\mathsf{tuple}_{2k}(\vec{x}_k, \vec{y}_k, z) = \mathsf{tuple}_k(\vec{x}_k, z) \cdot \mathsf{tuple}_k(\vec{y}_k, z)$$
$$\mathsf{tuple}_{2k+1}(\vec{x}_k, \vec{y}_{k+1}, z) = \mathsf{tuple}_{k+1}(\vec{x}_k, \varepsilon, z) \cdot \mathsf{tuple}_{k+1}(\vec{y}_{k+1}, z)$$
$$\langle \vec{x}_k \rangle_k = \mathsf{tuple}_k\big(\vec{x}_k, \mathsf{max}^{\mathsf{l}}_k(\vec{x}_k)\big)$$
$$\pi^1_1(x) = x$$
$$\pi^{2k}_\ell(x) = \begin{cases} \pi^k_\ell(x\blacktriangleleft) & \text{if } \ell \leq k \\ \pi^k_{\ell-k}(\blacktriangleright x) & \text{if } \ell > k \end{cases}$$
$$\pi^{2k+1}_\ell(x) = \begin{cases} \pi^{k+1}_\ell(x\blacktriangleleft) & \text{if } \ell \leq k \\ \pi^{k+1}_{\ell-k}(\blacktriangleright x) & \text{if } \ell > k \end{cases}$$

Let us now introduce a generalization of CRN where the recursion is defined on several variables at once. (We assume that $x_1, \ldots, x_m$ all have the same length, or are appropriately padded on the left with 0's to make them all the same length.)

CRN$_m$: f is defined from g and h by CRN$_m$ on $x_1, \ldots, x_m$ if $h(\vec{i}_m, \vec{x}_m, \vec{y}) = 0, 1$ for $i_1 = 0, 1; \ldots; i_m = 0, 1$ and

$$f(\vec{\varepsilon}_m, \vec{y}) = g(\vec{y})$$
$$f(x_1 i_1, \ldots, x_m i_m, \vec{y}) = f(x_1, \ldots, x_m, \vec{y}) \cdot h(i_1, \ldots, i_m, x_1, \ldots, x_m, \vec{y})$$

(We can also define *reverse* CRN$_m$ similarly to reverse CRN.) If f is defined from g and h by CRN$_m$ on $x_1, \ldots, x_m$, then we can define f using CRN as follows: We will define an auxiliary function aux_f by CRN on a parameter z; this function will mimic the recursion on $x_1, \ldots, x_m$ by using lb and lc to extract the correct substrings of $x_1, \ldots, x_m$ based on the length of z. Then, f is easily defined from aux_f by COMP.

$$\mathsf{aux}\_f(\varepsilon, \vec{x}_m, \vec{y}) = g(\vec{y})$$
$$\mathsf{aux}\_f(zi, \vec{x}_m, \vec{y}) = \mathsf{aux}\_f(z, \vec{x}_m, \vec{y}) \cdot h\big(\mathsf{lb}^{\mathsf{b}}(x_1, zi), \ldots, \mathsf{lb}^{\mathsf{b}}(x_m, zi), \mathsf{lc}(x_1, z), \ldots, \mathsf{lc}(x_m, z), \vec{y}\big)$$
$$f(\vec{x}_m, \vec{y}) = \mathsf{aux}\_f\big(\mathsf{max}^{\mathsf{l}}_m(\vec{x}_m), \mathsf{lp}_0(x_1, \mathsf{max}^{\mathsf{l}}_m(\vec{x}_m)), \ldots, \mathsf{lp}_0(x_m, \mathsf{max}^{\mathsf{l}}_m(\vec{x}_m)), \vec{y}\big)$$

Using CRN$_m$, we can now easily define the following useful functions, which perform bitwise operations on their arguments.

$$\mathsf{not}^{\mathsf{b}}(\varepsilon) = \varepsilon \qquad \mathsf{not}^{\mathsf{b}}(xi) = \mathsf{not}^{\mathsf{b}}(x) \cdot \neg^{\mathsf{b}} i$$
$$\mathsf{and}^{\mathsf{b}}_k(\vec{\varepsilon}_k) = \varepsilon \qquad \mathsf{and}^{\mathsf{b}}_k(x_1 i_1, \ldots, x_k i_k) = \mathsf{and}^{\mathsf{b}}_k(x_1, \ldots, x_k) \cdot (i_1 \wedge^{\mathsf{b}} \cdots \wedge^{\mathsf{b}} i_k)$$
$$\mathsf{or}^{\mathsf{b}}_k(\vec{\varepsilon}_k) = \varepsilon \qquad \mathsf{or}^{\mathsf{b}}_k(x_1 i_1, \ldots, x_k i_k) = \mathsf{or}^{\mathsf{b}}_k(x_1, \ldots, x_k) \cdot (i_1 \vee^{\mathsf{b}} \cdots \vee^{\mathsf{b}} i_k)$$
$$\mathsf{xor}^{\mathsf{b}}_k(\vec{\varepsilon}_k) = \varepsilon \qquad \mathsf{xor}^{\mathsf{b}}_k(x_1 i_1, \ldots, x_k i_k) = \mathsf{xor}^{\mathsf{b}}_k(x_1, \ldots, x_k) \cdot (i_1 \oplus^{\mathsf{b}} \cdots \oplus^{\mathsf{b}} i_k)$$
$$\mathsf{iff}^{\mathsf{b}}_k(\vec{\varepsilon}_k) = \varepsilon \qquad \mathsf{iff}^{\mathsf{b}}_k(x_1 i_1, \ldots, x_k i_k) = \mathsf{iff}^{\mathsf{b}}_k(x_1, \ldots, x_k) \cdot \big((i_1 =^{\mathsf{b}} i_2) \wedge^{\mathsf{b}} \cdots \wedge^{\mathsf{b}} (i_{k-1} =^{\mathsf{b}} i_k)\big)$$

Following Buss [8], we can now define functions which implement *carry-save addition*: CScar to compute the carry bits and CSadd to compute the addition bits.

$$\mathsf{CScar}_3(\varepsilon, \varepsilon, \varepsilon) = 0$$
$$\mathsf{CScar}_3(i_1 x_1, i_2 x_2, i_3 x_3) = \big((i_1 \wedge^{\mathsf{b}} i_2) \vee^{\mathsf{b}} (i_2 \wedge^{\mathsf{b}} i_3) \vee^{\mathsf{b}} (i_3 \wedge^{\mathsf{b}} i_1)\big) \cdot \mathsf{CScar}_3(x_1, x_2, x_3)$$
$$\mathsf{CSadd}_3(x_1, x_2, x_3) = \mathsf{xor}^{\mathsf{b}}_3(0x_1, 0x_2, 0x_3)$$
$$\mathsf{CScar}(x_1, x_2, x_3, x_4) = \mathsf{CScar}_3\big(\mathsf{CScar}_3(x_1, x_2, x_3), \mathsf{CSadd}_3(x_1, x_2, x_3), 0x_4\big)$$
$$\mathsf{CSadd}(x_1, x_2, x_3, x_4) = \mathsf{CSadd}_3\big(\mathsf{CScar}_3(x_1, x_2, x_3), \mathsf{CSadd}_3(x_1, x_2, x_3), 0x_4\big)$$

When a function $f(x, \vec{y})$ is defined by CRN on x from g and h, every bit in the output corresponds to one bit from x (with the exception of the first $|g(\vec{y})|$ bits). Now, we will show how to define a function where every bit of x corresponds to two bits in the output, and then generalize this to arbitrary values (where every group of 2^k bits in the input corresponds to a group of 2^n bits in the output, which we will call $(2^n, 2^k)$-CRN).

Following the notation mentioned above, we say that a function f is defined from g and h by $(2,1)$-*CRN* on x if $|h(i, x, \vec{y})| = 2$ for all $i \in \{0, 1\}$ and

$$f(x, \vec{y}) = \begin{cases} g(\vec{y}) & \text{if } x = \varepsilon, \\ f(x\blacktriangleleft, \vec{y}) \cdot h(x', x\blacktriangleleft, \vec{y}) & \text{otherwise.} \end{cases}$$

(We can also define *reverse* $(2,1)$-CRN.) If $f(x, \vec{y})$ is defined from g and h by $(2,1)$-CRN on x, we can define f using CRN as follows: We will first define an auxiliary function aux_$f(z, x, \vec{y})$ by CRN on z, to return the $|g(\vec{y})|+|z|$ leftmost bits of $f(x, \vec{y})$ and then define f from aux_f by COMP. Intuitively, aux_$f(z, x, \vec{y})$ uses $\mathsf{div}^{\mathsf{l}}_2(z)$ to

determine which bits of x to give as input to h and $\mathsf{mod}^{\mathsf{l}}_2(z)$ to determine which bit of h to output next.

$$\begin{aligned}\mathsf{aux\_}f(\varepsilon, x, \vec{y}) &= g(\vec{y})\\ \mathsf{aux\_}f(zi, x, \vec{y}) &= \mathsf{aux\_}f(z, x, \vec{y}) \cdot \\ &\qquad \mathsf{lb}^{\mathsf{b}}\big(h\big(\mathsf{lb}^{\mathsf{b}}(x, \mathsf{div}^{\mathsf{l}}_2(z)\cdot i), \mathsf{lc}(x, \mathsf{div}^{\mathsf{l}}_2(z)), \vec{y}\big), \mathsf{mod}^{\mathsf{l}}_2(z)\cdot i\big)\\ f(x, \vec{y}) &= \mathsf{aux\_}f(x\cdot x, x, \vec{y})\end{aligned}$$

Now, we can introduce the generalization mentioned above.

$(2^n, 2^k)$-CRN: f is defined from g and h by $(2^n, 2^k)$-CRN on x if $|h(z, x, \vec{y})| = 2^n$ for all $z \in \{0,1\}^{2^k}$ and

$$f(x, \vec{y}) = \begin{cases} g(x, \vec{y}) & \text{if } |x| < 2^k\\ f(x \lhd \widehat{2^k}, \vec{y}) \cdot h\big(\mathsf{rc}(x, \widehat{2^k}), x \lhd \widehat{2^k}, \vec{y}\big) & \text{otherwise}\end{cases}$$

(As before, we can also define *reverse* $(2^n, 2^k)$-CRN.) If f is defined from g and h by $(2^n, 2^k)$-CRN on x, then we can define f using CRN as follows. (The intuition is similar to that for $(2,1)$-CRN given above.)

$$\begin{aligned}\mathsf{aux\_}f(\varepsilon, x, \vec{y}) &= g(\mathsf{lmod}^{\mathsf{l}}_{2^k}(x), \vec{y})\\ \mathsf{aux\_}f(zi, x, \vec{y}) &= \mathsf{aux\_}f(z, x, \vec{y}) \cdot \mathsf{lb}^{\mathsf{b}}\Big(h\Big(\mathsf{rc}\big(\mathsf{lc}\big(x, \mathsf{mod}^{\mathsf{l}}_{2^k}(x)\cdot 2^k \times (\mathsf{div}^{\mathsf{l}}_{2^n}(z)\cdot i)\big), \widehat{2^k}\big),\\ &\qquad \mathsf{lc}\big(x, \mathsf{mod}^{\mathsf{l}}_{2^k}(x)\cdot 2^k \times \mathsf{div}^{\mathsf{l}}_{2^n}(z)\big), \vec{y}\Big), \mathsf{mod}^{\mathsf{l}}_{2^n}(z)\cdot i\Big)\\ f(x, \vec{y}) &= \mathsf{aux\_}f(2^n \times \mathsf{div}^{\mathsf{l}}_{2^k}(x), x, \vec{y})\end{aligned}$$

By combining the two generalizations above, we can show that any function defined by $(2^n, 2^k)$-CRN$_m$ can be defined using CRN and COMP alone. Using this very general form of CRN, we can now easily define the following functions: rins_j (resp. lins_j) inserts a "j" to the right (resp. to the left) of every bit of its argument; rmix_{2^k} (resp. lmix_{2^k}) interleaves the bits of its arguments starting with the rightmost bit (resp. with the leftmost bit); $\mathsf{rsel}^{2^k}_\ell$ (resp. $\mathsf{lsel}^{2^k}_\ell$) extracts every ℓ-th bit from 2^k interleaved strings, starting from the right (resp. starting from the left), so that $\mathsf{rsel}^{2^k}_\ell(\mathsf{rmix}_{2^k}(x_1, \ldots, x_{2^k})) = x_\ell$ (assuming all the x_i's have the same length) and similarly for $\mathsf{lsel}^{2^k}_\ell$ and lmix_{2^k}. Note that rmix_{2^k} and lmix_{2^k} produce the same output when their arguments all have the same length; they are different only when some arguments are shorter than others.

$$\begin{aligned}\mathsf{rins}_j(\varepsilon) &= \varepsilon \qquad \mathsf{rins}_j(xi) = \mathsf{rins}_j(x)\cdot ij\\ \mathsf{lins}_j(\varepsilon) &= \varepsilon \qquad \mathsf{lins}_j(ix) = ji\cdot \mathsf{lins}_j(x)\\ \mathsf{rmix}_{2^k}(\vec{\varepsilon}_{2^k}) &= \varepsilon \qquad \mathsf{rmix}_{2^k}(x_1 i_1, \ldots, x_{2^k} i_{2^k}) = \mathsf{rmix}_{2^k}(x_1, \ldots, x_{2^k})\cdot i_1 \cdots i_{2^k}\\ \mathsf{lmix}_{2^k}(\vec{\varepsilon}_{2^k}) &= \varepsilon \qquad \mathsf{lmix}_{2^k}(i_1 x_1, \ldots, i_{2^k} x_{2^k}) = i_1 \cdots i_{2^k}\cdot \mathsf{lmix}_{2^k}(x_1, \ldots, x_{2^k})\end{aligned}$$

$$\mathsf{rsel}^{2^k}_\ell(x) = \begin{cases} \mathsf{rb}(x, \widehat{\ell}) & \text{if } |x| < 2^k\\ \mathsf{rsel}^{2^k}_\ell(x \lhd \widehat{2^k})\cdot \mathsf{rb}^{\mathsf{b}}(\mathsf{rc}(x, \widehat{2^k}), \widehat{\ell}) & \text{otherwise}\end{cases}$$

$$\mathsf{lsel}^{2^k}_\ell(x) = \begin{cases} \mathsf{lb}(x, \widehat{\ell}) & \text{if } |x| < 2^k\\ \mathsf{lb}^{\mathsf{b}}(\mathsf{lc}(x, \widehat{2^k}), \widehat{\ell})\cdot \mathsf{lsel}^{2^k}_\ell(\widehat{2^k} \rhd x) & \text{otherwise}\end{cases}$$

2.3. Functions in TRN$[L_0]$. Before defining any particular functions in TRN$[L_0]$, let us introduce a generalization of TRN.

Because we cannot define a function by TRN from functions that are themselves defined by TRN, it will be useful to be able to define more than one function simultaneously by TRN.

TRN$_k$: The functions f_i $(1 \le i \le k)$ are defined from functions g_i, h_i, $\vec{l}$, and $\vec{r}$ by TRN$_k$ on x if for every $1 \le i \le k$,

$$f_i(x, \vec{y}) = \begin{cases} g_i(x, \vec{y}) & \text{if } x = \varepsilon, 0, 1,\\ h_i\big(x, f_1(x\blacktriangleleft, \vec{l}(x\blacktriangleleft, \vec{y})), f_1(\blacktriangleright x, \vec{r}(\blacktriangleright x, \vec{y})), \ldots,\\ \qquad f_k(x\blacktriangleleft, \vec{l}(x\blacktriangleleft, \vec{y})), f_k(\blacktriangleright x, \vec{r}(\blacktriangleright x, \vec{y})), \vec{y}\big) & \text{otherwise.}\end{cases}$$

If $\vec{f}_k$ are defined from $\vec{g}_k$, $\vec{h}_k$, $\vec{l}$, and $\vec{r}$ by TRN$_k$, we can define the k-tuple $F(x, \vec{y}) = \langle \vec{f}_k\rangle_k$ by TRN as follows.

$$F(x, \vec{y}) = \begin{cases} \langle \vec{g}_k(x, \vec{y})\rangle_k & \text{if } x = \varepsilon, 0, 1,\\ \big\langle \vec{h}_k\big(x, \pi^k_1(F(x\blacktriangleleft, \vec{l}(x\blacktriangleleft, \vec{y}))), \pi^k_1(F(\blacktriangleright x, \vec{r}(\blacktriangleright x, \vec{y}))), \ldots,\\ \qquad \pi^k_k(F(x\blacktriangleleft, \vec{l}(x\blacktriangleleft, \vec{y}))), \pi^k_k(F(\blacktriangleright x, \vec{r}(\blacktriangleright x, \vec{y}))), \vec{y}\big)\big\rangle_k & \text{otherwise.}\end{cases}$$

Then, a simple composition gives $f_i(x, \vec{y}) = \pi^k_i(F(x, \vec{y}))$ for $1 \le i \le k$. Note that technically speaking, the functions f_i are not in TRN$[L_0]$ but in L_1 because COMP is needed to define them.

Now here are some functions in TRN$[L_0]$. The first four perform Boolean operations on all the bits of their input, $|x|$ returns the length of x, expressed as a binary number, while $x \# y$ returns $|y|$ copies of x concatenated together (so that $|x \# y| = |x| \times |y|$). The next two functions are defined by TRN$_2$ (so they are not in TRN$[L_0]$ directly) and generalize $\mathsf{div}^{\mathsf{l}}_{2^k}$ and $\mathsf{mod}^{\mathsf{l}}_{2^k}$ so that $|\mathsf{div}^{\mathsf{l}}(x, y)| = |x|/|y|$ (integer division) and $|\mathsf{mod}^{\mathsf{l}}(x, y)| = |x| \bmod |y|$ (in the definition below, $\mathsf{aux\_mod}^{\mathsf{l}}(x, y) = x \lhd y\ ?^{\mathsf{l}}\ (x, x \lhd y)$ belongs to L_0). The last two functions are also defined by TRN$_2$ and count the number of 1's in the string x, using the carry-save technique of Buss [8].

$$\mathsf{AND}(x) = \begin{cases} x\ ?\ (1, 0, 1) & \text{if } x = \varepsilon, 0, 1\\ \mathsf{AND}(x\blacktriangleleft) \wedge^{\mathsf{b}} \mathsf{AND}(\blacktriangleright x) & \text{otherwise}\end{cases}$$

$$\mathsf{OR}(x) = \begin{cases} \approx^{\mathsf{b}} x & \text{if } x = \varepsilon, 0, 1\\ \mathsf{OR}(x\blacktriangleleft) \vee^{\mathsf{b}} \mathsf{OR}(\blacktriangleright x) & \text{otherwise}\end{cases}$$

$$\mathsf{XOR}(x) = \begin{cases} \approx^{\mathsf{b}} x & \text{if } x = \varepsilon, 0, 1\\ \mathsf{XOR}(x\blacktriangleleft) \oplus^{\mathsf{b}} \mathsf{XOR}(\blacktriangleright x) & \text{otherwise}\end{cases}$$

$$\mathsf{IFF}(x) = \begin{cases} 1 & \text{if } x = \varepsilon, 0, 1\\ \big(\mathsf{IFF}(x\blacktriangleleft) \wedge^{\mathsf{b}} \mathsf{IFF}(\blacktriangleright x)\big) \wedge^{\mathsf{b}} (`x =^{\mathsf{b}} x') & \text{otherwise}\end{cases}$$

$$|x| = \begin{cases} {}_1x & \text{if } x = \varepsilon, 0, 1\\ \mathsf{mod}^{\mathsf{l}}_2(x)\ ?^{\mathsf{l}}\ \big(|x\blacktriangleleft|\cdot 0, |x\blacktriangleleft|\cdot 1\big) & \text{otherwise}\end{cases}$$

$$x \# y = \begin{cases} y\ ?^{\mathsf{l}}\ (\varepsilon, x) & \text{if } y = \varepsilon, 0, 1\\ (x \# y\blacktriangleleft)\cdot (x \# \blacktriangleright y) & \text{otherwise}\end{cases}$$

$$\mathsf{mod}^{\mathsf{l}}(x,y) = \begin{cases} \mathsf{aux\_mod}^{\mathsf{l}}({}_1x, y) & \text{if } x = \varepsilon, 0, 1 \\ \mathsf{aux\_mod}^{\mathsf{l}}\big(\mathsf{mod}^{\mathsf{l}}(x\blacktriangleleft, y) \cdot \mathsf{mod}^{\mathsf{l}}(\blacktriangleright x, y), y\big) & \text{otherwise} \end{cases}$$

$$\mathsf{div}^{\mathsf{l}}(x,y) = \begin{cases} x\ ?^{\mathsf{l}}\ \big(\varepsilon, y\ ?^{\mathsf{l}}\ (\varepsilon, {}_1x \lhd y\!<)\big) & \text{if } x = \varepsilon, 0, 1 \\ \big(\mathsf{div}^{\mathsf{l}}(x\blacktriangleleft, y) \cdot \mathsf{div}^{\mathsf{l}}(\blacktriangleright x, y)\big) \cdot \\ \quad \big((\mathsf{mod}^{\mathsf{l}}(x\blacktriangleleft, y) \cdot \mathsf{mod}^{\mathsf{l}}(\blacktriangleright x, y)) \lhd y\ ?^{\mathsf{l}}\ (\varepsilon, 1)\big) & \text{otherwise} \end{cases}$$

$$\mathsf{CAR}(x) = \begin{cases} 0 & \text{if } x = \varepsilon, 0, 1 \\ \mathsf{CScar}(\mathsf{CAR}(x\blacktriangleleft), \mathsf{CAR}(\blacktriangleright x), \mathsf{ADD}(x\blacktriangleleft), \mathsf{ADD}(\blacktriangleright x)) & \text{otherwise} \end{cases}$$

$$\mathsf{ADD}(x) = \begin{cases} \approx^{\mathsf{b}} x & \text{if } x = \varepsilon, 0, 1 \\ \mathsf{CSadd}(\mathsf{CAR}(x\blacktriangleleft), \mathsf{CAR}(\blacktriangleright x), \mathsf{ADD}(x\blacktriangleleft), \mathsf{ADD}(\blacktriangleright x)) & \text{otherwise} \end{cases}$$

With $\mathsf{div}^{\mathsf{l}}$ and $\mathsf{mod}^{\mathsf{l}}$, we can now define (s,t)-CRN similarly to $(2^n, 2^k)$-CRN but where blocks of $|t(\vec{y})|$ input bits correspond to blocks of $|s(\vec{y})|$ output bits, for arbitrary $s, t \in L_1$ (and where $\vec{y}$ are the fixed input parameters).

2.4. Functions in L_1. One nice property of all functions in L_1 is that they are *length-determined*, *i.e.*, the length of the output is uniquely determined by the lengths of the inputs (and not their value). This makes it more natural to think of our functions as being computed by circuits and simplifies the definitions of many functions which perform "string manipulation" operations on their inputs. Unfortunately, it also makes it harder to define some functions which treat their inputs as numbers (*i.e.*, ignoring leading 0's). For example, the definition of $|x|$ give above is quite simple whereas the definition of $|x|^{\mathsf{n}}$ given below relies on some more complex functions.

The first functions we will define are equality for numbers and for strings.

$$x =^{\mathsf{n}} y = \mathsf{AND}(\mathsf{iff}_2^{\mathsf{b}}(x,y)) \qquad x =^{\mathsf{s}} y = (x =^{\mathsf{l}} y) \wedge^{\mathsf{b}} (x =^{\mathsf{n}} y)$$

Most of the functions that we will define next are *numerical*, *i.e.*, treat their arguments as numbers. First, let us define a successor and a predecessor function.

$$\mathsf{aux\_succ}^{\mathsf{n}}(\varepsilon) = \varepsilon \qquad \mathsf{aux\_succ}^{\mathsf{n}}(ix) = \mathsf{AND}(x)\ ?^{\mathsf{b}}\ (\neg^{\mathsf{b}} i, i) \cdot \mathsf{aux\_succ}^{\mathsf{n}}(x)$$

$$\mathsf{succ}^{\mathsf{n}}(x) = \mathsf{aux\_succ}^{\mathsf{n}}(0x)$$

$$\mathsf{pred}^{\mathsf{n}}(\varepsilon) = \varepsilon \qquad \mathsf{pred}^{\mathsf{n}}(ix) = \mathsf{OR}(x)\ ?^{\mathsf{b}}\ (i, \neg^{\mathsf{b}} i) \cdot \mathsf{pred}^{\mathsf{n}}(x)$$

Note that one unfortunate side-effect of the fact that the functions are length-determined is that the successor function always appends a bit to the left of its argument. So starting from ε and applying $\mathsf{succ}^{\mathsf{n}}$ repeatedly, we get a series of strings which represent $0, 1, 2, \ldots$ in binary, but whose lengths are also $0, 1, 2, \ldots$ Next, we define the numerical predicate $<^{\mathsf{n}}$.

$$\mathsf{less}^{\mathsf{n}}(\varepsilon, \varepsilon) = \varepsilon \qquad \mathsf{less}^{\mathsf{n}}(xi, yj) = \mathsf{less}^{\mathsf{n}}(x,y) \cdot \big((i <^{\mathsf{b}} j) \wedge^{\mathsf{b}} (x =^{\mathsf{n}} y)\big)$$

$$x <^{\mathsf{n}} y = \mathsf{OR}(\mathsf{less}^{\mathsf{n}}(x,y))$$

Together with $=^{\mathsf{n}}$, this allows us to define all other relational operators on numbers using the Boolean connectives. Note that when we defined $\mathsf{less}^{\mathsf{n}}$, we used Clote's "programming trick" [10] of making a sweep through the bits of the strings x and y, appending a 1 when some condition is met so that the final composition with OR yields 1 iff the condition was met at some position. Using AND, we can similarly define functions that test for some condition on every bit of their inputs.

The next function we want to define is $\mathsf{bit}^{\mathsf{n}}(x,z)$, which returns bit number z of x, from the right, where z is interpreted as a binary number. The easiest way to do this is by defining a function $\mathsf{pow}^{\mathsf{n}}(z,x)$ which returns a string of length $|x|$ consisting entirely of 0's except at bit position z (from the right), if $z \leq |x|$ (where z is interpreted as a binary number). Then, we define a function $\mathsf{maskbit}^{\mathsf{n}}(x,y)$ which treats y as a mask to determine which bit of x to return.

$$\mathsf{pow}^{\mathsf{n}}(z,\varepsilon) = \varepsilon \qquad \mathsf{pow}^{\mathsf{n}}(z, ix) = \big(|ix| =^{\mathsf{n}} z\big) \cdot \mathsf{pow}^{\mathsf{n}}(z,x)$$

$$\mathsf{maskbit}^{\mathsf{n}}(x,y) = \mathsf{OR}(\mathsf{and}_2^{\mathsf{b}}(x,y)) \qquad \mathsf{bit}^{\mathsf{n}}(x,z) = \mathsf{maskbit}^{\mathsf{n}}(x, \mathsf{pow}^{\mathsf{n}}(z,x))$$

Next, we want to define addition. This will require some work, especially to compute the carry bits. The function $\mathsf{mask}_1(x)$ simulates a function which strips leading ones from x by returning a bit mask for x which is zero exactly on the leading ones in x. The functions $\mathsf{first}_j(x)$ return a mask which is 1 on the leftmost bit of x equal to j and 0 elsewhere.

$$\mathsf{mask}_1(\varepsilon) = \varepsilon \qquad \mathsf{mask}_1(xi) = \mathsf{mask}_1(x) \cdot \neg^{\mathsf{b}}\mathsf{AND}(xi)$$

$$\mathsf{first}_1(\varepsilon) = \varepsilon \qquad \mathsf{first}_1(xi) = \mathsf{first}_1(x) \cdot \big(\mathsf{OR}(x)\ ?^{\mathsf{b}}\ (0,i)\big)$$

$$\mathsf{first}_0(x) = \mathsf{first}_1(\mathsf{mask}_1(x))$$

Then, we can define the function which computes the carry bits and the addition function itself, as follows.

$$\mathsf{carry}^{\mathsf{n}}(\varepsilon,\varepsilon) = 0$$

$$\mathsf{carry}^{\mathsf{n}}(ix,jy) = \mathsf{maskbit}^{\mathsf{n}}\big(\mathsf{and}_2^{\mathsf{b}}(ix,jy), \mathsf{first}_0(\mathsf{xor}_2^{\mathsf{b}}(ix,jy))\big) \cdot \mathsf{carry}^{\mathsf{n}}(x,y)$$

$$\mathsf{add}^{\mathsf{n}}(\varepsilon,\varepsilon) = \varepsilon$$

$$\mathsf{add}^{\mathsf{n}}(ix,jy) = \big(\mathsf{carry}^{\mathsf{n}}(x,y) \oplus^{\mathsf{b}} (i \oplus^{\mathsf{b}} j)\big) \cdot \mathsf{add}^{\mathsf{n}}(x,y)$$

$$x +^{\mathsf{n}} y = \mathsf{add}^{\mathsf{n}}(0x, 0y)$$

Finally, using the addition function, we can define a function which counts the number of ones in a string: $\mathsf{sum}(x) = \mathsf{CAR}(x) +^{\mathsf{n}} \mathsf{ADD}(x)$ and using this function, define the numerical length of x: $|x|^{\mathsf{n}} = \mathsf{sum}(\mathsf{mask}_1(\mathsf{not}^{\mathsf{b}}(x)))$. (Note that we could also have defined $|x| = \mathsf{sum}({}_1x)$ instead of directly using TRN.)

2.5. L_1 and NC^1. In this section, we prove the following claim.

CLAIM 2.3. $L_1 = FNC^1$.

PROOF. For an n-ary function $f \in FNC^1$, let $C_{\vec{m}}(f)$ be the multi-output NC^1 circuit which computes f on inputs of length $\vec{m} = m_1, \ldots, m_n$, and define the following symbols:

- $\mathsf{in}_{\vec{m}}(f) = \sum \vec{m}$ is the number of input gates of $C_{\vec{m}}(f)$.
- $\mathsf{out}_{\vec{m}}(f)$ is the number of output gates of $C_{\vec{m}}(f)$.
- $\mathsf{size}_{\vec{m}}(f)$ is the number of inner gates of $C_{\vec{m}}(f)$;
- $\mathsf{depth}_{\vec{m}}(f)$ is the depth of $C_{\vec{m}}(f)$;

Then, it is clear that for all $f \in \mathrm{BASE}$, there are simple NC^1 circuits which compute f such that

$$\mathsf{out}_{\vec{m}}(f) \leq \max\{\vec{m}\} + 1,$$

$$\mathsf{size}_{\vec{m}}(f) \leq 4 \max\{\vec{m}\},$$

$$\mathsf{depth}_{\vec{m}}(f) \leq 4.$$

REMARK 2.4. Note that among the base functions, the conditional "?" is the only function requiring circuits of non-zero size: all the other functions can be computed by circuits that connect input nodes to output nodes directly.

Moreover, if f is defined by COMP from g and $h_1, \ldots, h_k$, then an NC^1 circuit which computes f can easily be constructed from the circuits for g and $h_1, \ldots, h_k$ in the obvious way, such that

$$\begin{aligned}
\mathrm{out}_{\vec{m}}(f) &= \mathrm{out}_{\mathrm{out}_{\vec{m}}(h_1),\ldots,\mathrm{out}_{\vec{m}}(h_k)}(g),\\
\mathrm{size}_{\vec{m}}(f) &= \mathrm{size}_{\mathrm{out}_{\vec{m}}(h_1),\ldots,\mathrm{out}_{\vec{m}}(h_k)}(g) + \textstyle\sum_{1\le i\le k}\mathrm{size}_{\vec{m}}(h_i),\\
\mathrm{depth}_{\vec{m}}(f) &\le \mathrm{depth}_{\mathrm{out}_{\vec{m}}(h_1),\ldots,\mathrm{out}_{\vec{m}}(h_k)}(g) + \max\{\mathrm{depth}_{\vec{m}}(h_1),\ldots,\mathrm{depth}_{\vec{m}}(h_k)\}.
\end{aligned}$$

Also, if f is defined by CRN from g and h, then it is easy to construct an NC^1 circuit for f by wiring enough copies of the circuit for h in parallel, along with the circuit for g, such that

$$\begin{aligned}
\mathrm{out}_{m,\vec{m}}(f) &= \mathrm{out}_{\vec{m}}(g) + m,\\
\mathrm{size}_{m,\vec{m}}(f) &= \mathrm{size}_{\vec{m}}(g) + \textstyle\sum_{0\le i<m}\mathrm{size}_{1,i,\vec{m}}(h),\\
\mathrm{depth}_{m,\vec{m}}(f) &= \max\{\mathrm{depth}_{\vec{m}}(g), \mathrm{depth}_{1,0,\vec{m}}(h), \ldots, \mathrm{depth}_{1,m-1,\vec{m}}(h)\}.
\end{aligned}$$

Also, note that because the function h must output a single bit on all valid inputs, the size of the circuit computing h is constant (the circuit has constant depth and can only depend on constantly many input bits). Therefore, for all $f \in L_0$, it is easy to construct a uniform family of NC^1 circuits for f such that

$$\begin{aligned}
\mathrm{out}_{\vec{m}}(f) &= \mathcal{O}(\mathrm{in}_{\vec{m}}(f)),\\
\mathrm{size}_{\vec{m}}(f) &= \mathcal{O}(\mathrm{in}_{\vec{m}}(f)),\\
\mathrm{depth}_{\vec{m}}(f) &= \mathcal{O}(1).
\end{aligned}$$

Now, if f is defined by TRN from g, h, $\vec{l}$, and $\vec{r}$, then a straightforward, if tedious, construction yields uniform NC^1 circuits for f such that, for some constants k_1, k_2 depending on f,

$$\begin{aligned}
\mathrm{out}_{m,\vec{m}}(f) &\le \mathcal{O}(m^{k_1}\,\mathrm{in}_{m,\vec{m}}(f)),\\
\mathrm{size}_{m,\vec{m}}(f) &\le \mathcal{O}(m\,\mathrm{in}_{m,\vec{m}}(f)^{k_2}),\\
\mathrm{depth}_{m,\vec{m}}(f) &\le \mathcal{O}(\log m).
\end{aligned}$$

Therefore, $L_1 \subseteq FNC^1$. (Note that the technique used by Bloch in [6] can be extended to show the uniformity of all the circuits mentioned above.)

To prove the inclusion in the other direction, we show how to simulate the computation of an ATM using functions in L_1. Without loss of generality, let the ATM have the following properties.

1. There is a function $t \in L_1$ such that on inputs $\vec{x}$, the ATM runs for no more than $|t(\vec{x})|$ steps. Moreover, $2^{2|t(\vec{x})|}$ is greater than the number of states of the ATM for any inputs $\vec{x}$ (in other words, a string of length $2|t(\vec{x})|$ is long enough to encode the state of the ATM).
2. The ATM has n read-only input tapes represented by strings $x_1, \ldots, x_n$ and k worktapes represented by pairs of strings $y_1^l, y_1^r, \ldots, y_k^l, y_k^r$, each of length exactly $2|t(\vec{x})|$, where y_i^l represents the content of tape number i to the left of the tape head and y_i^r represents the content to the right, with the head scanning the rightmost symbol of y_i^l. Each of the three possible worktape symbols (1, 0, or blank) is encoded using two bits (11 for 1, 10 for 0, and 00 for blank). Initially, the worktapes are blank.
3. The computation tree of the ATM is a complete binary tree (each non-leaf node had exactly two successor configurations, a left successor and a right successor).
4. Access to the input occurs only at the leaves of the computation tree and is of the form "accept iff symbol number $y_i^l y_i^r$ (interpreted as a binary number) on input tape number j is equal to b", where i, j, and b are encoded in the current state of the ATM.

Then, if we let $\mathrm{CON} = \langle s, y_1^l, y_1^r, \ldots, y_k^l, y_k^r\rangle_{2k+1}$ represent a configuration of the ATM when in state s, we can define $l\mathrm{CON}$ and $r\mathrm{CON}$, the left and right successor configurations of CON, as follows (where the tuples are $(2k+1)$-tuples):

$$\begin{aligned}
l\mathrm{CON} &= \left\langle \mathsf{lstate}(\mathrm{CON}), \mathsf{ltape}_1^l(\mathrm{CON}), \mathsf{ltape}_1^r(\mathrm{CON}), \ldots, \mathsf{ltape}_k^l(\mathrm{CON}), \mathsf{ltape}_k^r(\mathrm{CON})\right\rangle,\\
r\mathrm{CON} &= \left\langle \mathsf{rstate}(\mathrm{CON}), \mathsf{rtape}_1^l(\mathrm{CON}), \mathsf{rtape}_1^r(\mathrm{CON}), \ldots, \mathsf{rtape}_k^l(\mathrm{CON}), \mathsf{rtape}_k^r(\mathrm{CON})\right\rangle,
\end{aligned}$$

where each of $\mathsf{lstate}, \mathsf{ltape}^l, \mathsf{ltape}^r, \mathsf{rstate}, \mathsf{rtape}^l, \mathsf{rtape}^r$ is easily seen to be in L_0, involving only simple string manipulations and finite table lookup on the state s (for example, $\mathsf{ltape}_i^l(\mathrm{CON}) = d_i \mathbin{?^{\mathsf{b}}} (00 \rhd ((y_i^l \lhd 00 \cdot b_i) \cdot \mathsf{lc}(y_i^r, 00)), (00 \cdot y_i^l) \lhd 00)$ computes the contents of tape i to the left of the head in the left successor of CON, where d_i (the direction of movement for head i) and b_i (the tape symbol to write on tape i) are obtained from the state and tape contents of CON using the conditional function). Moreover, the function $\mathsf{input}(\mathrm{CON}, \vec{x}) = \mathsf{bit}^{\mathsf{n}}(x_j, \mathsf{rsel}_2^2(y_i^l y_i^r)) =^{\mathsf{b}} b$ (where i, j, and b are extracted from the state s) is equal to the accept state of the given input configuration and is in L_1. Finally, we let CON_0 denote the initial configuration.

Now, given a string z which lists the type of each configuration in the computation tree according to an "in-order" traversal[1] (using 0 for universal configurations and 1 for existential), the following function outputs the concatenation of every leaf configuration in the computation tree of the ATM, given z and CON_0 as input.

$$\mathsf{comptree}(z, \mathrm{CON}) = \begin{cases} \mathrm{CON} & \text{if } z = \varepsilon, 0, 1,\\ \mathsf{comptree}(z\blacktriangleleft, l\mathrm{CON}) \cdot \mathsf{comptree}(\blacktriangleright z, r\mathrm{CON}) & \text{otherwise.}\end{cases}$$

Then, we can use the generalization of CRN mentioned at the end of Section 2.3 to define a function inputstr that replaces each leaf configuration in $\mathsf{comptree}(z, \mathrm{CON}_0)$ with its accept state according to $\mathsf{input}(\mathrm{CON}, \vec{x})$, as follows.

$$\mathsf{inputstr}(c, \vec{x}) = \begin{cases} \mathsf{input}(c, \vec{x}) & \text{if } |c| < \widehat{4k+2} \times |t(\vec{x})|,\\ \mathsf{inputstr}(c \lhd (\widehat{4k+2} \times |t(\vec{x})|), \vec{x}) \cdot & \\ \quad \mathsf{input}(\mathsf{rc}(c, \widehat{4k+2} \times |t(\vec{x})|), \vec{x}) & \text{otherwise.}\end{cases}$$

Finally, we define a function eval that uses the string z and the output of inputstr to evaluate the computation tree of the ATM.

$$\mathsf{eval}(z, u) = \begin{cases} u & \text{if } z = \varepsilon, 0, 1,\\ \setminus(\blacktriangleright z) \mathbin{?^{\mathsf{b}}} (\mathsf{eval}(z\blacktriangleleft, u\blacktriangleleft) \vee^{\mathsf{b}} \mathsf{eval}(\blacktriangleright z, \blacktriangleright u), & \\ \qquad \mathsf{eval}(z\blacktriangleleft, u\blacktriangleleft) \wedge^{\mathsf{b}} \mathsf{eval}(\blacktriangleright z, \blacktriangleright u)) & \text{otherwise.}\end{cases}$$

[1] For every node a with left child l and right child r, every node in the subtree rooted at l is listed before a and every node in the subtree rooted at r is listed after a.

In this way, the result of the computation of the ATM on inputs $\vec{x}$ is given by

$$\mathsf{eval}\big(z, \mathsf{inputstr}(\mathsf{comptree}(z, \mathrm{CON}_0), \vec{x})\big).$$

Note that if we impose on the ATM the additional condition that configurations in the computation tree strictly alternate between universal and existential, then we can generate z using the following function:

$$\mathsf{type}(x,y) = \begin{cases} \approx^{\mathsf{b}} y & \text{if } x = \varepsilon, 0, 1, \\ \mathsf{type}(x\blacktriangleleft, \neg^{\mathsf{b}} y) \cdot \approx^{\mathsf{b}} y \cdot \mathsf{type}(x\blacktriangleleft, \neg^{\mathsf{b}} y) & \text{otherwise,} \end{cases}$$

by giving it an appropriate function of $t(\vec{x})$ as first input (using repeated applications of $\#$ and $\cdot$ if necessary so that $\mathsf{type}(x,y)$ is long enough) and the type of the initial configuration as second input. □

3. The quantifier-free theory T_1

The theory T_1 which we will now describe is a *quantifier-free* system, *i.e.*, a free-variable theory with propositional connectives, modeled after Steve Cook's PV [15]. The language of T_1 consists of the function symbols

$$\{\varepsilon, 0, 1, {}_0, {}_1, \`, ', \gtrdot, \lessdot, \blacktriangleright, \blacktriangleleft, \cdot, \rhd, \lhd, ?\},$$

the function constructors $\{\lambda, \mathrm{CRN}, \mathrm{TRN}\}$, the predicate symbol $\{=\}$, and the usual propositional connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$. More precisely, we have the following definitions (where we use the informal notation $x0$ for $(x \cdot 0)$ and $0x$ for $(0 \cdot x)$—similarly for $x1$ and $1x$).

DEFINITION 3.1. The *function symbols* and *terms* of T_1 are defined as follows. (The intended interpretation of each function symbol is as given in Section 2, and we use the notation introduced there instead of the more formal prefix notation. Also, each function symbol and each term has a *rank* of either 0 or 1—which intuitively indicates which one of L_0 or L_1 the function symbol or term belongs to.)

1. Each variable $x_0, x_1, x_2, \ldots$ is a term of rank 0.
2. If f is an n-place function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term whose rank is the maximum of the ranks of $f, t_1, \ldots, t_n$ (*i.e.*, the rank of $f(t_1, \ldots, t_n)$ is 0 iff the rank of each one of $f, t_1, \ldots, t_n$ is 0).
3. $\varepsilon, 0, 1$ are 0-place function symbols (constants) of rank 0.
4. ${}_0, {}_1, \`, ', \gtrdot, \lessdot, \blacktriangleright, \blacktriangleleft$ are 1-place function symbols of rank 0.
5. $\cdot, \rhd, \lhd$ are 2-place function symbols of rank 0.
6. $?$ is a 3-place function symbol of rank 0.
7. If t is a term and $x_1, \ldots, x_n$ is a list of variables including all the variables in t, then $[\lambda x_1 \ldots x_n.t]$ is an n-place function symbol of the same rank as that of t.
8. If g is an n-place function symbol and h is an $(n+2)$-place function symbol, then $\mathrm{CRN}[g,h]$ is an $(n+1)$-place function symbol whose rank is the maximum of the ranks of g and h.
9. If g is an $(n+1)$-place function symbol of rank 0, h is an $(n+3)$-place function symbol of rank 0, and all of $l_1, \ldots, l_n, r_1, \ldots, r_n$ are $(n+1)$-place function symbols of rank 0, then $\mathrm{TRN}[g, h, \vec{l}, \vec{r}]$ is an $(n+1)$-place function symbol of rank 1.

DEFINITION 3.2. The *axioms* of T_1 are as follows (except for the propositional axioms, they simply define the function symbols).

0. Any standard, complete set of axioms for the propositional calculus (with equations of the form $t = u$ in place of propositional atoms, for some terms t and u).
1. $\varepsilon \neq 0 \qquad 0 \neq 1 \qquad 1 \neq \varepsilon$
2. (a) $\varepsilon \cdot x = x = x \cdot \varepsilon$
   (b) $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
   (c) $z \cdot x = z \cdot y \leftrightarrow x = y \leftrightarrow x \cdot z = y \cdot z$
   (d) $x \cdot y = \varepsilon \leftrightarrow (x = \varepsilon \wedge y = \varepsilon)$
   (e) $x \cdot y = 0 \leftrightarrow (x = \varepsilon \wedge y = 0) \vee (x = 0 \wedge y = \varepsilon)$
   $x \cdot y = 1 \leftrightarrow (x = \varepsilon \wedge y = 1) \vee (x = 1 \wedge y = \varepsilon)$
3. (a) ${}_0\varepsilon = \varepsilon \qquad {}_0(x0) = {}_0x \cdot 0 \qquad {}_0(x1) = {}_0x \cdot 0$
   (b) ${}_1\varepsilon = \varepsilon \qquad {}_1(x0) = {}_1x \cdot 1 \qquad {}_1(x1) = {}_1x \cdot 1$
4. (a) $\`\varepsilon = \varepsilon \qquad \`(0x) = 0 \qquad \`(1x) = 1$
   (b) $\varepsilon' = \varepsilon \qquad (x0)' = 0 \qquad (x1)' = 1$
5. (a) $\gtrdot\varepsilon = \varepsilon \qquad \gtrdot(0x) = x \qquad \gtrdot(1x) = x$
   (b) $\varepsilon\lessdot = \varepsilon \qquad (x0)\lessdot = x \qquad (x1)\lessdot = x$
6. (a) $\varepsilon \rhd x = x \qquad (0y) \rhd x = \gtrdot(y \rhd x) \qquad (1y) \rhd x = \gtrdot(y \rhd x)$
   (b) $x \lhd \varepsilon = x \qquad x \lhd (y0) = (x \lhd y)\lessdot \qquad x \lhd (y1) = (x \lhd y)\lessdot$
7. $\varepsilon\,?\,(x,y,z) = x$
   $(w0)\,?\,(x,y,z) = {}_0(z \lhd y) \cdot y$
   $(w1)\,?\,(x,y,z) = {}_0(y \lhd z) \cdot z$
8. (a) $(x\blacktriangleleft) \cdot (\blacktriangleright x) = x$
   (b) $(x\blacktriangleleft \lhd \blacktriangleright x) = \varepsilon \qquad \gtrdot(x\blacktriangleleft \rhd \blacktriangleright x) = \varepsilon$
   (c) $x\blacktriangleleft = x \lhd (\blacktriangleright x) \qquad \blacktriangleright x = (x\blacktriangleleft) \rhd x$
9. $[\lambda x_1 \ldots x_n.t](x_1, \ldots, x_n) = t$
10. $\mathrm{CRN}[g,h](x, \vec{y}) = x\,?\,(g(\vec{y}), t, t) \quad$ where $t = \mathrm{CRN}[g,h](x\lessdot, \vec{y}) \cdot h(x', x\lessdot, \vec{y})'$
11. $\mathrm{TRN}[g,h,\vec{l},\vec{r}](x,\vec{y}) = x\lessdot\,?\,(g(x,\vec{y}), t, t)$
    where $t = h\big(x, \vec{y}, \mathrm{TRN}[g,h,\vec{l},\vec{r}](x\blacktriangleleft, \vec{l}(x\blacktriangleleft, \vec{y})), \mathrm{TRN}[g,h,\vec{l},\vec{r}](\blacktriangleright x, \vec{r}(\blacktriangleright x, \vec{y}))\big)$

NOTE: By Claim 2.3, every NC^1 function is represented by some function symbol in T_1.

DEFINITION 3.3. The *rules of inference* of T_1 are as follows (where t, u, v stand for arbitrary terms, f stands for an arbitrary function symbol, x, y stand for arbitrary variables, and A stands for an arbitrary formula.)

0. Any standard, complete set of rules for the propositional calculus.
1. $\vdash t = t$
2. $t = u \vdash u = t$
3. $t = u, u = v \vdash t = v$
4. (a) $t_1 = u_1, \ldots, t_k = u_k \vdash f(t_1, \ldots, t_k) = f(u_1, \ldots, u_k)$
   (b) $t_1 = u_1, \ldots, t_k = u_k \vdash A[t_1/x_1, \ldots, t_k/x_k] \leftrightarrow A[u_1/x_1, \ldots, u_k/x_k]$
5. $A \vdash A[v/x]$
6. *Induction on Notation* (NIND) for A (where y does not occur in A).
   (a) $A[\varepsilon/x], A[y/x] \rightarrow A[0y/x], A[y/x] \rightarrow A[1y/x] \vdash A$
   (b) $A[\varepsilon/x], A[y/x] \rightarrow A[y0/x], A[y/x] \rightarrow A[y1/x] \vdash A$
7. *Tree Induction* (TIND) for A (where A has free variables $x, \vec{y}$).
   $A[\varepsilon/x], A[0/x], A[1/x],$
   $\big(A[w\blacktriangleleft/x, \vec{l}(w\blacktriangleleft, \vec{z})/\vec{y}] \wedge A[\blacktriangleright w/x, \vec{r}(\blacktriangleright w, \vec{z})/\vec{y}]\big) \rightarrow A[w/x, \vec{z}/\vec{y}]$
   $\vdash A$

4. Polysize $\mathcal{F}$-proofs for theorems of T_1

For every term t of T_1 with free variables $x_1, \ldots, x_k$, we will define a *length function* $\ell_t(m_1, \ldots, m_k)$ which gives the exact length of t as a function of the lengths of $x_1, \ldots, x_k$ (this function is well-defined because functions in L_1 are length-determined). Then, we will define a family of propositional *term formulas* $\langle t \rangle_1^{\vec{m}}, \ldots, \langle t \rangle_{\ell_t(\vec{m})}^{\vec{m}}$ which describe the bits of t in terms of the bits of $x_1, \ldots, x_k$ (where $\langle t \rangle_1^{\vec{m}}$ describes the leftmost bit of t), *i.e.*, given any truth-value assignment to the atoms representing the bits of $x_1, \ldots, x_n$, the truth value of $\langle t \rangle_i^{\vec{m}}$ represents the correct value for the corresponding bit of term t. Finally, for any formula A of T_1, we will define a family of *propositional translations* $[\![A]\!]^{\vec{m}}$, where $\vec{m}$ lists the lengths of all free variables in A, and show that there are short $\mathcal{F}$-proofs of $[\![A]\!]^{\vec{m}}$ whenever A is a theorem of T_1.

4.1. Length functions. Following the definition of the terms of T_1, the length functions are defined inductively as follows (where "sg" is the signum function).

$$\ell_x(m) = m$$

$$\ell_{f(t_1,\ldots,t_k)}(\vec{m}) = \ell_{f(x_1,\ldots,x_k)}(\ell_{t_1}(\vec{m}_1), \ldots, \ell_{t_k}(\vec{m}_k))^2$$

(where $x_1, \ldots, x_k$ occur in none of $t_1, \ldots, t_k$)

$$\ell_\varepsilon = 0 \qquad \ell_0 = 1 \qquad \ell_1 = 1$$

$$\ell_{0x}(m) = m \qquad \ell_{1x}(m) = m$$

$$\ell_{x'}(m) = \mathrm{sg}(m) \qquad \ell_{\lfloor x}(m) = \mathrm{sg}(m)$$

$$\ell_{x\leqslant}m = m \dot{-} 1 \qquad \ell_{\geqslant x}(m) = m \dot{-} 1$$

$$\ell_{x\blacktriangleleft}(m) = \lfloor m/2 \rfloor \qquad \ell_{\blacktriangleright x}(m) = \lceil m/2 \rceil$$

$$\ell_{x\lhd y}(m,n) = m \dot{-} n \qquad \ell_{x\rhd y}(m,n) = n \dot{-} m$$

$$\ell_{x\cdot y}(m,n) = m + n$$

$$\ell_{x?(y,z_0,z_1)}(m,n,o_0,o_1) = \text{if } m = 0 \text{ then } n \text{ else } \max\{o_0, o_1\}$$

$$\ell_{[\lambda\vec{x}.t](\vec{y})}(\vec{n}) = \ell_{t[\vec{y}/\vec{x}]}(\vec{n}')^3$$

$$\ell_{\mathrm{CRN}[g,h](x,\vec{y})}(m,\vec{n}) = \ell_{g(\vec{y})}(\vec{n}) + m$$

$$\ell_{\mathrm{TRN}[g,h,\vec{l},\vec{r}](x,\vec{y})}(m,\vec{n}) = \text{if } m \leq 1 \text{ then } \ell_{g(x,\vec{y})}(m,\vec{n}) \text{ else}$$

$$\ell_{h(x,\vec{y},z_0,z_1)}\Big(m, \vec{n}, \ell_{\mathrm{TRN}[g,h,\vec{l},\vec{r}](x\blacktriangleleft,\vec{l}(x\blacktriangleleft,\vec{y}))}(m,\vec{n}),$$
$$\ell_{\mathrm{TRN}[g,h,\vec{l},\vec{r}](\blacktriangleright x,\vec{r}(\blacktriangleright x,\vec{y}))}(m,\vec{n})\Big)$$

4.2. Term formulas. To every variable x of T_1 are associated propositional atoms $\langle x \rangle_1^m, \ldots, \langle x \rangle_m^m$. For other terms of T_1, the term formulas are defined inductively as follows. (When subscript i is used without specifying its range in the definition of $\langle t \rangle_i^{\vec{m}}$, it is implicitly assumed that $1 \leq i \leq \ell_t(\vec{m})$.)

$$\langle f(t_1, \ldots, t_k) \rangle_i^{\vec{m}} =$$

[2]Where $\vec{m}_i$ represents the lengths of the variables which occur in t_i.

[3]Where $\vec{n}'$ represents the lengths of the variables from $\vec{y}$ which actually occur in $t[\vec{y}/\vec{x}]$.

$$\langle f(x_1, \ldots, x_k) \rangle_i^{\ell_{t_1}(\vec{m}_1),\ldots,\ell_{t_k}(\vec{m}_k)} \Big[\langle t_j \rangle_{i_j}^{\vec{m}_j} / \langle x_j \rangle_{i_j}^{\ell_{t_j}(\vec{m}_j)} \Big]_{\substack{1\leq j\leq k \\ 1\leq i_j\leq \ell_{t_j}(\vec{m}_j)}}$$

(where $x_1, \ldots, x_k$ occur in none of $t_1, \ldots, t_k$)

$$\langle 0 \rangle_1 = \bot \qquad \langle 1 \rangle_1 = \top$$

$$\langle {}_0x \rangle_i^m = \bot \qquad \langle {}_1x \rangle_i^m = \top$$

$$\langle x' \rangle_1^m = \langle x \rangle_m^m \qquad \langle \lfloor x \rangle_1^m = \langle x \rangle_1^m$$

$$\langle x\leqslant \rangle_i^m = \langle x \rangle_i^m \qquad \langle \geqslant x \rangle_i^m = \langle x \rangle_{i+1}^m$$

$$\langle x\blacktriangleleft \rangle_i^m = \langle x \rangle_i^m \qquad \langle \blacktriangleright x \rangle_i^m = \langle x \rangle_{i+\lfloor m/2 \rfloor}^m$$

$$\langle x \lhd y \rangle_i^{m,n} = \langle x \rangle_i^m \qquad \langle x \rhd y \rangle_i^{m,n} = \langle y \rangle_{i+m}^n$$

$$\langle x \cdot y \rangle_i^{m,n} = \begin{cases} \langle x \rangle_i^m & \text{if } i \leq m \\ \langle y \rangle_{i-m}^n & \text{if } m < i \end{cases}$$

$$\langle x \,?\, (y, z_0, z_1) \rangle_i^{m,n,o_0,o_1} = \begin{cases} \langle y \rangle_i^n & \text{if } m = 0 \\ \big(\neg\langle x \rangle_m^m \wedge \langle {}_0(z_1 \lhd z_0) \cdot z_0 \rangle_i^{o_0,o_1}\big) & \\ \vee\big(\langle x \rangle_m^m \wedge \langle {}_0(z_0 \lhd z_1) \cdot z_1 \rangle_i^{o_0,o_1}\big) & \text{if } m > 0 \end{cases}$$

$$\langle [\lambda\vec{x}.t](\vec{y}) \rangle_i^{\vec{n}} = \langle t[\vec{y}/\vec{x}] \rangle_i^{\vec{n}'}$$

$$\langle \mathrm{CRN}[g,h](x,\vec{y}) \rangle_i^{m,\vec{n}} = \begin{cases} \langle g(\vec{y}) \rangle_i^{\vec{n}} & \text{if } m = 0 \\ \langle \mathrm{CRN}[g,h](x\leqslant,\vec{y}) \cdot h(x', x\leqslant, \vec{y})' \rangle_i^{m,\vec{n}} & \text{if } m > 0 \end{cases}$$

$$\big\langle \mathrm{TRN}[g,h,\vec{l},\vec{r}](x,\vec{y}) \big\rangle_i^{m,\vec{n}} = \begin{cases} \langle g(x,\vec{y}) \rangle_i^{m,\vec{n}} & \text{if } m \leq 1 \\ \Big\langle h\big(x, \vec{y}, \mathrm{TRN}[g,h,\vec{l},\vec{r}](x\blacktriangleleft, \vec{l}(x\blacktriangleleft,\vec{y})), & \\ \qquad \mathrm{TRN}[g,h,\vec{l},\vec{r}](\blacktriangleright x, \vec{r}(\blacktriangleright x,\vec{y}))\big)\Big\rangle_i^{m,\vec{n}} & \text{if } 1 < m \end{cases}$$

4.3. Propositional translations. The propositional translations of formulas of T_1 are defined inductively as follows.

$$[\![t = u]\!]^{\vec{m}} = \begin{cases} \bigwedge\limits_{1\leq i\leq \ell_t(\vec{m}_0)} \langle t \rangle_i^{\vec{m}_0} \leftrightarrow \langle u \rangle_i^{\vec{m}_1} & \text{if } \ell_t(\vec{m}_0) = \ell_u(\vec{m}_1), \\ \bot & \text{otherwise.} \end{cases}$$

$$[\![\neg A]\!]^{\vec{m}} = \neg[\![A]\!]^{\vec{m}}$$

$$[\![A \odot B]\!]^{\vec{m}} = [\![A]\!]^{\vec{m}_0} \odot [\![B]\!]^{\vec{m}_1}$$

(Where $\odot$ stands for any one of the binary propositional connectives, and $\vec{m}_0$ and $\vec{m}_1$ represent the lengths of the variables which occur in A and B (or t and u), respectively.)

4.4. The simulation result. Now, we can prove the following theorem.

THEOREM 4.1. *If A is provable in T_1, then for any $\vec{m}$, $[\![A]\!]^{\vec{m}}$ has polysize $\mathcal{F}$-proofs.*

PROOF. The proof is by induction on the number of inferences in the proof of A. If A is an axiom, then Subsection 4.4.1 shows that $[\![A]\!]^{\vec{m}}$ has short $\mathcal{F}$-proofs. If A is obtained by a derivation, then by the induction hypothesis, the propositional translations of the premises of the last inference all have short $\mathcal{F}$-proofs. Subsection 4.4.2 shows that in this case also, $[\![A]\!]^{\vec{m}}$ has short $\mathcal{F}$-proofs. □

4.4.1. *Axioms.* For most axioms of the form $t = u$, just writing down the definitions of $\langle t\rangle_i$ and $\langle u\rangle_i$ is enough to see that the axiom is a theorem with short proofs. We give a more detailed argument only for a few axioms.

0. The axioms for the propositional calculus can obviously be simulated by any $\mathcal{F}$-system.
1. $\langle 0\rangle_1 = \bot$ and $\langle 1\rangle_1 = \top$.
2. (a) $\langle \varepsilon \cdot x\rangle_i^m = \langle x\rangle_i^m = \langle x \cdot \varepsilon\rangle_i^m$ for all $1 \le i \le m$.
   (b) For all $1 \le i \le m + n + o$,

$$\langle (x \cdot y) \cdot z\rangle_i^{m,n,o} = \begin{cases} \langle x \cdot y\rangle_i^{m,n} = \begin{cases} \langle x\rangle_i^m & \text{if } i \le m \\ \langle y\rangle_{i-m}^n & \text{if } m < i \end{cases} & \text{if } i \le m+n \\ \langle z\rangle_{i-(m+n)}^o & \text{if } m+n < i \end{cases}$$

$$\langle x \cdot (y \cdot z)\rangle_i^{m,n,o} = \begin{cases} \langle x\rangle_i^m & \text{if } i \le m \\ \langle y \cdot z\rangle_{i-m}^{n,o} = \begin{cases} \langle y\rangle_{i-m}^n & \text{if } i-m \le n \\ \langle z\rangle_{i-m-n}^o & \text{if } n < i-m \end{cases} & \text{if } m < i \end{cases}$$

   (c)

$$[\![z : x = z \cdot y]\!]^{m,n,o} = \begin{cases} \bigwedge\!\!\!\bigwedge_{1\le i\le o} \langle z\rangle_i^o \leftrightarrow \langle z\rangle_i^o \wedge \bigwedge\!\!\!\bigwedge_{o<i\le m+o} \langle x\rangle_{i-o}^m \leftrightarrow \langle y\rangle_{i-o}^n & \text{if } m = n \\ \bot & \text{otherwise} \end{cases}$$

$$[\![x = y]\!]^{m,n} = \begin{cases} \bigwedge\!\!\!\bigwedge_{1\le i\le m} \langle x\rangle_i^m \leftrightarrow \langle y\rangle_i^n & \text{if } m = n \\ \bot & \text{otherwise} \end{cases}$$

$$[\![x \cdot z = y \cdot z]\!]^{m,n,o} = \begin{cases} \bigwedge\!\!\!\bigwedge_{1\le i\le m} \langle x\rangle_i^m \leftrightarrow \langle y\rangle_i^n \wedge \bigwedge\!\!\!\bigwedge_{m<i\le o+m} \langle z\rangle_{i-m}^o \leftrightarrow \langle z\rangle_{i-m}^o & \text{if } m = n \\ \bot & \text{otherwise} \end{cases}$$

   (d) $[\![x \cdot y = \varepsilon]\!]^{m,n}$ holds iff $m + n = 0$, and $[\![x = \varepsilon \wedge y = \varepsilon]\!]^{m,n}$ holds iff $m = 0$ and $n = 0$.
   (e) $[\![x \cdot y = 0]\!]^{m,n}$ holds iff $m+n = 1$ and $\langle x\rangle_1^1 = \bot$ or $\langle y\rangle_1^1 = \bot$. Similarly for $x \cdot y = 1$.
3. (a) $[\![0\varepsilon = \varepsilon]\!] = \top$ since $\ell_{0\varepsilon} = \ell_\varepsilon = 0$, and for $1 \le i \le m+1$, $\langle 0(x0)\rangle_i^m = \langle 0(x1)\rangle_i^m = \bot$ and $\langle 0x \cdot 0\rangle_i^m = \begin{cases} \langle 0x\rangle_i^m & \text{if } i \le m, \\ \langle 0\rangle_{i-m} & \text{if } m < i. \end{cases}$
   (b) Similarly to (a).
4. (a) $[\![\backslash\varepsilon = \varepsilon]\!] = \top$ since $\ell_{\backslash\varepsilon} = \ell_\varepsilon = 0$; also, $\langle \backslash(0x)\rangle_1^m = \langle 0x\rangle_1^m = \langle 0\rangle_1$ by definition, and the same reasoning applies to $\langle \backslash(1x)\rangle_1^m$.
   (b) $[\![\varepsilon' = \varepsilon]\!] = \top$ since $\ell_{\varepsilon'} = \ell_\varepsilon = 0$; also, $\langle (x0)'\rangle_1^m = \langle x0\rangle_{m+1}^m = \langle 0\rangle_1$ by definition, and the same reasoning applies to $\langle (x1)'\rangle_1^m$.
5. (a) $[\![\triangleright\varepsilon = \varepsilon]\!] = \top$ since $\ell_{\triangleright\varepsilon} = \ell_\varepsilon = 0$; also, for $1 \le i \le m$, $\langle \triangleright(0x)\rangle_i^m = \langle 0x\rangle_{i+1}^m = \langle x\rangle_i^m$ by definition; the same reasoning applies to $\langle \triangleright(1x)\rangle_i^m$.
   (b) $[\![\varepsilon \triangleleft = \varepsilon]\!] = \top$ since $\ell_{\varepsilon\triangleleft} = \ell_\varepsilon = 0$; also, for $1 \le i \le m$, $\langle (x0)\triangleleft\rangle_i^m = \langle x0\rangle_i^m = \langle x\rangle_i^m$ by definition; the same reasoning applies to $\langle (x1)\triangleleft\rangle_i^m$.
6. (a) $\langle \varepsilon \rhd x\rangle_i^m = \langle x\rangle_{i+0}^m$ for $1 \le i \le m$ by definition; also, for $1 \le i \le m-(n+1)$, $\langle (0y) \rhd x\rangle_i^{m,n} = \langle x\rangle_{i+n+1}^m = \langle y \rhd x\rangle_{i+1}^{m,n} = \langle \triangleright(y \rhd x)\rangle_i^{m,n}$, and the same reasoning applies to $\langle (1y) \rhd x\rangle_i^{m,n}$.

   (b) $\langle x \lhd \varepsilon\rangle_i^m = \langle x\rangle_i^m$ for $1 \le i \le m$ by definition; also, for $1 \le i \le m-(n+1)$, $\langle x \lhd (y0)\rangle_i^{m,n} = \langle x\rangle_i^m = \langle x \lhd y\rangle_i^{m,n} = \langle (x \lhd y)\triangleleft\rangle_i^{m,n}$, and the same reasoning applies to $\langle x \lhd (y1)\rangle_i^{m,n}$.
7. $\langle \varepsilon\ ?\ (x,y,z)\rangle_i^{m,n,o} = \langle x\rangle_i$ for $1 \le i \le m$, by definition; also, for $1 \le i \le \max\{n,o\}$,

$$\begin{aligned} \langle (w0)\ ?\ (x,y,z)\rangle_i^{k,m,n,o} &= \big(\neg\langle w0\rangle_{k+1}^k \wedge \langle 0(z \lhd y)\cdot y\rangle_i^{n,o}\big) \\ &\quad \vee \big(\langle w0\rangle_{k+1}^k \wedge \langle 0(y \lhd z)\cdot z\rangle_i^{n,o}\big) \\ &= \big(\neg\bot \wedge \langle 0(z \lhd y)\cdot y\rangle_i^{n,o}\big) \vee \big(\bot \wedge \langle 0(y \lhd z)\cdot z\rangle_i^{n,o}\big) \\ &\leftrightarrow \langle 0(z \lhd y)\cdot y\rangle_i^{n,o} \quad \text{and} \\ \langle (w1)\ ?\ (x,y,z)\rangle_i^{k,m,n,o} &= \big(\neg\langle w1\rangle_{k+1}^k \wedge \langle 0(z \lhd y)\cdot y\rangle_i^{n,o}\big) \\ &\quad \vee \big(\langle w1\rangle_{k+1}^k \wedge \langle 0(y \lhd z)\cdot z\rangle_i^{n,o}\big) \\ &= \big(\neg\top \wedge \langle 0(z \lhd y)\cdot y\rangle_i^{n,o}\big) \vee \big(\top \wedge \langle 0(y \lhd z)\cdot z\rangle_i^{n,o}\big) \\ &\leftrightarrow \langle 0(y \lhd z)\cdot z\rangle_i^{n,o} \end{aligned}$$

8. (a) For all $1 \le i \le m$,

$$\langle (x\blacktriangleleft)\cdot(\blacktriangleright x)\rangle_i^m = \begin{cases} \langle x\blacktriangleleft\rangle_i^m = \langle x\rangle_i^m & \text{if } i \le \lfloor m/2\rfloor \\ \langle \blacktriangleright x\rangle_{i-\lfloor m/2\rfloor}^m = \langle x\rangle_{i-\lfloor m/2\rfloor+\lfloor m/2\rfloor}^m & \text{if } \lfloor m/2\rfloor < i \end{cases}$$

   (b) $x\blacktriangleleft \lhd \blacktriangleright x = \varepsilon$ since $\ell_{x\blacktriangleleft}(m) \dot{-} \ell_{\blacktriangleright x}(m) = \lfloor m/2\rfloor \dot{-} \lceil m/2\rceil = 0$;
   $\triangleright(x\blacktriangleleft \rhd \blacktriangleright x) = \varepsilon$ since $\ell_{\blacktriangleright x}(m) \dot{-} \ell_{x\blacktriangleleft}(m) = \lceil m/2\rceil \dot{-} \lfloor m/2\rfloor \le 1$.
   (c) For $1 \le i \le m - \lceil m/2\rceil = \lfloor m/2\rfloor$, $\langle x \lhd \blacktriangleright x\rangle_i^m = \langle x\rangle_i^m = \langle x\blacktriangleleft\rangle_i^m$;
   for $1 \le i \le m - \lfloor m/2\rfloor = \lceil m/2\rceil$, $\langle x\blacktriangleleft \rhd x\rangle_i^m = \langle x\rangle_{i+\lfloor m/2\rfloor}^m = \langle \blacktriangleright x\rangle_i^m$.
9. $\langle [\lambda\vec{x}.t](\vec{x})\rangle_i^{\vec{m}} = \langle t[\vec{x}/\vec{x}]\rangle_i^{\vec{m}'} = \langle t\rangle_i^{\vec{m}'}$ for all $1 \le i \le \ell_t(\vec{m}')$.

The last two axioms are easy to prove if we note the following two facts.

(a) *For any term t, $t \lhd t = \varepsilon$ and therefore $\langle 0(t \lhd t)\cdot t\rangle_i^{\vec{m}} = \langle t\rangle_i^{\vec{m}}$ for $1 \le i \le \ell_t(\vec{m})$.*
(b) *Within any $\mathcal{F}$-system, the identity $(\neg p \wedge q) \vee (p \wedge q) \leftrightarrow q$ has short proofs.*

10. For all $1 \le i \le \ell_{\mathrm{CRN}[g,h](x,\vec{y})}(m,\vec{n})$,

$$\langle x\ ?\ (g(\vec{y}),t,t)\rangle_i^{m,\vec{n}} = \begin{cases} \langle g(\vec{y})\rangle_i^{\vec{n}} & \text{if } m = 0 \\ \big(\neg\langle x\rangle_m^m \wedge \langle 0(t \lhd t)\cdot t\rangle_i^{m,\vec{n}}\big) & \\ \vee\big(\langle x\rangle_m^m \wedge \langle 0(t \lhd t)\cdot t\rangle_i^{m,\vec{n}}\big) & \text{if } m > 0 \end{cases}$$

where $t = \mathrm{CRN}[g,h](x\triangleleft,\vec{y}) \cdot h(x', x\triangleleft, \vec{y})'$.

11. For all $1 \le i \le \ell_{\mathrm{TRN}[g,h,\vec{l},\vec{r}](x,\vec{y})}(m,\vec{n})$,

$$\langle x\triangleleft\ ?\ (g(x,\vec{y}),t,t)\rangle_i^{m,\vec{n}} = \begin{cases} \langle g(x,\vec{y})\rangle_i^{\vec{n}} & \text{if } m \dot{-} 1 = 0 \\ \big(\neg\langle x\triangleleft\rangle_{m\dot{-}1}^m \wedge \langle 0(t \lhd t)\cdot t\rangle_i^{m.\vec{n}}\big) & \\ \vee\big(\langle x\triangleleft\rangle_{m\dot{-}1}^m \wedge \langle 0(t \lhd t)\cdot t\rangle_i^{m,\vec{n}}\big) & \text{if } m \dot{-} 1 > 0 \end{cases}$$

where $t = h\big(x,\vec{y},\mathrm{TRN}[g,h,\vec{l},\vec{r}](x\blacktriangleleft,\vec{l}(x\blacktriangleleft,\vec{y})),\mathrm{TRN}[g,h,\vec{l},\vec{r}](\blacktriangleright x,\vec{r}(\blacktriangleright x,\vec{y}))\big)$.

4.4.2. *Rules of inference.* For all the rules in Definition 3.3, if one of the premises of the form $[\![t = u]\!]^{\vec{m}}$ degenerates to $\bot$ because $\ell_t(\vec{m}_0) \neq \ell_u(\vec{m}_1)$, then the rule becomes trivial. We therefore assume that none of the propositional translations of atomic formulas of T_1 are degenerate cases. Also, when we use the notation "$\bigwedge\!\!\!\bigwedge$" with no subscript, we implicitly assume that the conjunction is over all relevant values of the index of the term formulas involved.

0. Any standard, complete set of rules for the propositional calculus can be p-simulated within any $\mathcal{F}$-system.
1. By reflexivity, there are short $\mathcal{F}$-proofs of $\mathbb{A}\langle t\rangle_i^{\vec{m}} \leftrightarrow \langle t\rangle_i^{\vec{m}}$ for any term t.
2. We have short $\mathcal{F}$-proofs of $\mathbb{A}\langle t\rangle_i^{\vec{m}} \leftrightarrow \langle u\rangle_i^{\vec{n}}$. By commutativity, it is easy to get short $\mathcal{F}$-proofs of $\mathbb{A}\langle u\rangle_i^{\vec{n}} \leftrightarrow \langle t\rangle_i^{\vec{m}}$.
3. We have short $\mathcal{F}$-proofs of $\mathbb{A}\langle t\rangle_i^{\vec{m}} \leftrightarrow \langle u\rangle_i^{\vec{n}}$ and $\mathbb{A}\langle u\rangle_i^{\vec{n}} \leftrightarrow \langle v\rangle_i^{\vec{o}}$. By transitivity, it is easy to get short $\mathcal{F}$-proofs of $\mathbb{A}\langle t\rangle_i^{\vec{m}} \leftrightarrow \langle v\rangle_i^{\vec{o}}$.
4. (a) We have short $\mathcal{F}$-proofs of $\mathbb{A}\langle t_j\rangle_{i_j}^{\vec{m}_j} \leftrightarrow \langle u_j\rangle_{i_j}^{\vec{n}_j}$ for all $1 \le j \le k$. We also have short $\mathcal{F}$-proofs of $\mathbb{A}\langle f(x_1,\ldots,x_k)\rangle_i^{\ell_{t_1}(\vec{m}_1),\ldots,\ell_{t_k}(\vec{m}_k)} \leftrightarrow \langle f(x_1,\ldots,x_k)\rangle_i^{\ell_{u_1}(\vec{n}_1),\ldots,\ell_{u_k}(\vec{n}_k)}$. By substitution of equivalents, it is easy to get short $\mathcal{F}$-proofs of

$$\mathbb{A}\left(\begin{array}{l}\langle f(x_1,\ldots,x_k)\rangle_i^{\ell_{t_1}(\vec{m}_1),\ldots,\ell_{t_k}(\vec{m}_k)}\left[\langle t_j\rangle_{i_j}^{\vec{m}_j}/\langle x_j\rangle_{i_j}^{\ell_{t_j}(\vec{m}_j)}\right] \\ \leftrightarrow \langle f(x_1,\ldots,x_k)\rangle_i^{\ell_{u_1}(\vec{n}_1),\ldots,\ell_{u_k}(\vec{n}_k)}\left[\langle u_j\rangle_{i_j}^{\vec{n}_j}/\langle x_j\rangle_{i_j}^{\ell_{u_j}(\vec{n}_j)}\right]\end{array}\right)$$

   (b) The same reasoning as for part (a) applies.
5. We have short $\mathcal{F}$-proofs of $[\![A]\!]^{\vec{m},m}$. Substituting $\langle v\rangle_i^{\vec{o}}$ for $\langle x\rangle_i^m$ throughout these proofs yield short $\mathcal{F}$-proofs of $[\![A[v/x]]\!]^{\vec{m},\vec{o}}$.
6. (a) First, a few observations. Let $F = (x = \varepsilon \vee x = 0\cdot{>}x \vee x = 1\cdot{>}x)$. Then,

$$[\![x=\varepsilon]\!]^m = \begin{cases}\top & \text{if } m=0\\ \bot & \text{if } m>0\end{cases}$$

$$\begin{aligned}[\![x = 0\cdot{>}x]\!]^m &= \mathbb{A}\langle x\rangle_i^m \leftrightarrow \langle 0\cdot{>}x\rangle_i^m\\ &= \langle x\rangle_1^m \leftrightarrow \langle 0\rangle_1 \wedge \mathbb{A}_{1<i}\langle x\rangle_i^m \leftrightarrow \langle {>}x\rangle_{i-1}^m\\ &= \langle x\rangle_1^m \leftrightarrow \bot \wedge \mathbb{A}_{1<i}\langle x\rangle_i^m \leftrightarrow \langle x\rangle_i^m\end{aligned}$$

$$[\![x = 1\cdot{>}x]\!]^m = \langle x\rangle_1^m \leftrightarrow \top \wedge \mathbb{A}_{1<i}\langle x\rangle_i^m \leftrightarrow \langle x\rangle_i^m$$

   so that

$$\begin{aligned}[\![F]\!]^m &= \begin{cases}\top\vee\top\vee\top & \text{if } m=0\\ \bot\vee\left(\langle x\rangle_1^m \leftrightarrow \bot\wedge\mathbb{A}_{1<i}\langle x\rangle_i^m\leftrightarrow\langle x\rangle_i^m\right) & \\ \vee\left(\langle x\rangle_1^m\leftrightarrow\top\wedge\mathbb{A}_{1<i}\langle x\rangle_i^m\leftrightarrow\langle x\rangle_i^m\right) & \text{if } m>0\end{cases}\\ &= \begin{cases}\top\vee\top\vee\top & \text{if } m=0\\ \bot\vee\left(\left(\langle x\rangle_1^m\leftrightarrow\bot\vee\langle x\rangle_1^m\leftrightarrow\top\right)\wedge\mathbb{A}_{1<i}\langle x\rangle_i^m\leftrightarrow\langle x\rangle_i^m\right) & \text{if } m>0\end{cases}\\ &\leftrightarrow \begin{cases}\top\vee\top\vee\top & \text{if } m=0\\ \bot\vee\left(\left(\neg\langle x\rangle_1^m\vee\langle x\rangle_1^m\right)\wedge\top\right) & \text{if } m>0\end{cases}\end{aligned}$$

   Therefore, $[\![F]\!]^m$ has short $\mathcal{F}$-proofs (it is an easy tautology). Now, we have short $\mathcal{F}$-proofs of

$$\begin{aligned}A_\varepsilon &= [\![A[\varepsilon/x]]\!]^{\vec{m}},\\ A_0 &= [\![A[y/x]]\!]^{\vec{m},n} \to [\![A[0y/x]]\!]^{\vec{m},n},\end{aligned}$$

$$\text{and } A_1 = [\![A[y/x]]\!]^{\vec{m},n} \to [\![A[1y/x]]\!]^{\vec{m},n}.$$

If $m=0$, then by rule 4(b), there are short proofs of

$$[\![x=\varepsilon]\!]^m \to \left([\![A]\!]^{\vec{m},m} \leftrightarrow [\![A[\varepsilon/x]]\!]^{\vec{m}}\right),$$

which shows that there are short proofs of

$$[\![x=\varepsilon]\!]^m \to \left([\![A[\varepsilon/x]]\!]^{\vec{m}} \to [\![A]\!]^{\vec{m},n}\right). \tag{4.1}$$

If $m>0$, then substituting ${>}x$ for y in A_0 gives short proofs of

$$[\![A[{>}x/x]]\!]^{\vec{m},n} \to [\![A[0\cdot{>}x/x]]\!]^{\vec{m},n};$$

moreover, by rule 4(b), there are short proofs of

$$[\![x=0\cdot{>}x]\!]^m \to \left([\![A]\!]^{\vec{m},m} \leftrightarrow [\![A[0\cdot{>}x/x]]\!]^{\vec{m},m}\right),$$

and therefore, by transitivity, of

$$[\![x=0\cdot{>}x]\!]^m \to \left([\![A[{>}x/x]]\!]^{\vec{m},m} \to [\![A]\!]^{\vec{m},m}\right).$$

A similar argument shows that there are short proofs of

$$[\![x=1\cdot{>}x]\!]^m \to \left([\![A[{>}x/x]]\!]^{\vec{m},m} \to [\![A]\!]^{\vec{m},m}\right),$$

which, together with (4.1), implies that there are short proofs of

$$[\![F]\!]^m \to \left([\![A[{>}x/x]]\!]^{\vec{m},m} \to [\![A]\!]^{\vec{m},m}\right).$$

Applying modus ponens to this and $[\![F]\!]^m$ gives short proofs of

$$[\![A[{>}x/x]]\!]^{\vec{m},m} \to [\![A]\!]^{\vec{m},m}. \tag{4.2}$$

Repeated substitutions of ${>}x$ for x in the proof of (4.2) give short proofs of

$$\begin{aligned}&[\![A[{>}^2x/x]]\!]^{\vec{m},m} \to [\![A[{>}x/x]]\!]^{\vec{m},m}\\ &\qquad\vdots\\ &[\![A[{>}^mx/x]]\!]^{\vec{m},m} \to [\![A[{>}^{m-1}x/x]]\!]^{\vec{m},m}\end{aligned}$$

where ${>}^k = \overbrace{{>}\cdots{>}}^{k}$. Since ${>}^mx = \varepsilon$, using rule 4(b) and modus ponens gives short proofs of

$$[\![A[\varepsilon/x]]\!]^{\vec{m}} \to [\![A[{>}^mx/x]]\!]^{\vec{m},m},$$

and using transitivity $m+1$ times now gives short proofs of

$$[\![A[\varepsilon/x]]\!]^{\vec{m}} \to [\![A]\!]^{\vec{m},m}.$$

A final application of modus ponens with A_ε gives the short proofs of $[\![A]\!]^{\vec{m},m}$ we wanted: the size of this proof is $\mathcal{O}(m\cdot p(\vec{m}))$ where $p(\vec{m})$ was the size of the proofs of A_ε, A_0, A_1.

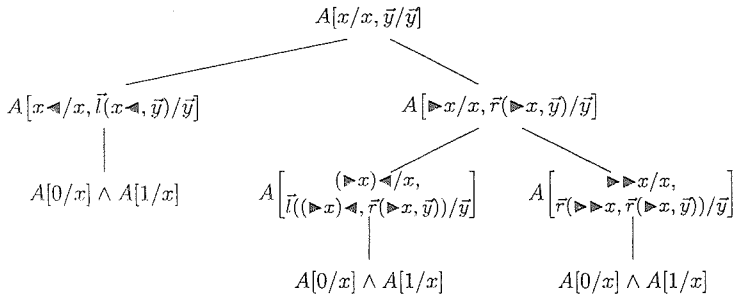
   (b) The same reasoning as for part (a) applies.
7. We have short proofs of $[\![A[\varepsilon/x]]\!]^{\vec{n}}$, $[\![A[0/x]]\!]^{\vec{n}}$, $[\![A[1/x]]\!]^{\vec{n}}$, and

$$\begin{aligned}A' = &\left([\![A[w{\blacktriangleleft}/x, \vec{l}(w{\blacktriangleleft},\vec{z})/\vec{y}]]\!]^{m,\vec{n}} \wedge [\![A[{\blacktriangleright}w/x, \vec{r}({\blacktriangleright}w,\vec{z})/\vec{y}]]\!]^{m,\vec{n}}\right)\\ &\to [\![A[w/x,\vec{z}/\vec{y}]]\!]^{m,\vec{n}}.\end{aligned}$$

If $m = 0$, using modus ponens twice on formula (4.1) gives a short proof of $[\![A]\!]^{m,\vec{n}}$. If $m > 0$, then by substituting x for w and $\vec{y}$ for $\vec{z}$ in the proof of A', we get short proofs of

$$A' = \left([\![A[x\blacktriangleleft/x, \vec{l}(x\blacktriangleleft,\vec{y})/\vec{y}]]\!]^{m,\vec{n}} \wedge [\![A[\blacktriangleright x/x, \vec{r}(\blacktriangleright x,\vec{y})/\vec{y}]]\!]^{m,\vec{n}}\right) \rightarrow [\![A[x/x,\vec{y}/\vec{y}]]\!]^{m,\vec{n}}.$$

and by repeatedly substituting first $x\blacktriangleleft, \vec{l}(x\blacktriangleleft,\vec{y})$ and then $\blacktriangleright x, \vec{r}(\blacktriangleright x,\vec{y})$ for $w,\vec{z}$ in the proof of A', we get a binary tree of short proofs of formulas of the form of A', where the formula at the root is $[\![A[x/x,\vec{y}/\vec{y}]]\!]^{m,\vec{n}}$, the formula at each node is implied by the conjunction of the formulas at its children nodes, and at the leaves, the terms being substituted for w can all be proved to be equal to 0 or 1 (single bits of x). For example, if $m = 3$, the tree would look like this (where we've indicated only the consequent of the formula being proved at each node, so that a node B with children C and D represents a proof of the formula $(C \wedge D) \rightarrow B$ and a node B with one child C represents a proof of the formula $C \rightarrow B$):



Therefore, the proofs of $[\![A[0/x]]\!]^{\vec{n}}$ and $[\![A[1/x]]\!]^{\vec{n}}$ can be used with rule 5 (substituting the right terms for $\vec{y}$) and modus ponens to prove the formulas at the first level, and going up level by level using modus ponens, we obtain a proof of the consequent of the formula at the root of the tree, *i.e.*, $[\![A]\!]^{m,\vec{n}}$. Moreover, the size of this proof is $\mathcal{O}(m \cdot p(\vec{n}))$ where $p(\vec{n})$ was the size of the proofs of the premises, since the tree has depth no more than $\lceil \log_2 m \rceil$ and thus size no more than $2m$.

5. Conclusion and future work

As the large section on the algebra L_1 has shown, this new recursive characterization of FNC^1 seems elegant and natural. Also, the theory T_1 based on L_1 has the desirable property that its appropriately translated theorems have short $\mathcal{F}$-proofs, and the proof of that fact is direct.

Unfortunately, I have not yet proved that T_1 can prove the soundness of $\mathcal{F}$. This essentially involves formalizing an algorithm for the Boolean Sentence Value Problem within T_1 and proving properties of that algorithm in T_1, which is not a trivial task. To this effect, Steve Cook suggested studying stronger forms of TRN, a suggestion which I have incorporated into this version of L_1 and T_1.

Even without knowing the exact formulation of T_1, it is interesting to consider what can be proved about the theory only from its desired properties. For example,

based on a similar result of Sam Buss [9], which uses Krajíček, Pudlák & Takeuti's Herbrand-type witnessing theorem [17], Steve Cook has argued that if QPV is conservative over QT_1, then $P = ALOGTIME$, where QPV and QT_1 are appropriately defined quantified theories. Hopefully, a similar theorem can be shown for the normal versions PV and T_1, and some consequences of conservativity results between PV and T_1 for the relationship between $\mathcal{F}$ and $e\mathcal{F}$ would be interesting.

Also, it would be interesting to define and study a first-order theory based on T_1, compare it to Takeuti and Clote's TNC^0, and study conservativity results between such a theory and S^1_2.

Finally, an obvious generalization of Definition 2.1 suggests itself: For $i \geq 1$, let $\mathrm{TRN}[L_i]$ be the set of all functions defined by TRN from functions in L_i, and let L_{i+1} be the closure of $L_i \cup \mathrm{TRN}[L_i]$ under COMP and CRN. A study of these classes (or of a similar extension for the theory T_1) might be interesting. (One fact about L_i which is relatively easy to prove is that it is a subset of the class of functions computable by uniform circuit families of $\mathcal{O}(\log^i)$ depth. It might be interesting to try to prove better results.)

Acknowledgements

I am deeply indebted to my advisor, Steve Cook, for suggesting this line of research and for many helpful discussions, suggestions, and comments. I have also benefited from correspondence with Peter Clote and Sam Buss, and from many discussions with Arnold Rosenbloom. Finally, my thanks go to the anonymous referee for pointing out many typos and helping me to make the presentation clearer.

References

1. Bill Allen, *Arithmetizing uniform NC*, Annals of Pure and Applied Logic **53** (1991), 1–50.
2. David A. Barrington, *Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1.*, Journal of Computer and System Science **38** (1989), 150–164.
3. Stephen Bellantoni and Stephen Cook, *A new recursion-theoretic characterization of the polytime functions*, Computational Complexity **2** (1992), 97–110.
4. ———, *The equivalence of certain first and second order feasible logics*, Manuscript, 1993.
5. Stephen Bloch, *Functional characterizations of uniform log-depth and polylog-depth circuit families*, Proceedings of IEEE 7th Annual Structure in Complexity Theory Conference (Boston, Massachusetts), June 1992, pp. 193–206.
6. ———, *Function-algebraic characterizations of log and polylog parallel time*, Computational Complexity **4** (1994), 175–205.
7. Samuel R. Buss, *The Boolean formula value problem is in ALOGTIME*, Journal of the Association for Computing Machinery (1987), 123–131.
8. ———, *Polynomial size proofs of the propositional pigeonhole principle*, Journal of Symbolic Logic **52** (1987), no. 4, 916–927.
9. ———, *Relating the bounded arithmetic and polynomial time hierarchies*, Manuscript, November 1994.
10. Peter Clote, *Sequential, machine-independent characterizations of the parallel complexity classes ALOGTIME, AC^k, NC^k, and NC*, Feasible Mathematics (S. R. Buss and P. Scott, eds.), Birkhäuser, 1989, pp. 49–69.
11. ———, *ALOGTIME and a conjecture of S. A. Cook*, Annals of Mathematics and Artificial Intelligence **6** (1992), 57–106, Extended abstract in: Proceedings of IEEE Symposium on Logic in Computer Science, Philadelphia, June 1990.
12. ———, *On polynomial size Frege proofs of certain combinatorial principles*, Arithmetic, Proof Theory and Computational Complexity (P. Clote and J. Krajíček, eds.), Oxford University Press, 1993, pp. 162–184.
13. Peter Clote and Gaisi Takeuti, *First order bounded arithmetic and small Boolean circuit complexity classes*, Manuscript (to appear in Feasible Mathematics II), December 1994.

14. Alan Cobham, *The intrinsic computational difficulty of functions*, Proceedings of the 1964 International Congress for Logic Methodology and the Philosophy of Science (Amsterdam) (Y. Bar-Hillel, ed.), North Holland, 1964, pp. 24–30.
15. Stephen A. Cook, *Feasible constructive proofs and the propositional calculus*, Proceedings of the Seventh Annual ACM Symposium on the Theory of Computing, May 1975, pp. 83–97.
16. Stephen A. Cook and Alasdair Urquhart, *Functional interpretations of feasibly constructive arithmetic*, Annals of Pure and Applied Logic **63** (1993), no. 2, 103–200.
17. Jan Krajíček, Pavel Pudlák, and Gaisi Takeuti, *Bounded arithmetic and the polynomial hierarchy*, Annals of Pure and Applied Logic **52** (1991), 143–153.
18. Gaisi Takeuti, *Frege proof system and TNC^0*, Manuscript, 1994.

Department of Computer Science, 10 King's College Road, Toronto ON M5S 3G4, CANADA
E-mail address: fpitt@cs.toronto.edu

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume **39**, 1998

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 $\Box_1^p$. 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 sized Frege proofs. Given these relationship between proof systems and bounded

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

arithmetic, determining whether or not Frege can polynomially simulate extended Frege could be important in solving whether $NC^1 = P$.

Besides S_2^i and T_2^i another important sequence of bounded arithmetic theories are the theories R_2^i developed in [1], [8], [18], and [2]. These theories are of interest to computer scientists since the Σ_1^b-definable functions of R_2^1 correspond to functions in the parallel computation class NC [1]. The theory R_2^1 lies between the theories TNC^0 and S_2^1 so a proof system for R_2^1 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_2^i has at its disposal and also show some closure properties for the Σ_i^b-definable multifunctions and Δ_i^b-predicates of R_2^i. Section 4 describes how to formalize quantified propositional logic within R_2^i. We describe in this section a $B(\Sigma_i^b)$-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_2^i. Proofs in the systems L_i^* are tree-like, $(\log)^2$-height quantified propositional sequent calculus proofs where formulas are restricted to the class $\Sigma_i^q \cup \Pi_i^q$ and the number of $\Sigma_i^q \cup \Pi_i^q$-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_2^1. Section 7 contains a witnessing argument used to show R_2^i can prove the validity of any Σ_i^q-formula derived by an L_i^*-proof. This statement is formally called i-$RFN(L_i^*)$. In Section 8, 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. This proof is similar to the simulation of T_2^i 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_i^b-$LLIND$ 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_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.

2. Preliminaries

The theories R_2^i 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_2^1. The language, L_2, of R_2^i 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_i^b and Π_i^b. The set $\Sigma_0^b = \Pi_0^b$ is the set of formulas all of whose quantifiers are sharply bounded. The set Σ_i^b is the smallest set containing Π_{i-1}^b and closed under conjunction, disjunction, sharply bounded quantification, and bounded existential quantification. The set Π_i^b is the smallest class containing Σ_{i-1}^b and closed under conjunction, disjunction, sharply bounded quantification, and bounded universal quantification.

We define several kinds of induction axioms:

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

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

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

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

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

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_2^i is axiomatized as $BASIC$+Σ_i^b-IND, the theory S_2^i is axiomatized as $BASIC$+Σ_i^b-$PIND$ axioms, and finally the theory R_2^i is axiomatized as $BASIC$+Σ_i^b-$PLIND$ axioms. The theories S_2^i and T_2^i 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_2^i in [18] also had a bit extensionality axiom shown to be unnecessary in [13].

Recall that a theory T can Σ_i^b-define a function $f(x)$, if there is a Σ_i^b-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_i^b with respect to a theory T if it is provably equivalent in T to both a Σ_i^b and a Π_i^b-formula. It is a theorem of [5] that once we can Σ_1^b-define a function f in a theory we can add the function symbol to the theory without changing the Σ_i^b or Π_i^b-consequences of the theory $i \geq 1$. A similar result holds for adding Δ_1^b-predicate symbols [5].

In concluding our preliminary discussion of R_2^i we would like to mention that the theory R_2^i can also prove Σ_i^b-LLIND axioms, Δ_i^b-$PIND$ axioms, and Δ_i^b-$LIND$ axioms [1], [18]. So R_2^i contains S_2^{i-1} and one can also show that S_2^{i-1} contains T_2^{i-2} [5].

Now that we have finished introducing R_2^i 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_i^q and Π_i^q of propositional formulas in analogous way to the first-order case. We define Δ_{i+1}^q to mean the class of Boolean combinations of Σ_i^q-formulas.

3. Functions definable in R_2^i

In this section, we describe some of the Δ_1^b-predicates and Σ_1^b-functions available to code sequences in R_2^1. These predicates and functions will enable us to formalize the syntax of quantified propositional formulas in R_2^1 and to give a truth predicate for these formulas. We also discuss what kinds of multifunctions are available in the theories R_2^i.

Methods in R_2^1 for coding a sequence of natural numbers $\langle a_1, \ldots, a_n \rangle$ as a single natural number have been described by both Allen [1] and Clote-Takeuti [8]. In particular, R_2^1 can prove the following functions and predicates which are useful in manipulating sequences to be either Σ_1^b-definable or Δ_1^b:

- $Seq(w)$ 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.
- $len(w)$ which returns the length of the sequence w.
- $Bit(i, a)$ which returns the ith bit of a number a.
- $Subseq(i, j, w)$ which returns the subsequence of the ith through the jth entry of w.
- $Front(w) := Subseq(1, len(w) - 1, w)$ which returns all but the last element of a sequence
- $Tail(w) := Subseq(2, len(w), w)$ 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 $1 \# y$
- $cond(x, y, z)$ which outputs y if x is 0 and outputs z otherwise. It allows us give definitions by cases.
- $\max(x, y)$ which equals $cond(x \dot{-} y, y, x)$
- $\min(x, y)$ which equals $cond(x \dot{-} y, x, y)$
- $\lceil \log_2 x \rceil := cond(2 \cdot x \dot{-} 2^{|x|}, |x| \dot{-} 1, |x|)$. We will frequently be lazy and write this as $\log x$.
- For A a Δ_1^b-predicate, R_2^1 can Σ_1^b-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_1^b-predicate, R_2^1 can Σ_1^b-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_2^i. 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_i^b-define a multifunction f if there is a Σ_i^b-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_{i+1}^b-definable functions of R_2^{i+1} are precisely the multifunctions in the class $FP^{\Sigma_i^p}(wit, \log^{O(1)})$, the class of multifunctions computable in polynomial time with $O(log^{O(1)})$ many queries to a Σ_i^p-oracle which returns witnesses for the truth of its "Yes" answers. It is useful, however, to further know that the Σ_i^b-definable multifunctions of R_2^i are closed under certain kinds of recursion. We now define two different kinds of weak recursion and show that the Σ_i^b-definable functions of R_2^i 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_2^i proves Σ_i^b-SLLIND.*
2. *The Σ_i^b-definable multifunctions of R_2^i are closed under the operations of composition, SW_2BPR and IC.*
3. *If f and g are Σ_i^b-defined functions in R_2^i then $f = g$ is Δ_i^b with respect to R_2^i.*
4. *The predicates Δ_i^b with respect to R_2^i are closed under boolean combinations, sharply bounded quantifications and substitutions by Σ_i^b-defined functions.*

PROOF. (Σ_i^b-*SLLIND*) It suffices to show for $A(x)$ a Σ_i^b-formula and m a fixed positive integer, that R_2^i proves $SLLIND_{A,m}$. This is proven by induction on m. The $m = 1$ case follows since R_2^i 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 $\lceil x_i \rceil$ 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, $\lceil\top\rceil$, or $\lceil\bot\rceil$. (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 $\lceil x_i \rceil$ then it has an ancestor labelled $\lceil \exists x_i \rceil$. (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 $\Sigma^q_i(\lceil A \rceil)$, $\Pi^q_i(\lceil A \rceil)$, and $\Delta^q_i(\lceil A \rceil)$. 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 $type_i(\lceil x_j \rceil, \lceil A \rceil)$. If there is a subformula w of $\lceil A \rceil$ beginning with $\lceil \exists x_j \rceil$ then $type_i(\lceil x_j \rceil, \lceil A \rceil)$ is the least number k less than i such that $w \in \Sigma^q_k$; otherwise, $type_i(\lceil x_j \rceil, \lceil A \rceil)$ 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 $\Gamma \rightarrow \Delta$ as an ordered pair of cedents.

We now want to define a predicate $TRU_i(\lceil A \rceil, \tau)$ which is true if $A \in \Sigma^q_i \cup \Pi^q_i$ and τ is a satisfying truth assignment for $\lceil A \rceil$. 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 $Wit^i(\lceil A \rceil, w, \tau)$. The formula $Wit^i(\lceil A \rceil, w, \tau)$ as we will define it checks whether replacing the variables x_j in A that have $type_i(x_j, \lceil A \rceil) = i$ 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 $\langle \lceil x_i \rceil, \lceil\top\rceil \rangle$ or $\langle \lceil x_i \rceil, \lceil\bot\rceil \rangle$ where x_i is the variable being assigned and $\lceil\bot\rceil$ or $\lceil\top\rceil$ 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 $Sub(\lceil A \rceil, \tau)$. This function returns a sequence which is the same as $\lceil A \rceil$ except that where a variable in $\lceil A \rceil$ appears in τ it is replaced by the value that τ assigns that variable. If $\lceil A \rceil$ is not a formula or τ is not a truth assignment then $Sub(\lceil A \rceil, \tau)$ is 0. It is not hard to Σ^b_1-define Sub in R^1_2 using IC.

In order to evaluate a Δ^b_0-formula all of whose leaves are either $\lceil\top\rceil$ or $\lceil\bot\rceil$ we first define a function $Evaltree_0(\lceil A \rceil)$ 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 $Evaltree_0(\lceil A \rceil)$ be the function with the following properties: (1) $Evaltree_0(\lceil A \rceil)$ outputs either 0 or a tree y with the same structure as $\lceil A \rceil$ but with different node labels. (2) It outputs 0 if any leaf of $\lceil A \rceil$ is a variable or if $\lceil A \rceil$ is not a QBF-formula. (3) Otherwise the leaves of y are labelled as in $\lceil A \rceil$. (4) If node i is labelled with a propositional connective in $\lceil A \rceil$ then i is labelled with $\lceil\top\rceil$ or $\lceil\bot\rceil$ in y according to the usual semantics of propositional logic from the values of its children. (5) If node i is labelled with an $\lceil \exists x_j \rceil$ in $\lceil A \rceil$ 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 $\lceil \exists x_j \rceil$).

Given that the Boolean Formula Value Problem (BFVP) is in NC^1 [7] it should seem plausible that we can Σ^b_1-define $Evaltree_0(\lceil A \rceil)$ 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^b_i-formulas

Consider a bounded formula $A(a_1, \ldots, a_k)$. The functions in the language L_2 are polynomial time computable, so there exists a polynomial $p_A(x)$ such that for $|n_1|, \ldots, |n_k| \leq m$ one only needs to compute numbers of length less than $p_A(m)$ to determine the truth of $A(n_1, \ldots, n_k)$. 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 $A(a_1, \ldots, a_k)$, which we will call $[\![A]\!]^m_{q(m)}$, using the variables

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

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

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

or

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

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 $\vec{p}_{a_1}, \ldots, \vec{p}_{a_k}$. 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 $\vec{\epsilon} = \epsilon^1, \ldots, \epsilon^{q(m)}$, 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 $S = \{(\epsilon_0, \ldots, \epsilon_{q(m)}) \in \{0,1\}^{q(m)+1} \mid (\forall i \geq |q(m)|)(\epsilon_i = 0)\}$.
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 $\exists x^0_y \ldots \exists x^{q(m)}_y [\![a \leq t \wedge A(a)]\!](\vec{x}_y/\vec{p}_a)$.
6. $[\![(\forall y \leq t)A(y)]\!]$ is $\forall x^0_y \ldots \forall x^{q(m)}_y [\![a \leq t \wedge A(a)]\!](\vec{x}_y/\vec{p}_a)$.

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 $A(a_1, \ldots, a_k)$ be a Σ^b_i-formula and $q(x)$ a bounding polynomial for A then*

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

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 $\rho(x_1, \ldots, x_k)$ be the Σ_1^b-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

$$TRU_0(\llbracket A \rrbracket^m_{q(m)}, \rho(x_1, \ldots, x_k)) \equiv$$
$$(|x_1| \leq m \wedge \ldots |x_k| \leq m \supset A(\vec{x}))$$

is a Δ_1^b-predicate in R_2^1 since the function $I^m \mapsto \llbracket A \rrbracket^m_{q(m)}$ and the function ρ are Σ_1^b-definable in R_2^1. Thus, R_2^1 using Δ_1^b-$LIND$ on $B(m)$ can prove the base case. Since $\llbracket A \rrbracket^m_{q(m)}$ is Δ_0^q it is not hard to show from the definition of TRU_i that $TRU_0(\llbracket A \rrbracket^m_{q(m)}, \rho) \Leftrightarrow TRU_i(\llbracket A \rrbracket^m_{q(m)}, \rho)$. □

6. Proof Systems

We proceed with the task of defining a propositional proof system for R_2^i. 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 $\Sigma_i^q \cup \Pi_i^q$-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*

$$(\forall d, A)(Q(d, \ulcorner A \urcorner) \wedge A \in \Sigma_j^q \cup \Pi_j^q \supset R(f(d, \ulcorner A \urcorner), \ulcorner A \urcorner)$$

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:*

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

Since TRU_i is a $B(\Sigma_i^b)$-formula, i-$RFN(P)$ is a $\forall\Sigma_i^b$-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:

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

(b) propositional quantifier rules defined as follows:

$$(\forall\text{:left}) \quad \frac{A(\vec{B}), \Gamma \to \Delta}{\forall \vec{x} A(\vec{x}), \Gamma \to \Delta} \qquad (\forall\text{:right}) \quad \frac{\Gamma \to \Delta, A(\vec{p})}{\Gamma \to \Delta, \forall \vec{x} A(\vec{x})}$$

$$(\exists\text{:left}) \quad \frac{A(\vec{p}), \Gamma \to \Delta}{\exists \vec{x} A(\vec{x}), \Gamma \to \Delta} \qquad (\exists\text{:right}) \quad \frac{\Gamma \to \Delta, A(\vec{B})}{\Gamma \to \Delta, \exists \vec{x} A(\vec{x})}$$

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 resiricted 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:*

$$R_2^1 \vdash \forall y (P \vdash \llbracket A \rrbracket^{|y|}_{p(|y|)}).$$

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 $A_1, \ldots, A_n$ 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 $A_1, \ldots, A_n$ be a proof in F of A_n. Let B_j be the balanced conjunction $\bigwedge_{i=1}^{j} A_i$. 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:

$$\dfrac{\dfrac{[d]}{B_j \rightarrow A_k} \quad \dfrac{\dfrac{\dfrac{A_k \rightarrow A_k}{A_k \rightarrow A_k, A_{j+1}}}{A_k \rightarrow A_{j+1}, A_k} \quad \dfrac{\dfrac{\dfrac{A_{j+1} \rightarrow A_{j+1}}{A_k, A_{j+1} \rightarrow A_{j+1}}}{A_{j+1}, A_k \rightarrow A_{j+1},}}{A_k \supset A_{j+1}, A_k \rightarrow A_{j+1}}}{B_j, A_k \supset A_{j+1} \rightarrow A_{j+1}}$$

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 $B_j \rightarrow B_{j+1}$ 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

$$\dfrac{\dfrac{\rightarrow \top}{\rightarrow \top^{n^{7+m}}} \quad \dfrac{[P]}{\dfrac{\rightarrow A_n}{\top^{n^{7+m}} \rightarrow A_n}}}{\rightarrow A_n}$$

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 $\Sigma^q_1 \cup \Pi^q_1$-formula used in our construction this will be an L^*_1-proof.

To formalize this simulation in R^1_2 is not hard and we only give the barest outline. First, R^1_2 can Σ^b_1-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^b_1-define a function using IC which given j and $A_1, \ldots, A_n$ outputs a L^*_1-proof $B_j \rightarrow B_{j+1}$ for $j > 1$ and $\rightarrow B_1$ for $j = 1$. From this function in turn we can Σ^b_1-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^i_2. Since $i\text{-}RFN(L^*_i)$ is a $\forall\Sigma^b_i$-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^b_{i+1}-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^q_i-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^q_i-formula. We now argue Wit^i is a Δ^b_i-predicate in R^i_2. From the definition of $WitTree_i$ the theory R^i_2 can prove if $\ulcorner A \urcorner$ is a Σ^q_i-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^i_2 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^b_i-definable function in R^i_2. So by Lemma 3 Wit^i is a Δ^b_i-predicate in R^i_2.

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

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

and

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

Since Wit^i is Δ^b_i with respect to R^i_2 by Lemma 3, both these predicates are also Δ^b_i with respect to R^i_2. Before we make precise how we will use the witness predicates to prove $i\text{-}RFN(L^*_i)$, let us establish what R^i_2 can say about witnessing.

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

$$R_2^1 \vdash Wit^i(\lceil A \rceil, w, \tau) \supset TRU_i(\lceil A \rceil, \tau)$$

PROOF. From the definition of Wit^i, the theory R_2^1 proves $Wit^i(\lceil A \rceil, w, \tau)$ implies $(\exists y)(Evaltree_i^{\exists}(\lceil A \rceil, \tau)) = y \wedge \beta(1, y) = \lceil \top \rceil)$. This then implies $TRU_i(\lceil A \rceil, \tau)$. □

The next result will allow us to prove $i\text{-}RFN(L_i^*)$ in R_2^i. Before we state it, we define the functions $l(\lceil \Gamma \rightarrow \Delta \rceil)$ and $r(\lceil \Gamma \rightarrow \Delta \rceil)$ which we will need because R_2^i has only a limited ability to manipulate L_i^*-proofs. For instance, given an L_i^*-proof of a sequent of Σ_i^q-formulas, it seems complicated to show R_2^i can transform this proof into one with only Σ_i^q-formulas.

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

Theorem 13. *($i \geq 1$) Let τ be a free variable. There is a Σ_i^b-definable in R_2^i multivalued function $ProofWit_i$ such that:*

$$R_2^i \vdash \forall d(d : L_i^* \vdash \Gamma \rightarrow \Delta \supset (Wit_{\wedge}^i(\lceil \Gamma^* \rceil, w, \tau) \supset Wit_{\vee}^i(\lceil \Delta^* \rceil, ProofWit_i(w, d, \tau))).$$

PROOF. $ProofWit_i$ is Σ_i^b-defined in R_2^i from three Σ_i^b-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_i^q-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_2^i 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 $Wit_{\wedge}^i$ or $Wit_{\vee}^i$ formulas. We will show the bounds on the height of an L_i^*-proof and on the number of Σ_i^q-cuts on a branch enables R_2^i to show $ProofWit_i$ eventually computes a witness for the succedent of the endsequent and since R_2^i 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

$$\langle \lceil \Gamma \rightarrow \Delta \rceil, w, w', \tau \rangle.$$

where $\lceil \Gamma \rightarrow \Delta \rceil$ 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 $\Gamma \rightarrow \Delta$. 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 $\Gamma \rightarrow \Delta$ into an L_i^* labelled proof d' where the root in d has been replaced with

$$\langle \lceil \Gamma \rightarrow \Delta \rceil, w, 0, \tau \rangle \tag{1}$$

and any cuts in d' are on Σ_i^q-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 $\Gamma \rightarrow \Delta$ 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 $\Gamma' \rightarrow \Delta'$ and $CompAnt_i$ can compute a witness w' for the $(\Gamma')^*$ and also a truth assignment τ' for $\Gamma' \rightarrow \Delta'$, then this sequent is replaced with

$$\langle \lceil \Gamma' \rightarrow \Delta' \rceil, w', 0, \tau' \rangle.$$

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_i^b-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

$$R_2^i \vdash Wit_{\wedge}^i((\Gamma')^*, w', \tau') \supset Wit_{\wedge}^i((\Gamma'')^*, w'', \tau'').$$

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

$$R_2^i \vdash Wit_{\wedge}^i(\Gamma^*, w, \tau) \supset Wit_{\wedge}^i((\Gamma')^*, w', \tau').$$

We now describe how $AntStep_i(n, d')$ is defined. It is defined in R_2^i using IC from the multifunction $AntCopy_i(j, n, d')$ which we will show is Σ_i^b-defined in R_2^i. 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

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

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 $\Gamma' \rightarrow \Delta', A$ and the case where the node j is $A, \Gamma' \rightarrow \Delta'$. 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 $\Gamma' \rightarrow \Delta', A$. Thus, we can use w' as a witness for the antecedent of $\Gamma' \rightarrow \Delta', A$ and τ' as a truth assignment for this sequent. Obviously,

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

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

In the second case, we are trying to find a witness for $A, (\Gamma)^*$ and a truth assignment for $A, \Gamma' \rightarrow \Delta'$. The multifunction $AntCopy_i$ first checks the node containing $\Gamma' \rightarrow \Delta', A$. 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 $A, \Gamma' \rightarrow \Delta'$. 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 $\Gamma' \rightarrow \Delta', A$ 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 $\Gamma' \rightarrow \Delta', A$ 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 $A, \Gamma' \rightarrow \Delta'$. It is not hard to see that

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

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

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

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

The two subcases we need to consider are the case where $\exists \vec{x} A$ is $\Sigma^q_i \backslash \Pi^q_i$-formula and the case where $\exists \vec{x} A$ is Σ^q_{i-1}-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 $type_i(\lceil y_j \rceil, \lceil A \rceil) = i$. 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

$$R^i_2 \vdash Wit^i_\wedge(\lceil \exists \vec{x} A, (\Gamma')^* \rceil, w', \tau') \supset Wit^i_\wedge(\lceil A, (\Gamma')^* \rceil, w'', \tau'').$$

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

In the second case, since $\exists \vec{x} A$ is a Σ^q_{i-1}-formula, we can use w' as our witness for $A, (\Gamma')^*$. To modify our truth assignment we make a witness query to a Σ^p_{i-1}-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

$$R^i_2 \vdash Wit^i_\wedge(\lceil \exists \vec{x} A, (\Gamma')^* \rceil, w', \tau') \supset Wit^i_\wedge(\lceil A, (\Gamma')^* \rceil, w', \tau'').$$

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

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^b_i-definable in R^i_2 the multifunction $AntCopy_i(j,n,d')$ is Σ^b_i-definable in R^i_2. As stated before $AntStep_i(n,d')$ is defined from $AntCopy_i$ using IC, so it too will be Σ^b_i-defined in R^i_2. Finally, $CompAnt_i(d')$ will be Σ^b_i-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^b_1-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^b_i-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

$$R_2^i \vdash Wit_\vee^i((\Delta'')^*, w'', \tau'') \supset Wit_\vee^i((\Delta')^*, w', \tau')$$

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_i^b-$SLLIND$ on the height of a L_i^*-proof d of $\Gamma \rightarrow \Delta$ one can show the following: If w is a witness for Γ^* and τ is a truth assignment for $\Gamma \rightarrow \Delta$ and there is a sequent $\Gamma' \rightarrow \Delta'$ 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

$$R_2^i \vdash Wit_\wedge^i(\Gamma^*, w, \tau) \supset Wit_\vee^i((\Delta')^*, w'', \tau').$$

We now describe how $SuccStep_i(n, d')$ is defined. Basically, it is defined using IC from a Σ_i^b-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

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

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 $A \rightarrow A$, a witness w' for A^* (the formula A may be Π_i^q), a witness 0 for the succedent, and a truth assignment τ' for the sequent, we define $SuccCopy_i(j, n, d')$ to be

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

For initial sequent of the form $\rightarrow \top$ and $\bot \rightarrow$ 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_0^q-sentences.

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

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

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

$$\Gamma' \rightarrow \Delta'.$$

We also have by hypothesis witnesses v' for $(\Delta')^*, A$ and maybe also a v'' for $(\Delta')^*$. Let w'' be $Front(v')$ if $Wit_\vee^i(\ulcorner (\Delta')^* \urcorner, Front(v'), \tau')$ and otherwise let w'' be v'' if $Wit_\vee^i(\ulcorner (\Delta')^* \urcorner, v'', \tau')$. In these cases we define $SuccCopy_i(j, n, d')$ to be $\langle \ulcorner \Gamma \rightarrow \Delta \urcorner, w', w'', \tau' \rangle$. It is not hard to see in these cases that R_2^i proves w'' is the desired witness. If both of these two cases do not hold than we just output the node as is. Since $Wit_\vee^i$ is a Δ_i^b-predicate we can Σ_i^b-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^i_2 can prove $ProofWit_i(d, w, \tau)$ computes what it is supposed to, one uses Σ^b_i-$SLLIND$ on this maximum number. For the induction step one uses the fact that we have already shown that R^i_2 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 $\Sigma^q_i \cup \Pi^q_i$-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

$$R^i_2 \vdash i\text{-}RFN(L^*_{i,m}).$$

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

$$R^i_2 \vdash \forall d(d : L^*_i \vdash A \supset \\ Wit^i_\wedge(\langle\rangle, \langle\rangle, \tau)) \supset Wit^i_\vee(\ulcorner A \urcorner, ProofWit(\langle\rangle, d, \tau), \tau)).$$

This implies

$$R^i_2 \vdash \forall d(d : L^*_i \vdash A \supset Wit^i(\ulcorner A \urcorner, ProofWit(\langle\rangle, d, \tau), \tau))).$$

By Proposition 12, we have

$$R^i_2 \vdash \forall d(d : L^*_i \vdash A \supset TRU_i(A, \tau)).$$

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

Theorem 15. *($i \geq 1$) The theory R^i_2 proves i-$RFN(L^*_i)$.*

8. Simulating the $\forall\Sigma^b_i$-consequences of R^i_2

We now show L^*_i can simulate the $\forall\Sigma^b_i$-consequences of R^i_2.

Theorem 16. *For $i \geq 1$, L^*_i simulates $\forall\Sigma^b_i(R^i_2)$.*

PROOF. Let A be a Σ^b_i-formula provable in R^i_2. Using some R^i_2-proof d of A, we want to show

$$R^1_2 \vdash \forall y(L^*_i \vdash [\![A]\!]^{|y|}_{p(|y|)}) \tag{2}$$

where p is some polynomial that bounds the size of all quantified propositional translations of formulas in d. By cut-elimination for R^i_2 we can choose a proof d of A such that all the formulas in d are Σ^b_i-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 $\Sigma^q_i \cup \Pi^q_i$-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 $\Sigma^q_i \cup \Pi^q_i$-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^q_0-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 $\Sigma^q_i \cup \Pi^q_i$-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 $\Sigma^q_i \cup \Pi^q_i$-cuts will remain unchanged for all of these inferences except a cut rule whose maximum number of $\Sigma^q_i \cup \Pi^q_i$-cuts along any branch can be bounded by

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

(∀:left)

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

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 $\Sigma^q_i \cup \Pi^q_i$-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_0^q-formulas $\vec{A}_t$ which make up the translation of t we can derive:

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

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 $\Sigma_i^q \cup \Pi_i^q$-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_2^1 proves there are G_i^* proofs of

$$[\![B(t)]\!], [\![\Gamma]\!] \rightarrow [\![\Delta]\!] \tag{7}$$

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

$$[\![t \leq s]\!], [\![B(t)]\!], [\![\Gamma]\!] \rightarrow [\![\Delta]\!]. \tag{8}$$

If $B(a)$ is atomic then we have already said there are G_1^*-proofs of the translation of the equality axiom

$$[\![t = a]\!], [\![B(a)]\!] \rightarrow [\![B(t)]\!] \tag{9}$$

with height bounded by $h_B \cdot \log(|y|)$ and all the cuts of Δ_0^q-formulas. By induction on the complexity of B the theory R_2^1 can show there are G_i^*-proofs of these sequents for $B \in \Sigma_i^q \cup \Pi_i^q$ and where the height of these G_i^*-proofs is bounded by $h_B \cdot \log(|y|) + h_B$ and where the number of $\Sigma_i^q \cup \Pi_i^q$-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 $[\![\Gamma]\!]$ and $[\![\Delta]\!]$ to (9), performing some exchanges and then cutting the result against (8) we derive:

$$[\![t \leq s]\!], [\![t = a]\!], [\![B(a)]\!], [\![\Gamma]\!] \rightarrow [\![\Delta]\!]. \tag{10}$$

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

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

where S is the set used for translations of sharply bounded quantifiers in the definition of $[\![\cdot]\!]$. 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

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

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

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

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

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

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)

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

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

$$[\![B(\lfloor \frac{1}{2} a \rfloor)]\!], [\![\Gamma]\!] \rightarrow [\![\Delta]\!], [\![B(a)]\!]$$

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:

$$[\![B(MSP(|t|, k+1))]\!], [\![\Gamma]\!] \rightarrow [\![\Delta]\!], [\![B(MSP(|t|, k))]\!] \tag{15}$$

Now cutting formulas of this type against each other in a balanced fashion we can derive the desired formula $[\![B(0)]\!], [\![\Gamma]\!] \rightarrow [\![\Delta]\!], [\![B(|t|)]\!]$. 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

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

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

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

Thus, R^1_2 proves

$$L^*_i \vdash\!\!\rightarrow \llbracket TRU_j(A,\tau)\rrbracket^{|y|}_{q'(|y|)} \tag{18}$$

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 $\llbracket(\exists z)(Evaltree^{\exists}_j(Sub(A),\tau) = z \wedge \beta(1,z) = \ulcorner\top\urcorner)\rrbracket^{|y|}_{q'(|y|)}$. Then R^1_2 proves

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

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

$$L^*_i \vdash \llbracket TRU_j(A,\tau)\rrbracket^{|y|}_{q''(|y|)} \rightarrow A$$

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

$(\forall y)(L^*_i \vdash \llbracket A\rrbracket^{|y|}_{q(|y|)})$ for some bounding polynomial q. So by Theorem 15, the theory T proves

$$(\forall y,\tau)(Assign(\tau) \supset TRU_j(\llbracket A\rrbracket^{|y|}_{q(|y|)},\tau))$$

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

Corollary 20. *$(0 \leq j \leq i, 1 \leq m)$ R^1_2 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^i_k. 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

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

Algebraic models of computation and interpolation for algebraic proof systems

Pavel Pudlák and Jiří Sgall

1. Introduction

We consider some algebraic models used in circuit complexity theory and in the study of the complexity of the propositional calculus. This direction of research has been getting attention recently with the hope that the connection to well-developed fields of mathematics like algebra can be helpful in proving lower bounds.

Span programs as a model of computation were introduced in [13]. A span program is a device for defining boolean functions, where the function is defined to be 1 iff a fixed vector can be expressed as a linear combination of vectors chosen by the input. Span programs polynomially simulate branching programs (for finite fields they are equivalent to counting branching programs), thus an exponential lower bound on the size of span programs computing a concrete boolean function would solve a major open problem. An important subclass of span programs are monotone span programs; they simulate both monotone formulas and monotone contact switching networks. Recently a superpolynomial lower bound was proved for this model [1, 2] (based on a combinatorial condition of [4]).

One direction of study of propositional proof systems is to prove lower bounds on the length of proofs in certain restricted proof systems. Exponential lower bounds were obtained for such systems as resolution [12], bounded depth Frege systems [15, 17], cutting planes [6, 18], and Nullstellensatz refutations [3, 8]. In this paper we are interested in systems that use uses polynomials instead of boolean formulas. From the previous list this includes the Nullstellensatz refutations. Recently a stronger system using polynomials was proposed, the polynomial calculus, also called the Groebner calculus [9].

The proof systems form a similar hierarchy as the complexity classes or classes of circuits in the computational complexity, but there is no direct relation between the two hierarchies. Recently a new method was found which makes it possible to prove lower bounds on the length of proofs for some propositional proof systems using lower bounds on circuit complexity. This method is based on proving computationally efficient versions of Craig's interpolation theorem for the proof system in question [14, 18]. For appropriate tautologies the interpolation theorem

1991 *Mathematics Subject Classification.* Primary 68Q99; Secondary 68Q05, 03B05.

This work was partially supported by grant A1019602 of AV ČR, by grant 93025 of the US-Czechoslovak Science and Technology Program and by cooperative research grant INT-9600919/ME-103 from the NSF (USA) and the MŠMT (Czech republic).

may lead to a monotone model of computation, which makes it possible to use available lower bounds for monotone models of computation to prove lower bounds for proof systems. Such interpolation theorem yields a simple proof of exponential lower bounds for resolution using known lower bounds for monotone boolean circuits [18]. It was also applied to yield an exponential lower bound for unrestricted cutting planes [18], in which case the proof is based on the new bounds for monotone real circuit [11, 18].

The main result of this paper is an interpolation theorem for Nullstellensatz refutations. We prove that the interpolants can be computed by span programs, and the span programs are monotone for suitable tautologies. Moreover, this characterization of interpolants is tight, namely every span program is a unique interpolant for some tautology provable by the Nullstellensatz system. In principle, this interpolation theorem can be used for proving lower bounds on Nullstellensatz proofs using lower bounds on monotone span program complexity, but due to the difficulty of proving lower bounds for monotone span programs known direct proofs are simpler and yield better bounds.

We introduce a new computational model, polynomial programs, which bounds the complexity of the interpolants in polynomial calculus in a similar way as span programs do in case of the Nullstellensatz system. The general version of polynomial programs over finite fields is equivalent to boolean circuits. The monotone version simulates both monotone boolean circuits and monotone span programs, thus it is a very strong monotone computational model. Unfortunately the lower bound technique for monotone span programs does not extend to monotone polynomial programs. Our original motivation for studying this model was to get non-trivial lower bounds for the polynomial calculus. During the refereeing process Razborov proved such a bound without using interpolation theorems [21]. However, lower bounds for polynomial programs would still be interesting.

We introduce yet another model of computation, dependency programs. These programs are similar to span programs but only linear dependence of chosen vectors is tested, instead of testing whether their span contains some vector. A communication complexity version of this model was studied in [19] as the projective dimension of graphs (a concept also related to the affine dimension of graphs of [20]). We prove an exponential lower bound for monotone dependency programs, using a simplification of the methods used for monotone span programs in [1, 4]. In the non-monotone case over finite fields the dependency programs turn out to be equivalent to span programs. However, they may be useful to consider in lower bound proofs, as they are in some sense simpler (as demonstrated by the monotone lower bound, which is much simpler than the analogous bound for span programs).

Finally, we investigate the closure properties and relations among the algebraic models of computation. We give simple constructions for some closure properties which in some cases generalize the known results; for example we show that span programs are closed under NC^1-reductions for arbitrary fields. One interesting open question is the relative strength of these models of computation when the underlying field is changed. We note that the most straightforward attempt to convert a span program over $\mathbf{R}$ into a span program over $\mathbf{Q}$ fails, since there exist matroids representable over $\mathbf{R}$ but not over $\mathbf{Q}$.

We start by the definitions in Section 2 and the properties of the computational models in Section 3. The main results on interpolation are proved in Section 4. The result on matroids is given in Section 5.

2. Definitions

We assume that a field K is fixed. Most often we consider the finite fields $GF(p)$ with p elements for p prime, or rational numbers $\mathbf{Q}$ or reals $\mathbf{R}$.

As usual, for a matrix $\mathbf{A}$, a_{ij} denotes its entry in ith row and jth column. The column vectors are denoted by $\vec{u} = (u_1, \ldots, u_k)^\top$, etc. The dot product of vectors $\vec{u}$ and $\vec{v}$ is $\vec{u}^\top \vec{v}$. Some special vectors are denoted by $\vec{0} = (0, \ldots, 0)^\top$, $\vec{1} = (1, \ldots, 1)^\top$, and $\vec{e} = (1, 0, \ldots, 0)^\top$. The unit matrix is denoted by $\mathbf{I}$ and the matrix with all entries zero is $\mathbf{0}$. The length of these vectors and the dimensions of the matrices will always be clear from the context.

2.1. Algebraic proof systems. A *Nullstellensatz refutation* (shortly *NS refutation*) of a set of polynomial equations $p_1 = 0, \ldots, p_m = 0$ is given by a system of polynomials $q_1, \ldots, q_m$ such that $\sum q_i p_i = 1$. The degree of such a refutation is $d = \max_i deg(q_i p_i)$.

This refutation system is complete if we look for solutions in an algebraically closed field (by the Hilbert Nullstellensatz). Here we are interested only in 0-1 solutions. To get the completeness for such solutions, we shall assume that for each used variable x_k the equation $x_k^2 - x_k = 0$ is present among $p_1 = 0, \ldots, p_m = 0$.

A *polynomial refutation* of a set of polynomial equations $p_1 = 0, \ldots, p_m = 0$ is sequence of polynomials $q_1, \ldots, q_k$ such that q_k is the constant 1 and each q_i is either some p_j, or a linear combination of the polynomials $q_1, \ldots, q_{i-1}$, or $x_l q_j$ for some $j < i$ and some variable x_l. The degree of the refutation is $d = \max_i deg(q_i)$.

2.2. Algebraic models of computation. For our purposes *literals* are denoted by x_i or $1 - x_i$, the latter corresponding to the negation of a variable. A *labelled matrix* is a matrix such that each row is labelled by some literal. Given a labelled matrix $\mathbf{A}$ and a truth assignment $\vec{u}$, we define $\mathbf{A}(\vec{u})$ to be the matrix consisting only of those rows of $\mathbf{A}$ labelled by literals that are true in the assignment $\vec{u}$ (i.e., either the row is labelled by x_i and $u_i = 1$, or the row is labelled by $1 - x_i$ and $u_i = 0$ in the assignment).

A *dependency program* is given by a labelled matrix $\mathbf{A}$. Its value on the assignment $\vec{u}$ is defined to be 1 if the rows of $\mathbf{A}(\vec{u})$ are linearly dependent, and 0 otherwise. The size of a dependency program is the number of columns of the matrix.

A *span program* is given by a labelled matrix $\mathbf{A}$. Its value on the assignment $\vec{u}$ is defined to be 1 if the vector $\vec{e}^\top = (1, 0, \ldots, 0)$ is a linear combination of the rows of $\mathbf{A}(\vec{u})$, and 0 otherwise. The size of a span program is the number of columns of the matrix. Note that nothing changes if we choose any non-zero vector instead of $(1, 0, \ldots, 0)$, as we can transform the basis of the vector space.

A *polynomial program of degree d* is given by a set of polynomials in $K[y_1, \ldots, y_m]$ ($y_1, \ldots, y_m$ are new formal variables), where each polynomial is labelled by a literal (x_i or $1 - x_i$). Its value on an assignment $\vec{u}$ is defined to be 1 if there exists a polynomial refutation of degree d from all those vectors that are labelled by literals that are true in the assignment $\vec{u}$. The size of a polynomial program is the number of monomials of $K[y_1, \ldots, y_m]$ of degree at most d, which is equal to $\sum_{i=0}^{d} \binom{m}{d}$. In particular, if d is a constant, the size is polynomial in the number of variables m.

A dependency, span, or polynomial program is *monotone*, if all the labels (of the vectors or polynomials) are positive literals. Clearly, the monotone programs

compute only monotone functions. Using these definitions it would be impossible to compute the constant 1 function in any of the monotone models. For that reason we augment our definitions so that the empty program is defined to compute the constant 1 in any of the models, and it is considered to be monotone. (An equivalent definition would be to allow also 1 as a label of a row, not only a literal. See Section 3.2 for more discussion of this.)

The minimal size of a dependency, span, or polynomial program computing a function f is denoted $DP_K(f)$, $SP_K(f)$, or $PP_{d,K}(f)$. The monotone variants are denoted $mDP_K(f)$, etc. The index K is omitted if the field is clear from the context; also for finite fields we write $SP_p(f)$ if the field is $GF(p)$. Note that there is some ambiguity in the literature regarding to the measure of size: most of the previous papers measure the number of rows instead of the number of columns. As discussed in Section 3.2, this is not essential for span or dependency programs, as these two measures differ at most by a factor of n. However, for polynomial programs the number of columns can be significantly larger, and hence it is a more appropriate measure.

If the arithmetic operation in the given field can be implemented efficiently (which is true of all finite fields), the dependency and span programs are efficient procedures, as their value can be found using the Gaussian elimination. An efficient decision procedure for polynomial programs follows from [9]; it uses the Groebner basis algorithm which generalizes the Gaussian elimination appropriately.

The relations among these models and some other variants will be studied in Section 3.2. It is easy to see that the models are increasingly more powerful. In the non-monotone case the polynomial programs over any finite field are polynomially equivalent to boolean circuits, and the dependency programs are polynomially equivalent to the span programs over the same field $GF(p)$. However, monotone span programs are exponentially stronger than monotone dependency programs.

3. Relations among the algebraic models of computation

3.1. An exponential lower bound for monotone dependency programs. This bound is based on the ideas of the papers [1, 4], which prove superpolynomial lower bounds for monotone span programs. Using their methods, we are able to prove an exponential lower bound on the size of monotone dependency programs for a very simple function.

THEOREM 3.1. *Let $f = (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge \ldots \wedge (x_{2n-1} \vee x_{2n})$. Then $mDP(f) \geq 2^n/n$, for an arbitrary field.*

We start by a lemma about minterms of any function computed by a monotone dependency program. A *minterm* of a function as is an assignment $\vec{u}$ such that $f(\vec{u}) = 1$ and for any $\vec{v} \leq \vec{u}$, $f(\vec{v}) = 0$ or $\vec{v} = \vec{u}$.

LEMMA 3.2. *Suppose that a boolean function f is computed by a monotone dependency program $\mathbf{A}$ with the number of rows smaller than the number of minterms of f. Then there exists a set of minterms U, $|U| \geq 2$, such that for any non-trivial partition $U = V \cup W$, $V, W \neq \emptyset$, $V \cap W = \emptyset$, there exists a minterm $\vec{u}$ of f such that for every i, $u_i = 1$ implies that both $(\exists \vec{v} \in V) v_i = 1$ and $(\exists \vec{w} \in W) w_i = 1$.*

PROOF. For every minterm $\vec{u}$ of f chose some linear dependence $\vec{c}^{\vec{u}}$ of the rows of $\mathbf{A}$ consistent with the labels of $\mathbf{A}$, i.e., $\vec{c}^{\vec{u}}$ is a vector such that $(\vec{c}^{\vec{u}})^\top \mathbf{A} = 0$, and

if $c^{\vec{u}}_j \neq 0$ and the jth row of $\mathbf{A}$ is labelled by x_i then $u_i = 1$. Such $\vec{c}^{\vec{u}}$ exists since $\vec{u}$ is accepted by $\mathbf{A}$.

The vectors $\vec{c}^{\vec{u}}$ are linearly dependent, as their length is the number of rows which is smaller than the number of minterms u. Let U be a minimal set of minterms such that $\{\vec{c}^{\vec{u}} \mid \vec{u} \in U\}$ is linearly dependent. Thus $\sum_{\vec{u} \in U} \alpha_{\vec{u}} \vec{c}^{\vec{u}} = \vec{0}$ for some $\alpha_{\vec{u}} \neq 0$. Now for any nontrivial partition $\vec{c} = \sum_{\vec{u} \in V} \alpha_{\vec{u}} \vec{c}^{\vec{u}} = -\sum_{\vec{u} \in W} \alpha_{\vec{u}} \vec{c}^{\vec{u}} \neq \vec{0}$. Define $u'_i = 1$ if there exist j such that $c_j \neq 0$ and jth row of A is labeled by x_i; let $\vec{u}$ be any minterm smaller than or equal to $\vec{u}'$ (i.e., $u_i \leq u'_i$ for all i). This $\vec{u}$ satisfies the conditions in the statement of the lemma. □

PROOF. (of Theorem 3.1.) An assignment $\vec{u}$ is a minterm of f from the statement of the theorem iff $u_{2i-1} + u_{2i} = 1$ for all $i = 1, \ldots, n$; there are 2^n minterms.

Suppose that f is computed by a monotone dependency program of size less than $2^n/n$. Then it is computed also by a monotone dependency program with less than 2^n rows (since all the rows labelled by the same x_i are either independent, or can be replaced by a single vector $\vec{0}$, cf. Section 3.2).

Let U be the set of minterms guaranteed by the lemma. Pick i and $\vec{v}, \vec{w} \in U$ such that $v_{2i-1} = 0$ and $w_{2i} = 0$; these exists because $|U| \geq 2$ and because of the particular structure of the minterms. Now set $V = \{\vec{v} \in U \mid v_{2i-1} = 0\}$ and $W = \{\vec{w} \in U \mid w_{2i} = 0\}$. Let $\vec{u}$ be a minterm guaranteed by the lemma. By the definition of V and W it follows that $u_{2i-1} = u_{2i} = 0$, which is impossible for a minterm, a contradiction. Hence f has no small monotone dependency programs. □

3.2. Closure properties and some variants of the definitions. First we prove that our models are closed under restrictions. For span programs this was noticed already in [13], using a more complicated argument.

LEMMA 3.3. *If g is a restriction of a function f, then $mDP(g) \leq mDP(f)$ for an arbitrary field K; similarly for span and polynomial programs and also for the non-monotone versions.*

PROOF. Suppose that x_i is assigned a constant, hence some rows (or polynomials for polynomial programs) are now labelled by 0 or 1 instead of a literal. We remove all rows labelled by 0.

For span and polynomial programs we replace each row labelled by 1 by multiple copies labelled by all the possible literals, using only monotone literals in the monotone case. The new program computes the restriction except for the case when it is the constant 1 function, which is by definition computed by the empty program. Monotonicity of the program is preserved and the size does not increase.

For dependency programs for each row labelled by 1 we change the basis so that it is the first basis vector and remove the first column of the program. This does not change the computed function, as the row can be used on any input and hence anything in the first column can be cancelled. If the row is $\vec{0}$, we cannot perform the previous transformation; however, the computed function is the constant 1 function, which is by definition computed by the empty program. Monotonicity of the program is preserved and the size does not increase. □

There are several variations in the definitions we can make. First, we can allow a row to be labelled by 1 instead of a literal, with the meaning that it can be used for any input. This is essentially the same as taking a restriction, hence by previous

lemma it does not change the size of the program. We will use this generalization in our constructions.

Second, we can measure the number of rows instead of the number of columns. For a minimal program, the number of rows is larger than the number of columns by at most a factor of $2n$, as we can take all the rows labelled by the same literal linearly independent. For the dependency or span programs even the number of rows does not increase if we take a restriction of the function; for the dependency programs it follows from the argument in the lemma, and for span programs it is possible to use a similar argument, too. By the same argument the factor between the number of rows and columns can be tightened from $2n$ to n. Also, for dependency and span programs the number of rows is always at least the number of columns, as we can work in the span of all the rows.

Third, it is possible to consider a more general variant of span programs where we have not one target vector but a vector subspace, with the zero vector excluded. If the target vectors are the first k basis vectors and the generalized span program is given by $(\vec{a}^1 \cdots \vec{a}^k\, \mathbf{A})$ (the first k columns of the matrix written separately), then it computes the same function as the disjunction of the k span programs $(\vec{a}^1\, \mathbf{A}),\ldots,$ $(\vec{a}^k\, \mathbf{A})$. In the next lemma we will prove that span programs are closed under disjunctions, hence the size of the usual span program is at most the square of the size of the generalized one.

Now we prove that span programs are closed under NC^1-reductions and monotone span programs are closed under monotone NC^1-reductions. This was known for non-monotone span programs over $GF(p)$, due to their equivalence to counting branching programs [7, 13], see also Section 3.3. Our proof is direct and works for an arbitrary field.[1]

We say that a function f is NC^1-reducible to a function g if there exists a family of circuits of depth $O(\log n)$ computing f with gates for NOT, OR and AND of fan-in two, and gates for g; the gates for g of fan-in k count as depth $\log k$. The reduction is monotone if there are no NOT gates in the circuits. We consider non-uniform reductions, since we consider non-uniform models of computation, unlike e.g. [7, 10].

THEOREM 3.4. *For arbitrary field K and all boolean functions f, g, g_1, ..., g_k,*

- $mSP(f \vee g) \le mSP(f) + mSP(g) - 1$, $SP(f \vee g) \le SP(f) + SP(g) - 1$,
- $mSP(f \wedge g) \le mSP(f) + mSP(g)$, $SP(f \wedge g) \le SP(f) + SP(g)$,
- $SP(\neg f) \le n \cdot SP(\neg f)$, *where n is the arity of f,*
- $mSP(f(g_1,\ldots,g_k)) \le mSP(f)(mSP(g_1) + \cdots + mSP(g_k))$.

Consequently, if f is (monotone) NC^1-reducible to g and g has a (monotone) span program of polynomial size, f has a (monotone) span program of polynomial size.

PROOF. Suppose that we have two span programs $(\vec{a}\,\mathbf{A})$ and $(\vec{b}\,\mathbf{B})$ ($\vec{a}$ and $\vec{b}$ are their first columns). Then the span program

$$\begin{pmatrix} \vec{a} & \mathbf{A} & 0 \\ \vec{b} & 0 & \mathbf{B} \end{pmatrix}$$

[1] It was pointed out by the referee that similar constructions are known for at least the first three properties, but were not published before.

with the same labels computes their disjunction. The span program

$$\begin{pmatrix} \vec{a} & \vec{0} & \mathbf{A} & 0 \\ \vec{0} & \vec{b} & 0 & \mathbf{B} \end{pmatrix}$$

with the same labels computes their conjunction, if the target vector is $(1,1,0,\ldots,0)$ instead of $(1,0,\ldots,0)$; the target vector can be changed by a linear transformation of the matrix.

Now we construct a span program which computes $\neg f$ given a span program $\mathbf{A}$ for f. Let $\vec{u}$ be an input for $\mathbf{A}$. Consider the matrix

$$\overline{\mathbf{A}(\vec{u})} = \begin{pmatrix} \vec{e} & \mathbf{A}(\vec{u})^\top \end{pmatrix}$$

By the linear programming duality, $\mathbf{A}$ does not accept $\vec{u}$ iff $(1,0,\ldots,0)$ is in the span of the rows of $\overline{\mathbf{A}(\vec{u})}$. It remains to construct a span program $\mathbf{B}$ such that $\mathbf{B}(\vec{u})$ behaves similarly as $\overline{\mathbf{A}(\vec{u})}$. Consider

$$\mathbf{B} = \begin{pmatrix} \vec{0} & \mathbf{I} \\ \vec{e} & \mathbf{A}^\top \end{pmatrix} \tag{1}$$

where the jth row is labelled by the negation of the label of jth row in $\mathbf{A}$, the remaining rows (corresponding to the rows of $\mathbf{A}^\top$) are labelled by 1. If the jth row in $\mathbf{A}$ is labelled by a literal which is assigned 0, the $j$the row of $\mathbf{B}$ is labelled by a literal which is assigned 1, and hence this basis vector can be used to cancel any number appearing in this column, which has the same effect as deleting this column in $\overline{\mathbf{A}(\vec{u})}$. Thus $\mathbf{B}(\vec{u})$ behaves equivalently to $\overline{\mathbf{A}(\vec{u})}$ and $\mathbf{B}$ computes the negation of the function computed by $\mathbf{A}$. The size of $\mathbf{B}$ is at most 1 larger than the number of rows of $\mathbf{A}$, which can be bounded by n times the size of $\mathbf{A}$.

Suppose that $f, g_1 \ldots g_k$ are computed by monotone span programs $\mathbf{A}, \mathbf{B}_1 \ldots \mathbf{B}_k$, and suppose that jth row of A is labelled by x_{i_j}. Then we claim that $f(g_1,\ldots,g_k)$ is computed by the span program

$$\begin{pmatrix} \mathbf{A} & \mathbf{I}_1 & \mathbf{I}_2 & \cdots \\ 0 & \mathbf{B}_{i_1} & 0 & \\ 0 & 0 & \mathbf{B}_{i_2} & \\ \vdots & & & \ddots \end{pmatrix}$$

where the rows corresponding to $\mathbf{A}$ are labelled all by 1 and the other rows are labelled as in the corresponding $\mathbf{B}_i$; $\mathbf{I}_j$ is the matrix which has a single 1 in the jth row and 1st column, all other entries are 0. If this span program accepts, it means that $(1,0,\ldots,0)$ is in a span of some rows of $\mathbf{A}$; moreover if jth row is used, the vector $(1,0,\ldots,0)$ from $\mathbf{I}_j$ must cancel with some other rows, which is possible only if $\mathbf{B}_{i_j}$ accepts. The other implication is easy as well.

To conclude that span programs of polynomial size are closed under NC^1-reducibility, note that in the circuits giving the reduction we can assume that NOT gates are either at the leaves or at the output of a gate for g; in that case we can substitute the span program for $\neg g$. □

As we shall see in Section 3.3, over finite fields $GF(p)$, dependency and span programs are equivalent, and hence have the same closure properties. For monotone dependency programs and dependency programs over general field we can only prove that they are closed under disjunction. We know that monotone dependency programs are not closed under conjunction, due to Theorem 3.1, where we proved

an exponential lower bound for a function which is a conjunction of polynomially many functions with constant size monotone dependency programs.

LEMMA 3.5. *For arbitrary field K and all boolean functions f and g, $g_1, \ldots, g_m$,*

- $mDP(f \vee g) \leq mDP(f) + mDP(g)$, *and*
- $DP(f \vee g) \leq DP(f) + DP(g)$.

PROOF. Suppose we have two dependency programs A and B. Then the dependency program

$$\begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix},$$

with the rows labelled as in A and B, computes their disjunction. This reduction preserves monotonicity. □

3.3. Dependency programs vs. span programs. In this section we compare the power of dependency and span programs. An easy reduction shows that span programs are at least as strong as dependency programs. In the monotone case, by Theorem 3.1 it follows that there is an exponential gap, as the function for which we proved an exponential lower bound for monotone dependency programs has linear-size span programs by Theorem 3.4. On the other hand, we prove that over finite fields $GF(p)$, non-monotone dependency programs are as strong as span programs.

LEMMA 3.6. *For an arbitrary field K and an arbitrary boolean function f, $mSP(f) \leq n \cdot (mDP(f))^2 + 1$ and $SP(f) \leq n \cdot (DP(f))^2 + 1$.*

PROOF. Let A be a dependency program, let k be the number of its rows. By Section 3.2 we may assume that $k \leq n \cdot DP(f)$. Then the function f is computed by the span program

$$\begin{pmatrix} \vec{e}_1 & \mathbf{A} & \mathbf{0} & \cdots & \mathbf{0} \\ \vec{e}_2 & \mathbf{0} & \mathbf{A} & & \mathbf{0} \\ \vdots & \vdots & & \ddots & \vdots \\ \vec{e}_k & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A} \end{pmatrix}$$

with all rows labelled as in A, where $\vec{e}_i$ denotes the ith vector of the standard basis. Monotonicity is preserved, and the size is as claimed. If an input is accepted by the dependency program, let i be an index of a row which appears in the non-trivial linear combination which sums to zero; then the same linear combination of the rows in the ith block is a non-zero multiple of the target vector. Conversely, if the target vector is a linear combination of the rows of the span program, take any block i containing a row with non-zero coefficient; the linear combination of the rows in this block gives a non-trivial linear dependency among the rows of A. □

THEOREM 3.7. *For any prime p and boolean function f, $DP_p(f) \leq (SP_p(f))^{O(1)}$.*

PROOF. In [13] it is observed that the size of a span program over $GF(p)$ for f is polynomially related to the size of a branching program counting modulo p computing f. (This equivalence is based on the results of [7] which proves that counting branching programs can perform rank computations, based on the results of [5, 16].)

Branching program is a directed acyclic graph with the edges labelled by literals or 1. A counting branching program accepts on a given assignment, if the number of accepting paths from the source to the sink is divisible by p; a path is accepting if all its edges are labelled by a true literal or 1.

We need to show that any function computed by a counting branching program can be computed by a dependency program with only polynomially larger size. We can represent a branching program by an adjacency matrix A with entries 1, x_i, and $1 - x_i$, according to the labelling of the edges. The matrix is upper triangular, assuming that the vertices are ordered topologically from source to sink. Let B be the adjacency matrix modified in the following way: put 1 in all diagonal entries. and remove the first column and the last row. In [7] it is shown that the number of accepting paths is equal to the determinant of B, up to a possible sign change. Hence it is sufficient to construct a dependency program that decides if the rows of B are linearly dependent (it will compute the negation of the function, but this does not matter since span programs are closed under negation).

Suppose that B has size $k \times k$. We construct a dependency program with $k(k+1)$ columns computing 1 iff the rows of B are dependent. Consider a row $(b_{j1}, \ldots, b_{jk})$; note that all the entries are literals, or constants 0 or 1. We replace this row by the matrix

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \vec{0}^\top & \vec{0}^\top & \cdots & \vec{0}^\top & \vec{1}^\top & \vec{0}^\top & \cdots & \vec{0}^\top \end{pmatrix}$$

where $\mathbf{I}$ is $k \times k$ identity matrix, and the vector $\vec{1}^\top$ is in the $(j+1)$st block of the matrix, i.e. in a unique position for each row of B. The last row is labelled by 1, for $j \leq k$ the jth row is labelled by c_j and the $(j+k)$th row is labelled by $1 - c_j$. Thus all rows are labelled by a literal or 0 or 1 (naturally, in the final stage we remove all rows labelled by constants, as in the proof that the dependency programs are closed under restrictions). Since the position of the vector $\vec{1}^\top$ is unique for each row of B, in any linear dependence the ones in these columns must cancel within the rows corresponding to the same row of B. It is easy to verify that the only vectors that can be obtained as a linear combination of the rows corresponding to the same rows of B and have zeroes in these positions necessarily have in the first block some multiple of the row of B $(b_{j1}, \ldots, b_{jk})$. It follows that the dependency program computes 1 iff the rows of B are linearly dependent (over the same field $GF(p)$). □

3.4. Polynomial programs. Polynomial programs can also be represented by a matrix, where each row is a vector of the coefficients of a polynomial at all the monomials of degree at most d. In this representation their size exactly corresponds to the size of span programs. It follows that each polynomial program of degree 1 is a span program, as in the degree 1 polynomial refutation only a constant polynomial can be multiplied, and this is clearly irrelevant.

LEMMA 3.8. *For an arbitrary field K and an arbitrary boolean function f, $mSP(f) = mPP_1(f)$ and $SP(f) = PP_1(f)$.*

We prove that polynomial programs over arbitrary finite field of any degree $d \geq 2$ are equivalent to boolean circuits, and hence are probably significantly stronger than span programs. Monotone polynomial programs of degree 2 simulate both monotone span programs and monotone boolean circuits. In [1] it has been shown that there are functions which can be computed by polynomial size monotone span

programs, but need at least $n^{\log n}$ size monotone circuits. No polynomial simulation of monotone circuits by monotone span programs is known and it is very unlikely that there is one. Thus monotone polynomial programs seem to be stronger than both previously considered models of monotone computations.

THEOREM 3.9. *Polynomial programs over an arbitrary finite field of an arbitrary constant degree $d \geq 2$ and boolean circuits mutually simulate each other. Monotone polynomial programs over any finite field simulate monotone circuits.*

PROOF. The simulation of polynomial programs by boolean circuits follows from the results of [9] which give a polynomial time procedure for deciding whether there exists a degree d refutation of given polynomials in polynomial calculus.

Let a circuit C be given; suppose C uses only $\vee, \wedge$ and literals. We construct a polynomial program of degree 2 which simulates C. Choose one variable for each vertex of the circuit. In particular we take some variable y_i for the input node x_i and z_i for the input node $\neg x_i$. Let g be a gate (i.e. non-input node) which is the conjunction, resp. disjunction, of the gates h and k; let u, v, w be the variables corresponding to g, h, k. Then we assign to g the polynomial $u + vw - v - w$, resp. $u - vw$. Let v_0 be the variable assigned to the output gate. The polynomial program consists of all polynomials assigned to the gates labelled by 1, the polynomial $1 - v_0$ labelled by 1, the polynomials y_i labelled by x_i, and the polynomials z_i labelled by $1 - x_i$.

It is easy to prove by induction that, for a given input, one can derive by a polynomial calculus derivation of degree 2 all polynomials v where v is a variable corresponding to a gate which computes 1 in the circuit. Hence, if the circuit outputs 1, v_0 is derivable and thus also 1 is derivable.

Now suppose that C outputs 0 for a given input. We shall show that there is a satisfying assignment to all equations chosen by the input, hence the constant polynomial 1 is not derivable. The assignment is given simply by putting $v = 0$, if the circuit computes 1 on the corresponding gate and $v = 1$ otherwise.

It is clear that the above simulation gives a monotone polynomial program, if the circuit is monotone. □

Note that for a general field we can always replace a polynomial program of degree d by a polynomial program of degree 2 with only a small increase in the size, by using more variables for the intermediate results.

4. Interpolation

4.1. Interpolation for Nullstellensatz and polynomial calculus. Clegg, Edmonds and Impagliazzo [9] showed that over a finite field it is possible to decide in polynomial time if a given set of polynomials has a polynomial refutation of a constant degree d. As a corollary they derive the following interpolation theorem.

THEOREM 4.1 ([9]). *Let a finite field be fixed. Suppose a set of polynomial equations of the form $p_i(\vec{x}, \vec{y}) = 0, q_j(\vec{x}, \vec{z}) = 0$ with $\vec{y}$ and $\vec{z}$ disjoint sets of variables, has a polynomial refutation of a constant degree. Then there exists a polynomial size circuit C with the following property. For a given input $\vec{u} \in \{0,1\}^n$, if C outputs 1 then $p_i(\vec{u}, \vec{y}) = 0$ are not satisfiable, and if C outputs 0 then $q_j(\vec{u}, \vec{z}) = 0$ are not satisfiable.*

PROOF. For a given input $\vec{u}$ test, using the polynomial time algorithm from [9] whether there exists a refutation of the given degree d of $p_i(\vec{u}, \vec{y}) = 0$; output 1 if there is such a refutation and 0 otherwise. If there is no such refutation then $q_j(\vec{u}, \vec{z}) = 0$ is not satisfiable, since otherwise we could get a refutation of $p_i(\vec{u}, \vec{y}) = 0$ from the refutation of $p_i(\vec{x}, \vec{y}) = 0, q_j(\vec{x}, \vec{z}) = 0$ by substituting $\vec{u}$. □

The same theorem obviously holds also for NS refutations, since it is a weaker system than polynomial refutations. We prove some refinements of this corollary which may be used for proving lower bounds on NS and possibly also polynomial refutations.

Let us call a polynomial $p(\vec{x}, \vec{y})$ *monotone* in $\vec{x}$ if it can be represented in the form $p'(\vec{x})p''(\vec{y})$ where $p'(\vec{x})$ is a monomial.

THEOREM 4.2. *Suppose a set of polynomial equations of the form $p_s(\vec{x}, \vec{y}) = 0, q_t(\vec{x}, \vec{z}) = 0$ has a NS refutation of a constant degree d. Then there exists a span program $\mathbf{C}$ with the property that if $\mathbf{C}$ accepts $\vec{u} \in \{0,1\}^n$ then $p_s(\vec{x}, \vec{y}) = 0$ are not satisfiable, and if C does not accept, then $q_t(\vec{x}, \vec{z}) = 0$ is not satisfiable. The size of $\mathbf{C}$ is polynomial in the number of variables $\vec{x}, \vec{y}$. Moreover, if all $p_s(\vec{x}, \vec{y})$ are monotone in $\vec{x}$, then $\mathbf{C}$ is monotone.*

PROOF. (I) First we consider a special case where all polynomials $p_s(\vec{x}, \vec{y})$ can be written in the form $x_i p'(\vec{y}) + p''(\vec{y})$ with x_i a single variable or just contain only variables $\vec{y}$. (In fact this is the usual form in applications, so the general case is considered here just for sake of having a nicer theorem.) We assume that the equations $u^2 - u = 0$ for all variables are present, so we have also polynomials of the form $x_i^2 - x_i = 0$. We can think of them as belonging to $q_t(\vec{x}, \vec{z}) = 0$.

Suppose that

$$x_i p'_{i,j}(\vec{y}) + p''_{i,j}(\vec{y}), \quad j = 1, \ldots, k_i,$$

are all the polynomials containing x_i.

The columns of the span program $\mathbf{C}$ will correspond to all monomials in $\vec{y}$ of degree $\leq d$, hence its size is polynomial in the number of variables, as d is constant.

The rows of $\mathbf{C}$ will be polynomials, or more precisely, the vectors of their coefficients, of the following three types. First, all polynomials of degree at most d which can be put in the form

$$r(\vec{y})(p'_{i,j}(\vec{y}) + p''_{i,j}(\vec{y})), \quad j = 1, \ldots, k_i;$$

labelled by x_i. Second, we include and all polynomials of degree $\leq d$ which can be put in the form

$$r(\vec{y})p''_{i,j}(\vec{y}), \quad j = 1, \ldots, k_i$$

labelled by $1 - x_i$. Third, we include all polynomials of degree at most d of the form

$$r(\vec{y})p(\vec{y})$$

where $p(\vec{y})$ is an initial polynomial which depends only on $\vec{y}$, and label them by 1.

It is easy to see that this span program tests whether for a given input $\vec{u}$, there is a NS refutation of $p_s(\vec{u}, \vec{y}) = 0$ of degree d.

If we have moreover the condition that the polynomials $p_s(\vec{x}, \vec{y}) = 0$ are monotone in variables $\vec{x}$, then the polynomials $p''_{i,k}(\vec{y})$ are 0. Hence all the polynomials labelled by $1 - x_i$ are 0, which means that the span program is monotone.

(II) Now we shall reduce a more general case to the case considered in (I). In the general case we modify the initial polynomials in the following way. For each variable x_i in it introduce a new variable w_i. Then replace all polynomials $p_s(\vec{x},\vec{y})$ by $p_s(\vec{w},\vec{y})$ and add the polynomials $x_i - w_i$ for all variables x_i. The new set of equation is satisfiable iff the original set of equations is, and the new polynomials have the required form. Also the original equations are derivable from the new ones.

Now consider the monotone case, i.e., suppose $p_s(\vec{x},\vec{y})$ has the form $x_{j_1}\dots x_{j_k}p(\vec{y})$. If $k \geq 2$, we replace this polynomial by k polynomials

$$x_{j_1}(p(\vec{y}) + w_2 + \ldots + w_k), \quad x_{j_2}w_2, \quad \ldots, \quad x_k w_k,$$

where $w_2, \ldots, w_k$ are new variables. It is easily seen that the old system of equations in variables $\vec{x}$, $\vec{y}$ is derivable from the new using a degree at most one larger than the degree of the original polynomials. It is also clear that the new set of equation is satisfiable iff the original set of equations is. □

The same considerations as above can be used to derive a version of Theorem 4.2 for polynomial refutations and polynomial programs. (The construction is even simpler, as we do not need to add all the multiples of the polynomials.) We obtain the following theorem. Note that for the finite fields the non-monotone part is equivalent to Theorem 4.1, because polynomial programs are equivalent to boolean circuits by Theorem 3.9.

THEOREM 4.3. *Suppose a set of polynomial equations of the form $p_s(\vec{x},\vec{y}) = 0, q_t(\vec{x},\vec{z}) = 0$ has a polynomial refutation of a constant degree d. Then there exists a polynomial program C with the property that if C accepts $\vec{u} \in \{0,1\}^n$ then $p_s(\vec{x},\vec{y}) = 0$ are not satisfiable, and if C does not accept, then $q_t(\vec{x},\vec{z}) = 0$ is not satisfiable. The size of C is polynomial in the number of variables $\vec{x},\vec{y}$. Moreover, if all $p_s(\vec{x},\vec{y})$ are monotone in $\vec{x}$, then C is monotone.*

4.2. A characterization of NS interpolants. Here we prove a converse to Theorem 4.2, which shows that we cannot reduce the class of interpolating functions any further.

THEOREM 4.4. *Let A be a span program of size m for inputs of size n. Then there are equations $p_s(\vec{x},\vec{y}) = 0, q_t(\vec{x},\vec{z}) = 0$ of degree 2 with $O(nm)$ variables which have a NS refutation proof of degree 3 and whose unique interpolant is the boolean function computed by A.*

PROOF. Let A be a matrix with m columns and M rows. By Section 3.2 we may assume that $M \leq nm$. Let l_j be the label of the jth row of A, i.e. x_i or $1-x_i$. Recall that by the construction in Theorem 3.4 equation (1) the negation of the function computed by A is computed by

$$\mathrm{B} = \begin{pmatrix} \vec{0} & \mathrm{I} \\ \vec{e} & \mathrm{A}^\top \end{pmatrix} \tag{2}$$

with the jth row labelled by $1 - l_j$, for $1 \leq j \leq m$, and the remaining rows labelled by one.

We shall write down equations $p_s(\vec{x},\vec{y}) = 0$ which describe A and $q_t(\vec{x},\vec{z}) = 0$ which describe B. This means that $p_s(\vec{u},\vec{y})$, resp. $q_t(\vec{u},\vec{z})$ will not be satisfiable iff A resp. B accepts the input $\vec{u}$. Since, for every $\vec{u}$, either A or B accepts $\vec{u}$, the equations together will not be satisfiable. We shall show that in fact they have

degree three NS refutation. It is also clear that the unique interpolant of such equations is the boolean function computed by A.

In order to describe A by polynomial equations we index the columns of A by the constant 1 and variables $y_2, \ldots, y_m$. The row vectors correspond to linear polynomials in $\vec{y}$, in particular the target vector corresponds to the constant polynomial 1. As the considered polynomials are linear, the polynomial 1 can be expressed from them iff it is a linear combination (with coefficients from the field) of them. So the system $p_s(\vec{x},\vec{y})$ contains polynomials

$$(a_{k,1} + \sum_{k=2}^{m} a_{k,j}y_j)l_k, \quad k = 1,\ldots,M \tag{3}$$

For B we index columns of B by 1, $z_1, \ldots, z_M$. The system $q_t(\vec{x},\vec{z})$ consists of polynomials

$$z_j(1 - l_j), \quad j = 1,\ldots,M, \tag{4}$$

$$1 + \sum_{k=1}^{M} a_{k,1}z_k, \tag{5}$$

$$\sum_{k=1}^{M} a_{k,j}z_k, \quad j = 2,\ldots,m. \tag{6}$$

Now we describe a NS refutation of degree 3. First multiply polynomials (3) by z_k, so that we get

$$(a_{k,1} + \sum_{j=2}^{m} a_{k,j}y_j)l_k z_k, \quad k = 1,\ldots,M \tag{7}$$

The equation $z_k(1 - l_k) = 0$ can be rewritten as $z_k = l_k z_k$. Hence by adding the appropriate multiples of polynomials (4) to the polynomials (7) we get

$$(a_{k,1} + \sum_{j=2}^{m} a_{k,j}y_j)z_k, \quad k = 1,\ldots,M \tag{8}$$

We sum all these equations to get

$$\sum_{k=1}^{M} a_{k,1}z_k + \sum_{j=2}^{m}\sum_{k=1}^{M} a_{k,j}y_j z_k. \tag{9}$$

Finally, we subtract the polynomials (5) and (6) multiplied respectively by 1, y_2, $\ldots$, y_m and get 1. □

4.3. An application. We shall show an application of Theorem 4.2 based on recent result [1]. In [1] a lower bound of $n^{\Omega(\log n/\log\log n)}$ is proved for the size of monotone span programs computing an explicitly defined boolean function. The form of the result allows us to deduce an $\Omega(\log n/\log\log n)$ lower bound on the degree of a NS refutation of an explicit set of polynomials of constant degree. This corollary is not interesting *per se*, since the system of equations is quite complicated and the lower bound is small compared to lower bounds from [8]. The importance of it is in showing that there is another context in which an interpolation theorem can be used to prove a lower bound. This gives us some hope to prove lower bounds in

this way for other systems, in particular for systems like polynomial calculus where we lack any lower bounds.

We shall start with the description of the result of [1]. Let $\Gamma \subseteq V_1 \times V_2$ be a bipartite graph, $|V_1| = |V_2| = n$, let $s \leq n$. We define two sets of subsets of vertices:

$$\mathcal{A} = \{X \subseteq V_1 \cup V_2 \mid \exists A \subseteq V_1,\ |A| = s,\ A \cup \Gamma(A) \subseteq X)\},$$

$$\mathcal{B} = \{X \subseteq V_1 \cup V_2 \mid \exists T \subseteq V_1,\ |T| \leq s,\ X \subseteq \bigcup_{B \subseteq V_1,\ |B|=s,\ B \cap T \neq \emptyset} (B \setminus T) \cup \Gamma(B)\}.$$

Here we denote by $\Gamma(C)$ the vertices which are connected by an edge to *all* vertices in C. While $\mathcal{A}$ is clearly an NP definition, it is not so obvious for $\mathcal{B}$. However, we can describe it as follows:

$$\begin{aligned}\mathcal{B} \;=\; & \{X \subseteq V_1 \cup V_2 \mid \exists T \subseteq V_1,\ |T| \leq s,\ X \cap T = \emptyset \text{ and} \\ & \forall j \in X \cap V_2\ \exists B_j \subseteq V_1,\ |B_j| = s,\ B_j \cap T \neq \emptyset,\ j \in \Gamma(B_j)\}.\end{aligned}$$

In [1] graphs Γ have been constructed such that for a suitable $s (s = \Theta(\log n / \log\log n))$ the sets $\mathcal{A}$ and $\mathcal{B}$ are disjoint and any monotone span program which accepts (the characteristic vectors of) sets of $\mathcal{A}$ and rejects sets of $\mathcal{B}$ has size $\Omega(\log n / \log\log n)$.

In order to be able to apply Theorem 4.2 we have to define these sets using polynomials of small degree and such that the polynomials defining $\mathcal{B}$ are monotone in variables $\vec{x}$ – the common variables of the two sets of polynomials which code the subsets X. The first thing can be done easily for any NP sets. By Cook's theorem we can express such predicates using 3-CNF's. Then each disjunction can be easily stated as an equation of degree ≤ 3. To prove the second condition first rewrite the definition of $\mathcal{B}$ using the predicate $j \in X$, instead of set operations and the inclusion relation. In the 3-CNF representation a variable x_j will represent the truth of $j \in X$. As the relation $j \in X$ occurs in the definition only negatively, so will the variables x_j in the 3-CNF. Thus the polynomials representing the disjunctions will be monotone in $\vec{x}$.

5. Different fields

Very little is known about the relative power of the algebraic models of computation if we change the underlying field. It is easy to see that a span or dependency program over a field with p^l elements can be written as a program over $GF(p)$ larger by a factor of l; this is implicit in [13]. Nothing is known if we change the characteristic of the field.

For characteristic 0, the natural question is whether a span program over reals can be converted into a span program over the rationals, and whether the length of numbers in a span program over rationals can be polynomially bounded. If both of these problems are resolved positively, it would show that functions computed by span programs over $\mathbf{R}$ can be computed by boolean circuits that are only polynomially larger.

The most natural approach is to replace the current coefficients of all vectors by (small) rational ones, so that all the linear dependencies are exactly preserved (for example by "moving" all the irrational points to rational ones carefully, so that the dependencies are preserved). In such a case the matroid represented by the matrix would not change. We show that this approach cannot work, as there exist matroids that are representable over $\mathbf{R}$ but not over $\mathbf{Q}$, and also matroids that are representable over $\mathbf{Q}$, but only with doubly exponential coefficients. This

result is based on the technique used to prove that the matroid of all vectors in the space K^3 has sufficient information to recover the whole field, see e.g. [23].

LEMMA 5.1. *Suppose that we are given real numbers $x_1 = 1, x_2, \ldots, x_n$ and polynomials $p_1, \ldots, p_m$ of n unknowns with integral coefficients such that the equations $p_j(\vec{x}) = 0$ are all satisfied. Then there exists a matroid M of elements $u_1, \ldots, u_n, \ldots, u_N$ such that for any representation of M over $\mathbf{R}$ the numbers $\tilde{x}_i = \alpha_i/\alpha_1$, where $\alpha_i = (u_i^\top u_{N-1})/(u_i^\top u_N)$, satisfy the equations $\tilde{x}_1 = 1$ and $p_j(\tilde{x}_1, \ldots, \tilde{x}_n) = 0$ for all $j \leq m$.*

PROOF. The idea is to use the usual geometric construction of sums and products in $\mathbf{R}^2$, namely the facts that $x_i + x_j = x_k$ iff the lines $(1,0),(0,x_i)$ and $(1,x_j),(0,x_k)$ are parallel, and $x_i x_j = x_k$ iff the lines $(1,0),(0,x_i)$ and $(x_j,0),(0,x_k)$ are parallel. To be able to speak about parallel lines in terms of linear dependencies, we use the projective plane represented by the vectors in $\mathbf{R}^3$. The point (a,b) is represented by $(a,b,1)$, up to scalar multiplication, and the lines are parallel if they both contain the same vector $(c,d,0)$ (corresponding to their point in the infinity). If we take the matroid of all the auxiliary points, any representation has to preserve the identities.

Due to the integrality of the coefficients we may assume that all the equations have form either $x_i + x_j = x_k$ or $x_i x_j = x_k$ (we may have to add some numbers representing the intermediate values; also we have to use the fact that $x_1 = 1$). Now consider the matroid M in $\mathbf{R}^3$ consisting of all the vectors $u_i = (0, x_i, 1)$, $v_i = (x_i, 0, 1)$, $w_i = (1, x_i, 1)$, $z_i = (-1, x_i, 0)$, and the three basis vectors. We claim that the matroid M satisfies the condition in the lemma.

Suppose that the matroid M is represented in some vector space over the reals. The basis vectors have to be represented by three linearly independent vectors, and all other vectors have to be represented by vectors linearly dependent on them. Hence we may assume that the vector space is $\mathbf{R}^3$, moreover, we may change the basis so that the basis vectors are represented by basis vectors. We may assume that any vector with non-zero last coordinate is represented with 1 in that coordinate: Multiplying a vector by a non-zero scalar does not change the linear dependencies; moreover, the values of α_i and hence $\tilde{x}_i$ are not affected by this change, since the last two vectors are already fixed to be the last two basis vectors.

After these transformations, the vectors u_i are represented by $(0, \alpha_i, 1)$. In M, all the triples u_i, v_i, z_1 are linearly dependent. It follows that there exists a non-zero scalar β such that all vectors v_i are represented by $(\beta\alpha_i, 0, 1)$. Also, since both triples $(1,0,0), u_i, w_i$ and $(0,1,0), v_1, w_i$ are linearly dependent, w_i must be represented by $(\beta\alpha_1, \alpha_i, 1)$.

Now we prove that the numbers $\tilde{x}_i$ satisfy all the equations. First suppose that the equation $x_i + x_j = x_k$ is present. Then both triples v_1, u_i, z_i and w_j, u_k, z_i are linearly dependent. It follows that $\alpha_i = \alpha_k - \alpha_j$ and hence $\tilde{x}_i + \tilde{x}_j = \tilde{x}_k$. Now suppose that the equation $x_i x_j = x_k$ is present. Then the triples v_1, u_i, z_i and v_j, u_k, z_i are both linearly dependent. It follows that $\alpha_i/\alpha_1 = \alpha_k/\alpha_j$ and hence $\tilde{x}_i \tilde{x}_j = \tilde{x}_k$. □

THEOREM 5.2. *There exists a matroid represented over $\mathbf{R}$ but not representable over $\mathbf{Q}$.*

PROOF. Use Lemma 5.1 for $x_2 = \sqrt{2}$ and the equation $x_2^2 = 2x_1$. It follows that if the matroid is represented over $\mathbf{Q}$, the equation has a solution in $\mathbf{Q}$, which

is a contradiction. (We use the fact that any vectors with rational coefficients are linearly dependent over $\mathbf{R}$ are also linearly dependent over $\mathbf{Q}$.) □

THEOREM 5.3. *There exists a matroid of $O(n)$ points which can be represented over $\mathbf{Q}$ only so that the ratio of some coordinates is at least 2^{2^n}.*

PROOF. Use Lemma 5.1 for $x_{i+1} = 2^{2^i}$ and equations equations $x_2 = 2x_1$, $x_{i+1} = x_i^2$ for $2 \leq i < n$. □

6. Open problems

Lower bounds for monotone polynomial programs.
The main open problem suggested by this paper is to prove lower bounds for monotone polynomial programs over finite fields. If such a bound is proved for a suitable function, a lower bound for polynomial calculus would follow by the interpolation theorem (Theorem 4.3).

Lower bounds for non-monotone span programs.
Proving a superpolynomial lower bound on non-monotone span programs would provide a lower bound on branching programs, and hence it would be a major achievement. Our results show that it could be beneficial to consider dependency programs instead of span programs, as their structure is simpler.

Better lower bounds for monotone span programs.
The best lower bound for monotone span programs is $n^{\Omega(\log n/\log\log n)}$ [1]; no exponential bound is known. Furthermore, even though we have this superpolynomial lower bound for monotone span programs, it would be interesting to have such a bound for a function computable by non-monotone span programs of polynomial size, to separate these two models.

Separation of programs over different fields.
All lower bounds for the monotone dependency or span programs known to us work for an arbitrary field. It would be interesting to have some technique which would distinguish the fields, similarly as in the results for bounded depth circuits with MOD_m gates [22]. Thus we ask to prove separation between $mDP_p(f)$ and $mDP_q(f)$ (or $mSP_p(f)$ and $mSP_q(f)$) for some explicit function f.

Power of span programs over the reals.
It is still open whether functions computed by polynomial size span programs over $\mathbf{R}$, or even $\mathbf{Q}$, can be computed by polynomial size circuits. The problem is that the span program can contain huge constants, whose size is not included in the size of the span program. (If, e.g., the span programs are uniform, the size of constants is bounded and the function is in P.)

Acknowledgements. We are grateful to Sam Buss for useful discussions and to the anonymous referee for the comments. The last open problem was communicated to us by Eric Allender and Avi Wigderson.

References

[1] L. Babai, A. Gál, and A. Wigderson. Superpolynomial lower bounds for monotone span programs. DIMACS technical report 96-37, submitted to Combinatorica, 1996.
[2] L. Babai, A. Gál, J. Kollár, L. Rónyai, T. Szabó, and A. Wigderson. Extremal bipartite graphs and superpolynomial lower bounds for monotone span programs. In *Proc. of the 28th Ann. ACM Symp. on Theory of Computing*, pages 603–611. ACM, 1996.

[3] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, and P. Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. *Proc. London Math. Soc.*, 73:1–26, 1996.
[4] A. Beimel, A. Gál, and M. Paterson. Lower bounds for monotone span programs. In *Proc. of the 36th Ann. IEEE Symp. on Foundations of Computer Sci.*, pages 674–681. IEEE, 1995.
[5] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inf. Process. Lett.*, 18:147–150, 1984.
[6] M. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proofs with small coefficients. In *Proc. of the 27th Ann. ACM Symp. on Theory of Computing*, pages 575–584. ACM, 1995.
[7] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of Logspace-MOD-classes. *Mathematical Systems Theory*, 25:223–237, 1992.
[8] S. Buss, R. Impagliazzo, J. Krajíček, P. Pudlák, A. A. Razborov, and J. Sgall. On constant-depth Frege systems with a modular counting connective. To appear in Computational Complexity.
[9] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proc. of the 28th Ann. ACM Symp. on Theory of Computing*, pages 174–183. ACM, 1996.
[10] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.
[11] S. A. Cook and A. Haken. An exponential lower bound for the size of monotone real circuits. Manuscript, 1995.
[12] A. Haken. The intractability of resolution. *Theoretical Comput. Sci.*, 39:297–308, 1985.
[13] M. Karchmer and A. Wigderson. On span programs. In *Proc. of the 8th Structure in Complexity Theory*, pages 102–111. IEEE, 1993.
[14] J. Krajíček. Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. To appear in J. of Symbolic Logic.
[15] J. Krajíček, P. Pudlák, and A. Woods. Exponential lower bount to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures and Algorithms*, 7:15–39, 1995.
[16] K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. In *Proc. of the 18th Ann. ACM Symp. on Theory of Computing*, pages 338–339. ACM, 1986.
[17] T. Pitassi, P. Beame, and R. Impagliazzo. Exponential lower bounds for the pigenhole principle. *Comput. Complexity*, 3:97–140, 1993.
[18] P. Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. To appear in J. of Symbolic Logic.
[19] P. Pudlák and V. Rödl. A combinatorial approach to complexity. *Combinatorica*, 12(2):221–226, 1992.
[20] A. A. Razborov. Applications of matrix methods for the theory of lower bounds in computational complexity. *Combinatorica*, 10:91–93, 1990.
[21] A. A. Razborov. Lower bounds for the polynomial calculus. Manuscript, 1996.
[22] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. of the 19th Ann. ACM Symp. on Theory of Computing*, pages 77–82. ACM, 1987.
[23] D. J. A. Welsh. *Matroid Theory*. Academic Press, 1976.

MATHEMATICAL INSTITUTE. AV ČR, ŽITNÁ 25, 115 67 PRAHA 1, CZECH REPUBLIC
E-mail address: pudlak@math.cas.cz

MATHEMATICAL INSTITUTE, AV ČR, ŽITNÁ 25, 115 67 PRAHA 1. CZECH REPUBLIC, AND DEPT. OF APPLIED MATHEMATICS, CHARLES UNIVERSITY, MALOSTRANSKÉ NÁM. 25, 118 00 PRAHA 1, CZECH REPUBLIC
E-mail address: sgall@math.cas.cz

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 39, 1998

Self-Reflection Principles and NP-Hardness

Dan E. Willard

Abstract

We will study several weak axiom systems that are capable of proving their own consistency. Our two goals will be to outline the general capacities of such self-justifying systems, and to explain how some of their properties are closely related to several open question about NP.

1. INTRODUCTION: Define an axiom system α to be *Self-Justifying* iff

i) one of the theorems implied by α is the statement of its own consistency (by at least some reasonable definition of self-consistency), and

ii) the axiom system α is in fact consistent.

Rogers [26] has noted that Kleene's Fixed Point Theorem implies that every r.e. axiom system α can be easily extended into a broader system α^* which satisfies condition (i). Kleene's proposal was essentially for the system α^* to contain all α's axioms plus the one additional axiom sentence:

> There is no proof of 0=1 from the union of α with this sentence.

It is well known [14, 26, 13]) that the Fixed Point Theorem makes it easy to encode the indented sentence above for essentially any r.e. axiom system α. However, the catch is that α^* can be inconsistent even while its added axiom justifies α^* 's consistency. Thus α^* will typically violate Part-ii of the definition of Self-Justification. This problem arises not only in Gödel's paradigm, where α had at least the power of Peano Arithmetic (PA), but also for many systems much weaker than PA. For instance, no system satisfying the Hilbert-Bernays properties can be self-verifying [10, 20]. Many other types of generalizations of Gödel's Second Incompleteness Theorem are also known [5, 7, 8, 24, 27, 28].

This paper was supported in part by NSF Grant CCR 93-02920
1991 Mathematics Subject Classifications 68QXX, 03FXX and 03BXX

Some notation is needed to define our goals. Let $x - y$ be defined to equal zero when $x < y$, and $\lfloor \frac{x}{y} \rfloor$ equal x when $y = 0$. Subtraction and Division, thus defined, are total functions. All the Π_1 sentences of Arithmetic can be trivially rewritten with Integer Subtraction and Division so replacing Addition and Multiplication. The advantage of this approach is that it allows an axiom system to avoid the assumption that Addition and Multiplication are total functions. We will see how this change in perspective shall enable us to avoid many of the limitations imposed by Gödel's Incompleteness Theorem.

Throughout this paper, $\lceil \Phi \rceil$ will denote Φ's Gödel number, and $\mathrm{Prf}_\alpha(x, y)$ will be a Δ_0 formula indicating y is the Gödel number of a proof from axiom system α of the sentence with Gödel number x. An axiom system α's *Canonical Reflection Principle* for the sentence Ψ is the sentence:

$$\{ \quad \exists\, y \quad \mathrm{Prf}_\alpha (\lceil \Psi \rceil , y) \quad \} \quad \supset \quad \Psi \tag{1}$$

It is well known that Löb's Theorem [19] asserts that *every consistent extension* α of Peano Arithmetic is capable of proving the validity of (1) in only the uninteresting degenerate case where α can prove Ψ by itself! One of our goals will be to investigate what axiom systems, not recognizing Addition and Multiplication as total functions, can actually evade Löb's Theorem. Another goal will be to investigate how this question is related to the P =? NP Open Problem.

Our objectives can perhaps be more easily understood when it is appreciated that the following three generalizations of Gödel's Incompleteness Theorem severely limit the capacity of any self-justifying axiom systems:

A. Pudlák [24] has shown that no extension of Robinson's System Q can verify its own Hilbert consistency. That is, there can exist no consistent axiom system α which recognizes Addition and Multiplication as total functions and which can prove the non-existence of a Hilbert-style proof of 0=1 from itself.

B. Robert Solovay (private communications [28]) has generalized Pudlák's theorem to systems α which merely recognize Successor as a total function and use Subtraction and Division to implicitly represent the properties of Addition and Multiplication. If such an α is consistent and can prove all Peano Arithmetic's Π_1 theorems about Subtraction and Division, then Solovay's theorem asserts α can not prove the non-existence of a Hilbert-style proof of 0=1 from itself.

C. A variation of Löb's Theorem is also valid for systems α which recognize none of Addition, Multiplication or even Successor as total functions. Our Theorem 6.2 will show that if such an α is consistent and can prove all Peano Arithmetic's Π_1 theorems about Subtraction and Division then it can not prove for every Π_1 sentence Ψ, the associated reflection principle (1).

The perhaps most surprising aspect of this article is that we will develop several examples of interesting self-justifying systems, despite the limitations above. These systems will contain sufficient robustness to offer at least a reasonable approximation of the reflection principle (1), while also recognizing the validity of Peano Arithmetic's Π_1 theorems about the properties of Subtraction and Division. These

examples are interesting partly because they will help clarify the nature and limitations of Gödel's Incompleteness Theorem. Our discussion will also show how several open questions about NP are related to the properties of self-justifying systems.

At first, we were tempted to focus this article almost exclusively on the latter topic because it is clearly more closely related to the chief interests of the DIMACS Symposium. However upon reflection, we decided to touch on both subjects, at least briefly in this paper, because they are so closely interwoven with each other.

OTHER RELEVANT LITERATURE: Kriesel and Takeuti define in [17, 18, 30] a different type of self-justifying axiom system, based on a Second-Order Logic formalism that is a generalization of Gentzen's Cut-Free Sequent Calculus. We recommend [17, 18, 30] highly to the reader because their mathematical treatment is quite illuminating. However, since the Kriesel-Takeuti formalism rests on a Second-Order Cut-Free formalism, whereas we use a First Order formalism permitting Gentzen-style cuts, it is clear that the two approaches are traveling mostly in orthogonal directions.

The literature on Definable Cuts casts some interesting perspectives on our investigations. A definable cut is roughly defined to be a formally specified set of numbers, closed under the operations of Successor and Less-Than, which is not formally known by a particular axiom system α to actually include the full set of all integers. Although axiom systems such as Robinson's Q can not prove their general self-consistency, Nelson did show that they could prove their cut-free or Herbrand consistency local to definable cuts [21]. Pudlák, [12, 24] later generalized Nelson's theorem for any finite extension of Q satisfying [24]'s Sequential condition. Pudlák, also showed that this result would not remain valid if the deductive method was changed to either a Hilbert proof system (or equivalently to Gentzen's Sequent Calculus with cuts). Other interesting research into definable cuts certainly includes the work of Dimitracopoulous, Hájek, Krájicek, Paris, Solovay and Wilkie [11, 12, 15, 23, 28, 32]. The (A) and (B) versions of the Incompleteness Theorem, discussed earlier in this section, were derived by Pudlák and Solovay through the employment of Definable Cuts. Wilkie and Paris show how growth functions, such as Ω_1, can be used to prove various equi-consistency theorems. In general, research by Friedman, Paris, Parikh, Pudlák and Wilkie [9, 22, 25, 32] illustrates various respects in which the size of large numbers and the presence of fast growing functions affects a system's ability to recognize its own self-consistency.

2 Tangibility Reflection Principles

Let Φ denote a prenex normal sentence. Throughout this paper, Φ^i_j will denote a sentence identical to Φ except that each unbounded universally quantified variable in Φ is bounded by i and each unbounded existentially quantified variable in Φ is bounded by j. Also Φ^i will be an abbreviation for Φ^i_∞.

Let $\mathrm{Tangible}(x)$ be a formula which asserts that the number x is not of unusually large size, i.e. its size is very "tangible". Since our axiom systems α will not necessarily assume that any of Multiplication, Addition or even Successor are total functions, three possible definitions of tangibility are given below. In the second and third definitions, it is assumed that $k \geq 2$.

1. $\text{TangPred}(x) \quad = \quad \{ \ \exists z \ \ x = z - 1 \ \}$
2. $\text{TangDiv}_k(x) \quad = \quad \{ \ \exists z \ \ x < z/k \ \}$
3. $\text{TangRoot}_k(x) \quad = \quad \{ \ \exists z \ \ x < z^{1/k} \ \}$

For any of the three preceding definitions of tangibility, define an axiom system α's *Tangibility Reflection Principle* for the sentence Ψ to be the assertion:

$$\forall x \ \ \{ \ \ [\ \ \exists \ y \ \ \text{Prf}_\alpha (\ulcorner \Psi \urcorner , \ y \ ) \wedge \text{Tangible}(x) \ \ ] \ \ \supset \Psi^x \ \ \} \qquad (2)$$

Given an arbitrary consistent axiom system A (which is say an extension of Peano Arithmetic), we will show that there exists a self-justifying axiom systems α which supports each of the three preceding tangibilty reflection principles and can also validate all the Π_1 theorems that A proves (when these theorems are rewritten in a form where the Subtraction and Division primitives replace Addition and Multiplication).

One reason the preceding result will be interesting is that it will provide a possible partial answer to the paradoxical question below:

* How do Human Beings manage to muster the physical energy and psychological desire to think (and prove theorems) when the various different types of generalizations of Gödel's Incompleteness Theorem assert that no reasonable conventional axiom system can confidently assume its own consistency?

We can provide no perfect answer to the preceding question because there can obviously never exist any completely satisfactory answer to a logical paradox. However, our partial answer to the logical paradox $*$, raised by Gödel's Incompleteness Theorem, will be that a Thinking Being can indeed assume that if he proves Ψ, then Ψ will be valid when it is restricted to all numbers of at least reasonable size (i.e. the "tangible" numbers specified in Eq. (2)).

Part of what will make the Reflection Principle (2) interesting is that Variation-C of Gödel's Incompleteness (from the preceding section) states that Eq. (1)'s slightly more general canonical reflection principle always leads to an inconsistent axiom system. Thus, the compromise reflection principle (2) appears to be close to the best we can hope for. A second reason for interest in the Tangibility Reflection Principle is its relationship to the P =? NP Open Question (discussed in the second half of this article).

Finally, we wish to close this section by certainly acknowledging that there clearly will be major disadvantages to our self-justifying axiom systems α, since the presence of the Tangibility Reflection Principles (2) will force all our axiom systems to drop the assumption that Multiplication is a total function (Many of our axiom systems α will also exclude the axioms that Addition and Successor are total functions). However, paradoxes never have perfect solutions. The partial virtues of the ISREF axiom systems will be that they will offer at least a partial answer to the Paradoxical Question $*$, as well as provide at least some new perspectives into the P =? NP Open Question.

3 Definition of ISREF(A) Axiom System

In order to define the several axiom systems we shall study, it is desirable to first define a 4-tape universal Turing machine $\mathcal{M}$ that they shall simulate. The machine $\mathcal{M}$ will use "left-sided tapes", i.e. tapes that can be traversed infinitely far to the left, but not to the right of the address Zero.

The alphabet Σ stored on the Turing tapes will consist of three letters, denoted as "0", "1" and "2". The first two letters will describe the bit values of "0" and "1", and they will be called the *non-null letters*. The letter "2" will be a null-value symbol. It will be assumed that only a finite number of letters on any tape are non-null symbols. The rightmost letter on the tapes will always be a null marker (to indicate that the tape head can not move any further right). The remaining null values on the tape will appear as an infinite sequence of consecutive null markers on the tape's left side. The symbol " * " will denote this infinite sequence of null values. Thus, if $b_n, b_{n-1}, ...b_1$ represents a sequence of non-null bits then $*, b_n, b_{n-1}, ...b_1, 2$ indicates how this sequence would be stored on a left-sided Turing tape.

Define the sequence $b_n, b_{n-1}, ...b_1$ to be a *Normalized Representation* of the integer J iff its length $n = \text{Max}(1, \lceil 1+\text{Log}_2 J \rceil)$ and $J = \sum_{i=1}^{n} b_i \cdot 2^{i-1}$. The advantage of the preceding definition is that each natural number J can be mapped onto an unique normalized representation encoding it as a bit string. For simplicity, we assume throughout this article that all four tapes of the $\mathcal{M}$ always contain such normalized bit strings.

One of our four Turing tapes, denoted as $T_{\mathcal{E}}$, will store an "encoding" of the machine $\mathcal{M}$'s instruction set, and the other three tapes, T_1, T_2 and T_3, will designate three input tapes, containing the data that should be processed. The "output" of our Turing Machine $\mathcal{M}$ will be assumed to be stored on the tape T_1 at the time $\mathcal{M}$ halts. It will also be assumed that $\mathcal{M}$ is a *"Universal"* 4-tape machine, in that it satisfies the following two quite natural criteria:

1. Given any 3-input computable partial function $f(x_1, x_2, x_3)$, it will be possible to simulate f by loading some instruction encoding e_f on the tape $T_{\mathcal{E}}$ and loading the triple (x_1, x_2, x_3) on M's three input tapes.
2. If the above partial function f has a running time $g(x_1, x_2, x_3)$ on a random access machine $\mathcal{R}$ then e_f's running time on $\mathcal{M}$ will be no worse than $k \cdot g(x_1, x_2, x_3)^k$, for some fixed constant k whose value depends only on $\mathcal{M}$.

Our ISREF axiom systems will contain three sets of functions for simulating the actions of the Turing Machine $\mathcal{M}$ above. Let Z denote an arbitrary 4-tuple of integers, which represent the transcribed numbers stored on $\mathcal{M}$'s four tapes at the time execution commences. Then ISREF's Turing simulation functions will include

a. $Tape_i(Z, x, t)$ designating a number 0, 1, or 2, according to which letter is stored on $\mathcal{M}$'s i-th tape in position x at time t when the 4-tuple Z designates the initial configuration on $\mathcal{M}$'s tapes.

b. $Head_i(Z, t)$ indicating the address of the i-th tape's head at time t when the 4-tuple Z designates the initial configuration on $\mathcal{M}$'s tapes.

c. $State_i(Z,t) = 1$ if the machine $\mathcal{M}$ is in state i at time t when the 4-tuple Z designates the initial configuration on $\mathcal{M}$'s tapes. $State_i(Z,t) = 0$ otherwise.

Define $F(a_1, a_2, ...a_j)$ to be a *Non-Growth Function* iff for all values of $a_1, a_2, ...a_j$, the function F satisfies the inequality $F(a_1, a_2, ...a_j) \leq Maximum(2, a_1, a_2, ...a_j)$. Unlike the ISTR and IS$^\lambda$ axiom formalisms discussed later in this paper, only non-growth functions will be allowed in the language of our ISREF axiom system. They will include the *Tape*, *Head* and *State* Turing functions (defined above) and Integer Subtraction and Division (defined in Section 1). Other non-growth functions will include: $Maximum(x,y)$, $Logarithm(x) = \lceil Log_2(x+1) \rceil$, $Predecessor(x) = \text{Max}(x-1, 0)$, $Root(x,y) = \lceil x^{1/y} \rceil$, $Count(x,j)$ designating the number of "1" bits among x's rightmost j bits, and $Bit(x,y) = Count(x,y) - Count(x,y-1)$ (i.e. $Bit(x,y)$ designates the value of x's y-th bit). The eight non-Turing functions defined in this paragraph are called the *"Grounding Functions"* in several of our papers.

We will follow mostly conventional logic notation when discussing the ISREF axiom systems. Thus a *term* is defined to be a constant symbol, a variable symbol or a function symbol (whose input arguments are recursively defined terms). If t is a term then the quantifiers in the wffs $\forall\, v \leq t\ \ \Psi(v)$ and $\exists\, v \leq t\ \ \Psi(v)$ will be called *bounded quantifiers*. These two wffs will be semantically equivalent to the formulae $\forall v\ \ (\ v \leq t\ \supset\ \Psi(v))$ and $\exists v\ \ (\ v \leq t\ \wedge\ \Psi(v))$. A formula Φ will be called Δ_0^+ iff all its quantifiers are bounded. A formula Υ will be called Π_1^+ iff it is written in the form $\forall v_1\ \forall v_2\ ...\ \forall v_n\ \ \Phi$, where Φ is Δ_0^+.

Our definitions of Δ_0^+ and Π_1^+ formulae (above) are the same as the conventional definitions of Δ_0 and Π_1 formulae except that the Δ_0^+ and Π_1^+ formulae use different atomic base functions than the Addition and Multiplication primitives employed by Arithmetic's conventional Δ_0 and Π_1 formulae. Since the Subtraction and Division primitives can represent Addition and Multiplication as relations, it follows that every conventional Π_1 formula can be translated into an equivalent Π_1^+ formula.

Note only non-growth function symbols are used in our Δ_0^+ and Π_1^+ formulae (and functional growth can thus be represented in these formulae only through the presence of unbounded existential quantifiers). We follow this notation convention because the ISREF(A) axiom system (defined in the next paragraph) recognizes the existence of only non-growth functions.

The acronymn "IS" stands for "Introspective Semantics", and "ISREF" is an abbreviation for "Introspective Semantics with Reflection". There will be two variants of our *ISREF* axiom systems studied in this paper, called ISREF(A) and ISREF^R(A). We will discuss only ISREF(A) in the next two sections because it is the easier of the two variants to analyze. The **"ISREF MAPPING"** will be a transformation that maps an initial axiom system A onto an axiom system ISREF(A), such that the latter can simultaneously recognize its own consistency and prove all A's Π_1^+ theorems. Formally, the system ISREF(A) is defined as consisting of the following four groups of axioms:

Group-Zero: The axiom system ISREF(A) will contain one constant symbol $\bar{n}$ for each natural number $n \geq 0$. The Group-Zero axioms will define these constants formally. Its axioms include $\bar{1} \neq \bar{0}$, Predecessor($\bar{0}$) $= \bar{0}$, and for each $n > 0$, the axioms: Predecessor($\bar{n}$) $= \overline{n-1}$, $\overline{2n} - \bar{n} = \bar{n}$ and $\overline{2n+1} - \bar{n} - \bar{1} = \bar{n}$. (All our theorems about ISREF(A) would remain valid if the latter two types of Group-zero axioms were omitted from ISREF(A)'s definition. The footnote[1] explains why the redunduncy produced by the axioms schemes $\overline{2n} - \bar{n} = \bar{n}$ and $\overline{2n+1} - \bar{n} - \bar{1} = \bar{n}$ is desirable and interesting.)

Group-1: All the axioms in Group-1 will be Π_1^+ sentences. They will assign the " =" and "<" predicates their usual logical properties. Also, Group-1 will contain a finite set of axioms defining *ISREF*'s atomic functions, so that for each function F and set of constants $\bar{k}$, $\bar{c_1}$, $\bar{c_2}$, ...$\bar{c_m}$, the combination of the Group-Zero and Group-1 axioms will imply $F(\bar{c_1}, \bar{c_2}, ...\bar{c_m}) = \bar{k}$ when this sentence is true. It is easy *(actually trivial)* to construct a set of Π_1^+ sentences that meet the preceding conditions (for example, see the footnote below[2].) Any valid formulation can be used to define the Group-1 axioms.

Group-2: For each Π_1^+ sentence Φ, Group-2 will contain a corresponding axiom of the form:

$$\forall\, y\ \ \{\ \text{Prf}_A(\ \lceil \Phi \rceil\ ,\ y\ )\ \supset \Phi\ \} \tag{3}$$

Group-3: Define *Size(y)* to be function where Size(y) $= c$ when y is the Gödel number of a proof whose largest stored constant is c, and where Size(y) $= 0$ otherwise. For each prenex normal sentence Ψ, ISREF(A) will contain a corresponding "size reflection" axiom, whose formal encoding appears in the Appendix, and which can be intuitively be thought of as being equivalent to the prototype sentence (4) below.

$$\forall x\ \forall y\ \ \{\ \ \text{Prf}_{\text{ISREF}(A)}(\ \lceil \Psi \rceil\ ,\ y\ )\ \wedge\ x > \text{Size}(y)\ \ \supset\ \ \Psi_{x-1}^{x-1}\ \} \tag{4}$$

FIRST CLARIFYING COMMENT: It is important to recall that the axiom system ISREF(A) recognizes no growth functions. Otherwise, a sentence, where Ψ contains a subscript similar to (4), could certainly not be universally valid.

SECOND CLARIFYING COMMENT: Because the axiom (4) refers to an axiom system which includes itself, its formal encoding (in the Appendix)

[1] One function of the axiom schemes $\overline{2n} - \bar{n} = \bar{n}$ and $\overline{2n+1} - \bar{n} - \bar{1} = \bar{n}$ is that they enable us to compress proofs using ISREF(A). This is because the constant $\overline{m}$ would require exponentially more bits to unambiguously define it if one had to rely exclusively on the axiom scheme Predecessor($\bar{n}$) $= \overline{n-1}$ to define $\overline{m}$. (In particular, one would need m rather than Log m axioms to define $\overline{m}$ if only the Predecessor axioms were present.) It is noteworthy that the ISREF(A) axiom system's consistency will be proven by Theorem 4.1 to be unaffected by the presence of the two further axioms $\overline{2n} - \bar{n} = \bar{n}$ and $\overline{2n+1} - \bar{n} - \bar{1} = \bar{n}$. This fact is interesting because the same will not be true for Section 10's ISTM(A) system.

[2] The Π_1^+ definition of the "Grounding Functions" is quite conventional and simple, since these primitives are quite ordinary non-growth functions. On the other hand, the definition of the Turing functions is somewhat novel. It will have three parts. Its first fragment will trivially describe the initial values of the Head and State functions at time zero. The second fragment will employ the Bit and Logarithm functions to assure that the bit sequence stored on the tape $T_\mathcal{E}$, T_1, T_2, and T_3, at time zero corresponds to the four integers associated with the 4-tuple Z. The third fragment will inductively define the values of the Head, State, and Tape functions at time t in terms of the prior values at time $t-1$. More details about the exact formulation of the Group-1 axioms are unimportant because any formulation of these axioms as Π_1^+ sentences is equally suitable for our purposes.

involves a diagonalization construction, employing essentially either Kleene's or Gödel's Fixed Point Theorem. Our recommendation is that the reader postpone reading the Appendix until after the other parts of this article are finished because the implication of Equation (4)'s *"self-consistency statement"* are much more interesting than the tedious details of its formal encoding.

THIRD CLARIFYING COMMENT: Some readers may initially feel disturbed by the fact that ISREF(A) is able to recognize its self-consistency by simply making the Group-3 sentences, asserting its consistency, into merely formal axioms. It would be natural for many readers to thereby wonder: *"How is ISREF(A) nontrivial?"* The answer was provided in the first paragraph of Section 1, where it was noted that every r.e. axiom system can be easily extended to satisfy Part (i) of the definition of self-justification, but most such extended axiom systems fail Part (ii) of this definition. That is, most axiom systems become inconsistent when they are augmented to include an assertion simply declaring that *"I am consistent"*. However, ISREF(A) will be different. Theorem 4.1 will prove that *if A is consistent then ISREF(A) is automatically consistent.* Thus, if A = Peano Arithmetic, our construction will assure that ISREF(A) can prove all the Π_1 theorems of Peano Arithmetic, and that its consistency will be as certain as that of Peano Arithmetic!

FOURTH CLARIFYING COMMENT: Note that if Ψ designates the sentence " 0=1 " then for every j the equivalence $\Psi_j^j \equiv \{\ 0=1\ \}$ trivially holds (because the sentence " 0=1 " contains no quantified variables). The axiom scheme (4), for this particular Ψ, thus easily implies:

$$\forall y \quad \{\quad \mathrm{Prf}_{\mathrm{ISREF}(A)}\ (\ulcorner 0=1 \urcorner\ ,\ y\ ) \quad \supset \quad 0=1 \quad \} \tag{5}$$

Since ISREF(A) can prove $0 \neq 1$, it can deduce (6) from (5).

$$\forall y \quad \neg\ \mathrm{Prf}_{\mathrm{ISREF}(A)}\ (\ulcorner 0=1 \urcorner\ ,\ y\ ) \tag{6}$$

The preceding observations shows that ISREF(A) can derive from its axiom scheme (4) the rather conventional definition of its self-consistency given in (6).

FIFTH CLARIFYING COMMENT: There are many sentences Φ where the implication $\Phi \supset \Phi_i^i$ is false for all integers i. The reason (4) will turn out to be a valid axiom, despite this fact, is essentially that none of the troubling sentences Φ, that could cause problems, are actually provable from ISREF(A).

4 Consistency Preservation

This section will prove the consistency property of the ISREF axiom formalism. More precisely define a base axiom system A to be *Regularly Consistent* iff

1. the axiom system A is sound.
2. and $\mathrm{Prf}_A(s,x)$ is also encoded as a Δ_0^+ formula.

Define the "mapping" I($\circ$) to be *Consistency-Preserving* iff I(A) is consistent whenever A is regularly consistent. Say some Gödel encoding methodology satisfies

the *Conventional Sizing* constraints iff it is impossible for its encoded proofs p to physically contain a constant symbol representing a number $c > p-2$. Using our Size(p) function, this means that every proof p satisfies:

$$\mathrm{Size}(p) \quad \leq \quad p-2 \tag{7}$$

Since all conventional encoding methods satisfy constraints substantially more severe than (7), this definition is safely all-encompassing. Thus we may henceforth assume that the formula "Prf" in Equation (4)'s Group-3 axiom satisfies the inequality (7).

Theorem 4.1. If A is regularly consistent then ISREF(A) will be consistent. (Thus, ISREF($\circ$) is a consistency-preserving mapping.)

Proof Sketch. For the sake of contradiction, suppose ISREF(A) is inconsistent but A is regularly consistent. The latter implies all ISREF(A)'s Group-zero, Group-1 and Group-2 axioms are valid in the Standard Model of the Natural Numbers. Hence, at least one of ISREF(A)'s Group-3 axioms must be invalid in the Standard Model (since otherwise ISREF(A) would be consistent). From the definition of ISREF(A)'s Group-3 axiom (in (4)), this implies there exists a proof p of some theorem Ψ where

1. $i \geq \mathrm{SIZE}(p)$ and
2. $\neg\ \Psi_i^i$ is valid in the Standard Model.

Hence, there exists some triple $(\ \Psi_i^i\ ,\ p\ ,\ i\ )$ which satisfies *both conditions* (1) and (2) and has minimal value in its third component. We shall assume that $(\ \Psi_i^i\ ,\ p\ ,\ i\ )$ denotes this minimal triple. Our proof by contradiction will be completed by constructing from $(\ \Psi_i^i\ ,\ p\ ,\ i\ )$ another triple $(\ \Phi_j^j\ ,\ q\ ,\ j\ )$ with $j < i$.

Let M_i denote a finite model of the natural numbers which assumes that the only integers which exist are 0, 1, 2, ... i. Since p employs axioms from ISREF(A) to prove a theorem Ψ (which is invalid in M_i), certainly at least one axiom *physically appearing in* p must also be invalid in M_i. Since A is regularly consistent and since every constant symbol within p must represent an integer less than $p-1$, it is evident that all the Group-Zero, Group-1 and Group-2 axioms *appearing* in p are valid in M_i. Thus, since all other possibilities have been precluded, some Group-3 axiom lying within p must be invalid in M_i. The generic form of ISREF(A)'s Group-3 axioms is illustrated below:

$$\forall x\ \forall y \quad \{\quad \mathrm{Prf}_{\mathrm{ISREF}(A)}\ (\ulcorner \Phi \urcorner\ ,\ y\ ) \quad \wedge \quad x > \mathrm{Size}(y) \quad \supset \quad \Phi_{x-1}^{x-1} \quad \} \tag{8}$$

Since (8) is invalid in the model M_i, it follows that there must exist $q \leq i$ and $j \leq i-1$ such that $\mathrm{Prf}_{\mathrm{ISREF(A)}}\ (\ulcorner\ \Phi\ \urcorner\ ,\ q\ )$, $j \geq \mathrm{Size}(q)$ and $\neg\ \Phi_j^j$ hold. The triple $(\ \Phi_j^j\ ,\ q\ ,\ j\ )$ thus contradicts the minimality of $(\ \Psi_i^i\ ,\ p\ ,\ i\ )$. This contradiction shows that ISREF(A) must be consistent because the contrary assumption otherwise leads to an inherently impossible condition. □

5 Generalizations of Theorem 4.1.

From Equation (4), it is immediate that the ISREF(A) axiom system will support (2)'s tangibility reflection principle under each of the TangPred, TangDiv and

TangRoot definitions of tangibility. The distinction between these three definitions of tangibility becomes clearer when we consider some modifications of the ISREF axiom systems that include growth functions.

For example, if we add merely the bitwise-OR function to ISREF's Group-1 axioms then Theorem 4.1 will no longer be valid. To restore its validity, we will need to weaken the Group-3 axiom schema so that the resulting system (called ISTR(A)) has a Group-3 schema that supports the the TangDiv_2 but not TangPred definitions of tangibility.

Similarly, if the Group-1 axioms are extended to recognize Addition as a total function, then the Group-3 axiom will need to be further weakened so that they will recognize the validity of (2)'s tangibility reflection principle only when its formula "Prf" is taken to be a cut-free proof method (such as Herbrand proofs, semantic tableaux proof, or cut-free sequent calculus proofs) and simultaneously when a TangRoot_λ definition of tangibility is employed (for a suitably chosen constant $\lambda > 1$). The latter axiom systems is called $\text{IS}^\lambda(A)$. (A curious aspect of $\text{ISTR}(A)$ and $\text{ISREF}(A)$ is that although the B-Version of the Incompleteness Theorem (stated in Section 1) will not allow a consistent system to recognize Successor (or any faster growing operation) as a total function, it does not preclude them from recognizing an *encoded notion* of *"probability"* and a total function G(x) such that there is a "probability" of 1 that $G(x) > x$ for all x.)

Both $\text{ISTR}(A)$ and $\text{IS}^\lambda(A)$ are discussed by us in the papers [34, 36]. We postpone their discussion because $\text{ISTR}(A)$ and $\text{IS}^\lambda(A)$ are more likely to be of interest to logicians than to computer scientists. The remainder of the present article will be devoted to first introducing a new variant of the Incompleteness Theorem, which shows that further improvements of the $\text{ISREF}(A)$ will be very difficult, and then explaining how this subject matter is partially related to open questions about NP.

6 A New Variation of the Second Incompleteness Theorem:

This section will prove the new variation of Gödel's Second Incompleteness Theorem that was mentioned by Item C of Section 1. The main significance of Theorem 6.2 is that it will show that it is impossible to increase the superscript and subscript $x - 1$ in ISREF's Group-3 axiom by even a quantity as small as one.

Throughout the rest of this article, the symbol *PAX* will denote the trivial extension of Peano Arithmetic whose function symbols include all the Group-1 function symbols of ISREF, in addition to the usual function symbols of Addition and Multiplication. Let α denote any consistent axiom system which can prove all PAX's Π_1^+ theorems. We will show that α is unable to verify the reflection principle below for every Π_1^+ sentence $\forall v\, \phi(v)$.

$$\forall y \quad \{ \quad \text{Prf}_\alpha (\lceil \forall v\, \phi(v) \rceil , y) \quad \supset \quad \forall v\, \phi(v) \quad \} \tag{9}$$

It is well known how to Gödel-encode a sentence DIAGONAL(α), whose translation into English is roughly that:

There is no proof of this sentence from the axiom system α.

One common technique is to first construct the Δ_0^+ formula $\text{SUBST}(g, h)$ defined below:

SUBST(g, h) = The integer g is an encoding of a formula, and h encodes a sentence identical to g, except that all free variables in g are replaced with a constant, whose value equals g.

Then DIAGONAL(α) can be defined as being simply the sentence $\Omega(\bar{n})$, where $\Omega(g)$ denotes the formula given in (10) and $\bar{n}$ denotes $\Omega(g)$'s Gödel number.

$$\forall x\, \forall h \leq x \quad \text{SUBST}(g, h) \quad \supset \quad \neg\, \text{Prf}_\alpha (h , x) \tag{10}$$

Theorem 6.1. (*A corollary directly implied by Gödel's Incompleteness Theorem*): Let α be an axiom system which includes ISREF's Group-Zero and Group-1 axioms and where $\text{Prf}_\alpha(s, x)$ is a Δ_0^+ formula. If α proves DIAGONAL(α) then α is inconsistent.

Gödel's Proof Summarized: *If α could prove* DIAGONAL(α) then there would exist some standard integer p , which is the formal encoding of this proof. In this case, α must be able to prove the formal theorem-sentence below (since α contains all ISREF's Group-Zero and Group-1 axioms):

The integer p encodes a proof of DIAGONAL(α).

But the sentence (above) contradicts DIAGONAL(α), which is another theorem of α . Hence, α must be inconsistent if it proves DIAGONAL(α), □

Theorem 6.2. Suppose α is a consistent axiom system employing Hilbert deduction, $\text{Prf}_\alpha(s, x)$ is a Δ_0^+ formula, and α can verify all PAX's Π_1^+ theorems. Then α can not verify every Π_1^+ Reflection Sentence (whose canonical form, illustrated in Equation (9), assumes that $\phi(v)$ is a Δ_0^+ formula whose only free variable is v).

Proof Sketch. Let Θ be an abbreviation for DIAGONAL(α). Since $\text{Prf}_\alpha(h, x)$ and $\text{SUBST}(g, h)$ are Δ_0^+ formulae, Θ must be Π_1^+. Suppose for the sake of contradiction that α could prove all the Π_1^+ Reflection Principles (illustrated by (9)). Then since Θ is Π_1^+, the system α would then prove:

$$\forall y \quad \{ \quad \text{Prf}_\alpha (\lceil \Theta \rceil , y) \quad \supset \quad \Theta \quad \} \tag{11}$$

Hence (12) would also be a theorem of α , since it follows immediately from (11).

$$\{ \forall y \neg\, \text{Prf}_\alpha (\lceil \Theta \rceil , y) \quad \} \quad \vee \quad \Theta \tag{12}$$

Equation (13) (below) is clearly valid, since Θ = DIAGONAL(α) by definition. However, we need slightly more than this fact because we must establish that the *weak axiom system* α can verify (13). The latter is easy to justify because Θ's diagonalization construction implies there exists an integer $\bar{n}$ such that Θ's Gödel number is the *unique integer* satisfying $\text{SUBST}(\bar{n}, \lceil \Theta \rceil)$. More precisely, the reason α can prove $\lceil \Theta \rceil$'s Uniqueness is that it is provable from PAX as a

Π_1^+ theorem, and it is thus accessible to α (by Theorem 6.2's hypothesis). Once α has verified $\ulcorner \Theta \urcorner$'s Uniqueness, it can easily confirm:

$$\{ \forall y \neg \text{Prf}_\alpha (\ulcorner \Theta \urcorner , y) \} \quad \Leftrightarrow \quad \Theta \tag{13}$$

But if α can prove (12) and (13), it can clearly derive Θ as another theorem. By Theorem 6.1, the latter implies α is inconsistent. Hence, α's inconsistency must necessarily follow when α simultaneously proves all PAX's Π_1^+ theorems and it also validates all (9)'s Π_1^+ Reflection Principles. □

Remark 6.3. The generalizations of Theorem 6.2 for cut-free methods of deduction, Herbrand deduction and semantic tableaux deduction are also valid [36].

Remark 6.4. The first paragraph of this section claimed that ISREF(A) would become inconsistent if one changed its subscript and superscript of $x-1$ to x in its axiom sentence (4). To understand why this is true, let us assume that Ψ designates the Π_1^+ sentence $\forall v\ \phi(v)$ where $\phi(v)$ is Δ_0^+. The revised Group-3 axiom scheme of ISREF(A) would then imply the validity of

$$\forall x \forall y \ \{ \ \text{Prf}_{\text{ISREF}_{(A)}} (\ulcorner \forall v \phi(v) \urcorner , y) \wedge x > \text{Size}(y) \ \supset \ \forall v \leq x \ \phi(v) \ \} \tag{14}$$

Since Equation (7) indicates $\text{Size}(y) \leq y-2$, we can infer from (14) the validity of both (15) and (16).

$$\forall y \ \{ \ \text{Prf}_{\text{ISREF}_{(A)}} (\ulcorner \forall v \phi(v) \urcorner , y) \ \supset \ \forall v \leq y \ \phi(v) \ \} \tag{15}$$

$$\forall x \forall y \ \{ \ \text{Prf}_{\text{ISREF}_{(A)}} (\ulcorner \forall v \phi(v) \urcorner , y) \wedge x > y \ \supset \ \forall v \leq x \ \phi(v) \ \} \tag{16}$$

Moreover, the latter two equations easily imply (17) is valid for every Π_1^+ sentence of the form $\forall v\ \phi(v)$.

$$\forall y \ \{ \ \text{Prf}_{\text{ISREF}_{(A)}} (\ulcorner \forall v \phi(v) \urcorner , y) \ \supset \ \forall v \ \phi(v) \ \} \tag{17}$$

Hence we have established that if the " $x-1$ " in ISREF(A)'s Group-3 axioms was changed to " x ", then the modified-ISREF(A) would satisfy the hypothesis of Theorem 6.2. The force of Theorem 6.2 will then substantiate our earlier claim that this modified version of ISREF(A) is inconsistent.

7 Self Justification and NP

We will now turn to the P =? NP Open Question. Let OUTPUT(e,t,x) be a Δ_0^+ formula asserting that if e is an integer designating the string of bits initially lying on $\mathcal{M}$'s "encoding tape" $T_\mathcal{E}$ and if its three "input tapes" T_1, T_2 and T_3 each initially encode the integer zero, then x will designate the bit-sequence on $\mathcal{M}$'s "output tape" T_1 at the time t. Also, define $\text{Prf}^*{}_\alpha(x,y)$ to be a Δ_0^+ formula indicating y is the smallest integer which is a Gödel number of a proof from axiom system α of the sentence with Gödel number x. Let us assume that $\forall v\, \phi(v)$ is a Π_1^+ sentence where $\phi(v)$ is Δ_0^+. For any prespecified non-decreasing function $R(y)$, satisfying $R(y) < y$, define ISREF^R(A) to be an axiom system identical to ISREF(A), except that its Group-3 axioms are of the form:

$$\forall y \forall x \leq y \forall e \leq R(y) \forall t \leq R(y) \ \{ [\ \text{Prf}^*{}_{\text{ISREF}^R(\text{A})} (\ulcorner \forall v\, \phi(v) \urcorner , y) \wedge \text{OUTPUT}(e,t,x)] \supset \phi(x) \} \tag{18}$$

As before, the Fixed Point Theorem will imply (18) easily has a formal Gödel encoding. In particular, the technique for encoding (18)'s self-referencing property is essentially identical to the Appendix's encoding of the self-referencing property for Equation (4).

We will be mostly interested in the ISREF^R(A) axiom system for the special case where its function R(y) satisfies the inequality $R(y) \gg \text{PolyLog}(y)$. (Here the notation " $R(y) \gg \text{PolyLog}(y)$ " means that the limit of the ratio $\frac{R(y)}{\text{PolyLog}(y)}$ approaches infinity as y approaches infinity.) For such R it will be unclear whether or not the ISREF^R mapping satisfies an analog of Theorem 4.1's Consistency-Preservation property. However, Section 6 will prove P≠NP if this analog of Theorem 4.1 holds for ISREF^R.

We will now describe a second variation of the same theorem. Recall that PAX denotes the trivial extension of Peano Arithmetic whose function symbols include all the Group-1 function symbols of ISREF, in addition to the usual function symbols of Addition and Multiplication. Let *SAT(a,b)* be the statement that the algorithm with Gödel number a can successfully resolve any SAT problem of length N in deterministic time bounded by N^b . Theorem 9.5 will show that if ISREF^R(PAX) *is consistent for some function* $R(y) \gg \text{O}(\text{PolyLog}(y)$ then Peano Arithmetic will be unable to prove any P=NP theorem of the explicit form: *SAT(a,b)*.

The italicized phrase (above) is called the *Cognitive Conjecture*. The conjecture merely requires that there exist a function $R(y)$, growing *very slightly* faster than PolyLog(y), such as perhaps say $R(y) = Log(y)^{LogLogLog(y)}$, where ISREF^R(PAX) is consistent. This would be sufficient to establish that Peano Arithmetic is unable to firmly prove P=NP.

Moreover, if the stronger version of the Cognitive Conjecture holds (indicating that the ISREF^R mapping is actually a consistency-preserving transformation) then Theorem 9.6 will imply P≠NP.

Our remaining discussion will be divided into two parts. Section 8 explains why we suspect that some version of the Cognitive Conjecture is likely to be correct. Section 9 will present our main theorems showing that the Cognitive Conjecture implies P≠NP.

8 Intuition Behind The Cognitive Conjecture

Let us assume that $\forall v\, \phi(v)$ is a Π_1^+ sentence where $\phi(v)$ is Δ_0^+. Define ISREF*(A) to be an axiom system identical to ISREF(A), except that its Group-3 axioms are of the form:

$$\forall x \ \forall y \ \{ \text{Prf}^* \ \text{ISREF}^*(\text{A}) (\ulcorner \forall v\, \phi(v) \urcorner , y) \ \wedge \ \text{TangPred}(x) \ \} \ \supset \ \phi(x) \tag{19}$$

Once again, the Fixed Point Theorem easily implies (19) has a formal Gödel encoding. The noteworthy aspect of the system ISREF*(A) is that its Group-3 axioms are patently weaker than ISREF(A)'s Group-3 axioms. This is because the latter's Equation (4) asserts the validity of a reflection principle for all sentences Ψ , whereas the former's reflection principle (in Eq. (19)) applies only to Π_1^+ sentences of the form $\forall v\, \phi(v)$, where $\phi(v)$ is Δ_0^+.

Thus for each axiom system A where ISREF(A) is consistent, we may immediately infer that so is ISREF*(A) consistent. In particular, we may infer from Theorem 4.1 that ISREF*(PAX) is consistent.

Now let us compare the axiom system ISREF*(PAX) to ISREF^R(PAX), and consider whether it is likely that the former system could be consistent but the latter system inconsistent?

The key point is that there are two respects in which ISREF^R(PAX)'s reflection principle (18) is demonstrably weaker than ISREF*(PAX)'s reflection principle (19). The first is due to the presence of the Δ_0^+ formula OUTPUT(e,t,x) in ISREF^R(PAX)'s reflection principle (18), and the second is because ISREF^R(PAX)'s reflection principle replaces a Prf(x, y) formula with a Prf$^*(x, y)$ formula. (Recall that Prf$^*_\sigma(x, y)$ is satisfied only when y designates *the smallest* Gödel number which is a proof of the sentence x).

The net effect is that if " $\forall v\, \phi(v)$ " is a theorem of ISREF^R(PAX) then there are only a *finite set of predictions* following from its associated reflection principle (18). In particular, this finite set of predictions is constructed by first finding the integer N which is *the smallest* proof of " $\forall v\, \phi(v)$ ", and then finding those integers x that satisfy the condition:

$$\exists e \leq R(N)\ \ \exists t \leq R(N) \quad \text{OUTPUT}(e,t,x) \tag{20}$$

On the other hand, if " $\forall v\, \phi(v)$ " is a theorem of ISREF*(PAX), then there are actually *an infinite number of predictions* following from its reflection principle (19). In particular, all the standard integers x are predicted by (19) to satisfy $\phi(x)$ (because all the standard integers obviously satisfy TangPred(x)). Indeed since there can be *at most* one non-standard integer that does not have a successor under any model of ISREF*(PAX), there can be at most one single (non-standard) integer x that fails to satisfy $\phi(x)$!

Thus at a superficial glance, it would certainly appear that ISREF^R(PAX)'s reflection principle (18) provides a much less restrictive set of predictions than ISREF*(PAX)'s reflection principle (19). Moreover, what reinforces this appearance further is that our proofs in the next section will allow us to employ any function $R(y)$, that grows ever so slightly faster than PolyLog(y), such as for example $R(y) = Log(y)^{LogLogLog(y)}$.

Our open question is, thus, whether or not the appearances, described by the preceding three paragraphs, are really deceiving? If these appearances are not misleading, then the fact that ISREF^R(PAX) is consistent, will imply that so is ISREF^R(PAX) consistent (when it is assigned a slow enough growing function $R(y) \gg$ PolyLog(y)).

The latter conjecture, if correct (?), in conjunction with the next section's Theorem 9.5, would provide a partial answer to P =? NP Open Problem.

9 THE P =? NP QUESTION

Let Θ^R be an abbreviation for DIAGONAL(ISREF^R(PAX)). Recall that *SAT(a,b)* specifies that the algorithm a can resolve any SAT problem of length N in time less

than N^b.

Lemma 9.1 Suppose P=NP and SAT(a,b) is true. From such (a,b), we can construct an ordered pair (q,d) such that *if any integer exists* which is a proof from ISREF^R(PAX) of Θ^R then the procedure q (running on Section 3's multi-tape machine $\mathcal{M}$) will find the minimal integer s which is a proof from ISREF^R(PAX) of Θ^R in no more than $(\text{Log}\, s)^d$ units of time.

Proof. An algorithm q that meets the Lemma's requirements will be a 2-part procedure, whose two modules are called q_1 and q_2. The module q_1 will walk through the set of positive integers in ascending order, and for each integer i apply the "SAT-Resolution subroutine" a to test whether there exists a bit string s of length=i such that s is a proof of Θ^R from ISREF(A). The module q_1 will never halt, if it does not find such a number i. If it does find the required integer, then the procedure q will ask its second module q_2 to construct the minimal integer s that is a proof of Θ^R.

The module q_2 will construct the string s by essentially making i further subroutine-calls to the the procedure a. Basicly, the j-th subroutine-call will supply a with a list of s's previously computed first $j-1$ bits $b_1, b_2, ...b_{j-1}$, and it will ask a whether there exists a proof of Θ^R, which begins with these first $j-1$ bits followed by the bit zero (and which has a length $= i$). If a replies the answer is "Yes" then q_2 will set the bit b_j equal to zero; otherwise it will set this bit equal to one. The particular bit string that is constructed by q_2, after its i subroutine calls to a are completed, constitutes the smallest integer which is a proof from ISREF^R(PAX) of Θ^R. This integer will be output of the procedure q.

Since the subroutine a can solve SAT problems of length N in deterministic time $< N^b$, it follows that the procedure q can use its subroutine-calls to a to do all its hard work. Thus, there will exist a constant d such that q will construct s in no more than $N^d \leq (\text{Log}s)^d$ time (when s exists). □

NOTATION: The following two logical expressions will be employed by our next three lemmas:

PARADOX$^R(y, c)$ is a Δ_0^+ formula which states that y is the smallest integer that represents a proof from ISREF^R(PAX) of Θ^R, and that a procedure c running on our multi-tape Turing Machine M with a configuration $(0,0,0)$ initially stored on its three input tapes, will halt before $R(y)$ units of time with the integer y written as its final output.

$\Upsilon_c(k)$ is a Π_1^+ sentence asserting that $\forall y \geq k$ { Prf$^*_{\text{ISREF}^R(\text{PAX})}(\lceil \Theta^R \rceil, y) \supset$ PARADOX$^R(y,c)$ }.

Lemma 9.2. If $R(y) \gg$ PolyLog(y) then the following two assertions hold:

A. SAT(a,b) implies that there exists two constants c and k such that $\Upsilon_c(k)$ is true.

B. If Peano Arithmetic (or equivalently PAX) can prove SAT(a,b) then there exists two constants c and k such that ISREF^R(PAX) can prove $\Upsilon_c(k)$.

Comment. Before proving Lemma 9.2, we should explain its intuitive significance. From an ordered pair (a, b) satisfying $\mathrm{SAT}(a, b)$, Lemma 9.2 will show how to construct a second ordered pair (c, k) satisfying $\Upsilon_c(k)$. Such an ordered pair (c, k) satisfying $\Upsilon_c(k)$ will shown by Propositions 9.3 through 9.5 to force $\mathrm{ISREF}^R(\mathrm{PAX})$ to be inconsistent if Peano Arithmetic can prove $\mathrm{SAT}(a, b)$. It will be this fact that will enable Theorem 9.6 to establish that there is a connection between the P≠NP open problem and the properties of the ISREF^R axiom family.

Proof of Assertion A. Consider the formula $\mathrm{Prf}^*_{\mathrm{ISREF}^R(\mathrm{PAX})}(\lceil \Theta^R \rceil, y)$ on the left side of $\Upsilon_c(k)$'s horn clause. If y satisfies this formula, then Lemma 9.1's procedure q will output y in $(\mathrm{Log}\ y)^d$ units of time. Moreover since $R(y) \gg \mathrm{PolyLog}(y)$, there will exist some constant k (whose values depend on d and on the function $R(y)$) such that all $y \geq k$ will satisfy $(\mathrm{Log}\ y)^d < R(y)$. The preceding two observations immediately imply one can always construct an ordered pair (c, k) from (a, b), where $\mathrm{SAT}(a, b)$ implies $\Upsilon_c(k)$. □

Proof of Assertion B. It is easy to see that Peano Arithmetic (and also its minor extension PAX) can *themselves* prove Assertion A (because all the steps in both the proof of Lemma 9.1 and in the paragraph above are easily accessible to these formalisms). Hence, if PAX could *additionally prove* $\mathrm{SAT}(a, b)$, then (using Assertion A) it could also prove a theorem of the form $\Upsilon_c(k)$, for some pair of constants (c, k). But the latter sentence is Π_1^+ . This implies that the power of $\mathrm{ISREF}^R(\mathrm{PAX})$'s Group-2 axioms will enable $\mathrm{ISREF}^R(\mathrm{PAX})$ (itself) to prove $\Upsilon_c(k)$. □

Lemma 9.3 For each fixed constant c, $\mathrm{ISREF}^R(\mathrm{PAX})$ can prove the theorem statement $\forall y \ \neg \mathrm{PARADOX}^R(y, c)$.

Proof. Choose a constant k sufficiently large so that $R(k) > c$. In order to verify Lemma 9.3, it is sufficient to show that $\mathrm{ISREF}^R(\mathrm{PAX})$ can prove BOTH (i) and (ii) below:

i. $\forall y \leq k \ \neg \mathrm{PARADOX}^R(y, c)$.

ii. $\forall y > k \ \neg \mathrm{PARADOX}^R(y, c)$.

Proof that $\mathrm{ISREF}^R(\mathrm{PAX})$ can verify (i): If $\mathrm{ISREF}^R(\mathrm{PAX})$ is inconsistent then it can trivially prove any sentence, including (i). On the other hand, if $\mathrm{ISREF}^R(\mathrm{PAX})$ is consistent then it is immediate from the definition of $\Theta^R = \mathrm{DIAGONAL}(\mathrm{ISREF}^R(\mathrm{PAX}))$, that no proof of Θ^R can exist (since otherwise Theorem 6.1 would imply $\mathrm{ISREF}^R(\mathrm{PAX})$ is inconsistent). Using $\mathrm{PARADOX}^R(y, c)$'s definition, this implies that the Δ_0^+ formula given in (i) is true. However, every valid Δ_0^+ formula is automatically provable from $\mathrm{ISREF}^R(\mathrm{PAX})$. Thus $\mathrm{ISREF}^R(\mathrm{PAX})$ proves (i). Hence, it proves (i) in both the cases where $\mathrm{ISREF}^R(\mathrm{PAX})$ (is) and (is not) consistent. □.

Proof that $\mathrm{ISREF}^R(\mathrm{PAX})$ can verify (ii): Let $\forall x\, \phi(x)$ be an abbreviation for Θ^R . Then since $\phi(x)$ is a Δ_0^+ formula, it follows that $\mathrm{ISREF}^R(\mathrm{PAX})$ will have a Group-3 axiom of the form:

$$\forall y \forall s \leq y \forall e \leq R(y) \forall t \leq R(y) \ \{[\ \mathrm{Prf}^*_{\mathrm{ISREF}^R(\mathrm{PAX})}(\lceil \forall x \phi(x) \rceil, y) \wedge \mathrm{OUTPUT}(e, t, s)] \supset \phi(s)\} \tag{21}$$

Having y substitute for s, having c substitute for e, and using the fact that $R(k) > c$, $\mathrm{ISREF}^R(\mathrm{PAX})$ can trivially derive (22) from (21).

$$\forall y > k \ \forall t \leq R(y) \quad \{\ \mathrm{Prf}^*_{\mathrm{ISREF}^R(\mathrm{PAX})}(\lceil \forall x \phi(x) \rceil, y) \wedge \mathrm{OUTPUT}(c, t, y)\ \} \supset \phi(y) \tag{22}$$

We claim that $\mathrm{ISREF}^R(\mathrm{PAX})$ can deduce (ii) from its theorem (22). From the definitions of $\mathrm{PARADOX}^R(y, c)$, ϕ and Θ^R, one can fairly readily verify that (22) implies (ii). However, this is technically insufficient to prove our claim because it will require that *the weak axiom system* $\mathrm{ISREF}^R(\mathrm{PAX})$ can deduce (ii) from (22).

In order to show that $\mathrm{ISREF}^R(\mathrm{PAX})$, *itself (!)*, can deduce (ii) from (22), we will employ a technique analogous to Theorem 6.2's proof. Since $\mathrm{ISREF}^R(\mathrm{PAX})$'s Group-2 axioms enable it to prove all the Π_1^+ theorems of PAX, $\mathrm{ISREF}^R(\mathrm{PAX})$ can prove the Π_1 theorem that

> $\lceil \Theta^R \rceil$ is the unique integer satisfying $\mathrm{SUBST}(\bar{n}, \lceil \Theta^R \rceil)$ (where $\bar{n}$ once again denotes the base integer employed by equation (10)).

As in the proof of Theorem 6.2, once $\mathrm{ISREF}^R(\mathrm{PAX})$ has verified that $\lceil \Theta^R \rceil$ is the unique integer satisfying $\mathrm{SUBST}(\bar{n}, \lceil \Theta^R \rceil)$, the remainder of its proof of (ii) is straightforward. That is, Assertion (ii) follows quite directly from the indented sentence and the definitions of PARADOX, Θ^R, ϕ and Eq. (22). (The remaining details of (ii)'s proof, which are quite straightforward, are provided in the footnote below[3].) □

Lemma 9.4. If $\mathrm{ISREF}^R(\mathrm{PAX})$ can prove $\Upsilon_c(k)$, for any two constants c and k, then $\mathrm{ISREF}^R(\mathrm{PAX})$ is inconsistent.

Proof of Lemma 9.4. when k=0 : Lemma 9.3 indicated that $\mathrm{ISREF}^R(\mathrm{PAX})$ can prove that $\forall y \ \neg \mathrm{PARADOX}^R(y, c)$. Thus, *if it could also prove* $\Upsilon_c(0)$ then $\mathrm{ISREF}^R(\mathrm{PAX})$ could combine these two theorems to immediately deduce that

> "No integer constitutes a proof from $\mathrm{ISREF}^R(\mathrm{PAX})$ of Θ^R "

Recall that $\Theta^R = \mathrm{DIAGONAL}(\mathrm{ISREF}^R(\mathrm{PAX}))$, by definition. Hence the indented sentence is equivalent to Θ^R, and we are forced to thereby conclude that $\mathrm{ISREF}^R(\mathrm{PAX})$ can prove Θ^R ! Applying Theorem 6.1, it is apparent that the axiom system $\mathrm{ISREF}^R(\mathrm{PAX})$ is inconsistent. □

Proof when $k > 0$. First, suppose $\mathrm{ISREF}^R(\mathrm{PAX})$ can prove the formal sentence that $\forall y \leq k \ \neg \mathrm{Prf}_{\mathrm{ISREF}^R(\mathrm{PAX})}(\lceil \Theta^R \rceil, y)$ (in addition to proving

[3]The easiest way to show that "Eq. (22) implies (ii)" is to verify its contrapositive (i.e. to prove the statement that "Not (ii) implies Not Eq. (22)"). So let us therefore assume that $\exists y > k$ $\mathrm{PARADOX}^R(y, c)$. By the definition of PARADOX, this implies that y is a proof of Θ^R . By the definitions of Θ^R and ϕ, the preceding sentence implies $\neg\phi(y)$ (essentially because $\phi(y)$, by definition, states that y is not a proof of Θ^R from $\mathrm{ISREF}^R(\mathrm{PAX})$). Hence, y fails to satisfy the right side of (22)'s horn clause. At the same time, the definition of PARADOX implies that y must satisfy the curly bracket expression in (22). These two observations implies Eq. (22) is false. Hence, we have shown that "Not (ii) implies Not Eq. (22)". The main point is perhaps not the detailed proof, delineated in this footnote, which is actually a quite routine diagonalization argument. Rather the main point is that the indented sentence (in the last paragraph of Theorem 9.3's proof) assured that $\mathrm{ISREF}^R(\mathrm{PAX})$ can internally verify all the details given in this footnote's proof, *after* $\mathrm{ISREF}^R(\mathrm{PAX})$ has confirmed that $\lceil \Theta^R \rceil$ is actually the unique integer satisfying $\mathrm{SUBST}(\bar{n}, \lceil \Theta^R \rceil)$.

$\Upsilon_c(k)$.) From these two theorems, ISREF^R(PAX) can easily deduce the further theorem $\Upsilon_c(0)$. Hence, ISREF^R(PAX) must be inconsistent (by the argument, for the case $k = 0$, given in the preceding paragraph).

On the other hand, suppose that ISREF^R(PAX) *can not prove* the formal sentence that $\forall y \leq k \neg \text{Prf}_{\text{ISREF}^R(\text{PAX})}(\lceil \Theta^R \rceil, y)$. Since all valid Δ_0^+ sentences are provable, this implies $\exists y \ \text{Prf}_{\text{ISREF}^R(\text{PAX})}(\lceil \Theta^R \rceil, y)$. The latter (via Theorem 6.1) implies ISREF^R(PAX) is inconsistent. Hence it is inconsistent in both cases. □

Theorem 9.5. If the function $R(x) \gg \text{O}(\text{PolyLog}(x))$ and ISREF^R(PAX) is consistent then Peano Arithmetic (and also PAX) will be unable to prove a theorem of the form SAT(a, b).

Proof. Suppose for the sake of contradiction that Peano Arithmetic could prove SAT(a, b). Then ISREF^R(PAX) would prove a theorem of the form $\Upsilon_c(k)$ (by Lemma 9.2B), which would in turn imply that ISREF^R(PAX) is inconsistent (by Lemma 9.4). Hence, Theorem 9.5 must be valid because it is infeasible for simultaneously ISREF^R(PAX) to be consistent and Peano Arithmetic to prove SAT(a, b).

Theorem 9.6. Suppose that the function $R(x) \gg \text{O}(\text{PolyLog}(x))$ and the corresponding ISREF^R mapping satisfies the "consistency-preservation" property (defined in Section 4). Then P≠NP.

Proof Sketch. Let α denote an arbitrary regularly consistent axiom system that is an extension of PAX. It is straightforward to generalize Theorem 9.5's proof to establish that:

> If ISREF$^R(\alpha)$ is consistent then α is unable to prove a theorem of the form SAT(a, b).

But if P=NP, then we can construct a consistent system α which is simply the union of PAX with some sentence of the form SAT(a, b). Since this system trivially proves SAT(a, b), the indented statement requires that ISREF$^R(\alpha)$ be inconsistent. Since α is consistent but ISREF$^R(\alpha)$ is inconsistent, the ISREF^R mapping is certainly not consistency-preserving. Hence, Theorem 9.6 is valid because it is infeasible for simultaneously P=NP and ISREF^R to be consistency-preserving. □.

Added Comment: The enticing aspect of Theorems 9.5 and 9.6 is that they offer a possible new approach for studying the P =? NP Open Question. However, one should harbor no illusions about how difficult it may ultimately turn out to be to prove that ISREF^R(PAX) is consistent. The perplexing aspect about ISREF^R(PAX) is that it looks very similar to both the consistent axiom system ISREF*(PAX), defined in Section 8, and to the inconsistent axiom systems α, discussed by Theorem 6.2. A very subtle mathematical argument will thus likely be needed to prove that ISREF^R(PAX) is consistent, if indeed our conjecture about its consistency, given in Section 8, is correct (?).

10 Further Results

There is also a second respect in which self-justifying axiom systems have properties very closely linked to the P =? NP Open Problem. This second connection requires

a full-length paper, unto itself, if one wishes to do the proofs rigorously. Therefore, we will only provide a short summary of it in this section, with the proofs and a more formal description postponed until another article [35].

Define ISTM$^\lambda(A)$ to be an axiom system, similar to ISREF(A), except for the following two changes:

i. The Group-Zero axioms of ISTM$^\lambda(A)$ consists of the axiom $\bar{1} \neq \bar{0}$, plus one additional axiom of the form Predecessor$(\overline{n})$ $= \overline{n-1}$, for each $n > 0$. This is different from ISREF(A)'s Group-Zero schema, which also contained additional axioms of the forms $\overline{2n} - \overline{n} = \overline{n}$ and $\overline{2n+1} - \overline{n} - \bar{1} = \overline{n}$.

ii. For a prespecified constant λ, the Group-3 axiom of ISTM$^\lambda(A)$ will state that $\forall z_1 \ \forall z_2 \ \forall z_3 \ \forall z_4 \ \forall p \ \forall t$ the multi-tape Turing machine $\mathcal{M}$, which initially contains the tuple $(z_1, z_2, z_3, z_4,)$ stored on its four tapes and which simulates Section 3's $Tape_i(e, x, t)$, $Head_i(e, t)$, and $State_i(e, t)$ functions, will never place at any time $t < p^\lambda$ a bit-stream on any of its tapes which corresponds to a proof of 0=1 from the axiom system ISTM$^\lambda(A)$.

It is possible to apply the Fixed Point Theorem to formally encode the revised Group-3 schema above, provided we select a proper initial constant λ which satisfies the inequality $0 < \lambda < 1$. (The formal proof of this fact is one of the many details which are postponed until the paper [35].)

The key distinction between ISTM$^\lambda(A)$ and ISREF(A) is that the former axiom system has a weaker form of Group-Zero axiom schema, but an essentially stronger form of Group-3 schema. It turns out that it is necessary to make both changes in ISTM$^\lambda(A)$'s definition (because ISTM$^\lambda(A)$ would be inconsistent if we made only the second change.) In particular, it is shown in [35] that the ISTM$^\lambda(A)$ formalism satisfies an analog of Theorem 4.1, which will state that the mapping from an initial axiom system A onto a second axiom system ISTM$^\lambda(A)$ is a "consistency-preserving" transformation.

Recall that an axiom system S is said to *Firmly Prove* P=NP iff there exists an ordered pair (a, b) such that S can prove that the algorithm with Gödel number a can solve any SAT problem of length $N \geq 2$ in deterministic time $< N^b$ (i.e. S proves a theorem of the form SAT(a, b) for some pair of constants (a, b)). Also, define CASCONS$_\alpha(k)$ to be the assertion that:

> $\forall x_1 \ \forall x_2 \ \ldots \ \forall x_k$ the natural bitwise concatenation of the integers $x_1, x_2, \ldots x_k$ does not represent a formal proof of "0=1" from the axiom system α.

Say an axiom system α is *Cascade-Self-Verifying* iff it can prove the theorem CASCONS$_\alpha(k)$ for each fixed integer k. In this context, the two main theorems of [35] are listed below:

1. Either ISTM(PAX) will be a cascade-self-verifying axiom system or Peano Arithmetic must be unable to firmly prove P=NP.

2. Either some extension A of Peano Arithmetic will map onto a cascade-self-verifying ISTR$^\lambda(A)$ system or P≠NP.

The curious aspect of the two theorems above is that it is very difficult to construct cascade-self-verifying axiom systems. For example, none of the axiom systems ISREF(A), ISTR(A) or IS$^{\lambda}$(A), defined in Sections 3 and 5, were cascade-self-verifying. It is therefore reasonable to conjecture that none of the members of the ISTM$^{\lambda}$(A) axiom family will be cascade-self-verifying. But the latter conjecture, if correct (?), would imply a resolution to the P =? NP Open Problem, via Theorems 1 and 2 (above).

Thus, the article [35] will establish a second type of link between self-justifying axiom systems and the P =? NP Open Problem, quite different from Theorems 9.5 and 9.6 from the preceding section.

Acknowledgement: I would like the thank the referee for his many very helpful comments about how to improve the presentation of this paper.

Appendix: The Gödel Encoding of Group-3 Axioms

This section will explain how to formally encode the Group-3 axioms for ISREF(A). (The same techniques are also applicable to the ISREF^{R}(A) and ISREF*(A) axiom systems.) In our discussion, the reader should keep in mind that Rogers [26] and Jeroslow [13] have noted that Kleene's Fixed Point Theorem [14] can be used to expand any initial r.e. axiom system α into a broader system α^{*}, whose one additional axiom is:

> The union of the system α **WITH THIS ADDITIONAL SENTENCE** is a consistent axiom system.

The reason such systems α^{*} have not occurred very often in the prior literature is that they are typically inconsistent, despite the fact that they axiomatically do justify their own consistency (as Section 1 had already explained). The most novel aspects of ISREF(A) will thus *not be* that it contains Group-3 axioms, similar to the indented sentence above, but rather that it is actually consistent (i.e. see Theorem 4.1). For this reason, we strongly recommend that the reader postpone examining this Appendix's formal Gödel encoding of ISREF(A)'s Group-3 axiom, until after he has read the more novel results in Sections 2 through 9. (Indeed, Sections 2-9 were carefully written so that they can be understood by a reader who has not examined this Appendix.)

Like all Gödel encodings, the discussion in this Appendix will be somewhat tedious. We will employ the notation defined below.

i. Prf$_{\alpha}$ (s, y) will once again be a formula designating the byte-string y is an encoding of a proof of the sentence s from the axiom system α.

ii. UNION(A) will denote the union of ISREF's Group-Zero, Group-1 and Group-2 axioms.

iii. ExPrf$_{\alpha}$ (s, y, h) is a formula stating that y is a proof of s from the union of the axiom system α together with the axiom sentence whose integer encoding $= h$. The special case of ExPrf where α corresponds to UNION(A) will be denoted as ExPrf$_{\text{UNION}(A)}$(s, y, h).

iv. SUBST(g, h) will denote Gödel's classic substitution formula, which yields TRUE when g is an encoding of a formula, and h is an encoding of a sentence which replaces all occurrence of free variables in g with simply a constant, whose value equals the integer g.

Let IS_(A) denote an axiom system identical to ISREF(A) except that IS_(A)'s Group-3 axiom "informally" represents the expression (23) below.

$$\forall y \quad \neg \text{Prf}_{\text{IS\_}(A)}(\lceil 0 = 1 \rceil , y) \tag{23}$$

The Group-3 axiom of IS_(A) is trivial to "formalize" because (23) is the only sentence belonging to IS_(A)'s Group-3 axiom schema. In particular, let $\Theta(g)$ denote the formula:

$$\forall y \; \forall h \quad \{ \text{ SUBST}(g, h) \supset \neg \text{ ExPrf}_{\text{UNION}(A)} (\lceil 0 = 1 \rceil , y , h) \} \tag{24}$$

Let N denote the Gödel number of the formula $\Theta(g)$ above. Then the "formal Gödel encoding" of IS_(A)'s Group-3 axiom (23) can be simply defined as the sentence $\Theta(N)$.

The main goal in this appendix is to describe the analog of $\Theta(N)$ needed by the more complex Group-3 axioms of ISREF(A). (Note that ISREF(A)'s Group-3 schema will be more complicated than IS_(A)'s Group-3 axiom (23) primarily because ISREF(A)'s Group-3 schema will consist of an infinite number of distinct axiom-sentences).

Define a *g-pseudo-formula* to be a logical formula whose only free variable is g and which differs from conventional formulae by allowing a pseudo-symbol, denoted as " ♣ ", to appear anywhere in the formula. Usually, $\Xi(g, ♣)$ will denote a g-pseudo-formula. The symbol " ♣ " will be a place-holder where an arbitrary prenex normal sentence Φ may be later inserted into Ξ. It *would be slightly inaccurate* to view " ♣ " as a second-order variable because " ♣ " can *designate either* Φ or its Gödel number $\lceil \Phi \rceil$, depending on exactly where in Ξ the marker " ♣ " is located. (The intended meaning of the symbol " ♣ " will always be quite self-evident within the context of where this symbol appears within the formula Ξ.) We will also use the following notation:

v. ClubSet(K, Ξ) is defined only when Ξ is a g-pseudo-formula. In this case (with the integer K fixed), ClubSet(K, Ξ) will consist of the set of all formulae of the form $\Xi(K, \Phi)$ where Φ is prenex normal and where the ordered pair (K, Φ) substitutes for $(g, ♣)$ in the formula $\Xi(g, ♣)$

vi. ExPrf$^{*}_{\alpha}$ (s, y, h) will be a generalization of item (iii)'s ExPrf$_{\alpha}$ (s, y, h) formula, designed to take into account g-pseudo-formulae. In particular, if h is the Gödel number of a g-pseudo-formula $\Xi(g, ♣)$ then ExPrf$^{*}_{\alpha}$ (s, y, h) will indicate that y is a proof of the sentence s from the union of the two axiom systems α and ClubSet(h, Ξ). On the other hand, if h is not the Gödel number of a g-pseudo-formula then ExPrf$^{*}_{\alpha}$ (s, y, h) is defined to equal Prf$_{\alpha}$ (s, y).

Added Comment. Theorem A.1 will prove that ExPrf$^{*}_{\text{UNION(A)}}$(s, y, h) can be actually encoded as a Δ_0^{+} formula.

Section 3 had indicated that ISREF(A) would contain one Group-3 axiom Ω_Ψ for each prenex normal sentence Ψ. These axioms were previously "informally" defined (in Section 3) as:

$$\forall x \, \forall y \quad \{ \quad \text{Prf}_{\text{ISREF}(A)} (\ulcorner \Psi \urcorner , y) \ \wedge \ x > \text{Size}(y) \quad \supset \quad \Psi_{x-1}^{x-1} \ \} \qquad (25)$$

In order to "formally" Gödel-encode (25), it is useful to employ the g-pseudo-formula $\Xi(g, \clubsuit)$ below. Note (26) differs from (25) by replacing " Ψ " with " $\clubsuit$ " and replacing " $\text{Prf}_{\text{ISREF}(A)} (\circ , y)$ " with " $\text{ExPrf}^*_{\text{UNION}(A)} (\circ , y , g)$ ".

$$\forall x \, \forall y \quad \{ \quad \text{ExPrf}^*_{\text{UNION}(A)} (\ulcorner \clubsuit \urcorner , y , g) \ \wedge \ x > \text{Size}(y) \quad \supset \quad \clubsuit_{x-1}^{x-1} \ \} \qquad (26)$$

Let N denote the Gödel number of (26). (Thus, N represents " $\Xi(g, \clubsuit)$ "). Then the *informal expression* " $\text{Prf}_{\text{ISREF}(A)} (s , y)$ " can be *formally encoded* as: " $\text{ExPrf}^*_{\text{UNION}(A)} (s , y , N)$ ". This implies that the *Formal Gödel Encoding* of ISREF(A)'s Group-3 axioms (from (25)) is illustrated by Equation (27):

$$\forall x \, \forall y \quad \{ \quad \text{ExPrf}^*_{\text{UNION}(A)} (\ulcorner \Psi \urcorner , y , N) \ \wedge \ x > \text{Size}(y) \quad \supset \quad \Psi_{x-1}^{x-1} \ \} \qquad (27)$$

Theorem A.1. $\text{ExPrf}^*_{\text{UNION(A)}}(s, y, h)$ can be encoded as a Δ_0^+ formula. Hence all ISREF(A)'s Group-3 axioms (illustrated by Eq. (27)) can be encoded as Π_1^+ sentences.

Proof. It is obvious that an algorithm (running on say a Random Access Machine) can determine whether or not a triple (s, y, h) satisfies the predicate $\text{ExPrf}^*_{\text{UNION(A)}}(s, y, h)$ in say O (PolyLog $(s+y+h)$) time. Hence, Part-2 of our definition of a "Universal" 4-tape Turing Machine (see Section 3) implies that such a machine $\mathcal{M}$ will also be able check whether this triple (s, y, h) satisfies $\text{ExPrf}^*_{\text{UNION(A)}}(s, y, h)$ in O (PolyLog $(s+y+h)$) time. This immediately implies that we can employ Section 3's Turing functions to provide a straightforward encoding of $\text{ExPrf}^*_{\text{UNION(A)}}(s, y, h)$ as a Δ_0^+ formula. □

Remark A.2. Theorem A.1 is significant because ISREF(A) is an axiom system which does not recognize any of Successor, Addition or Multiplication as total functions. In this context, $\text{ExPrf}^*_{\text{UNION(A)}}(s, y, h)$ would not retain its natural meaning, if it was not encoded as a Δ_0^+ formula. Similarly, the absence of growth functions would cause Equation (27) not to retain its naturally intended meaning, if it was not encoded as a Π_1^+ sentence. This is the reason that we were very careful to insure that $\text{ExPrf}^*_{\text{UNION(A)}}(s, y, h)$ could be encoded as a Δ_0^+ formula.

Remark A.3. It is possible to establish a slightly stronger version of Theorem A.1, which states that it is not necessary to employ ISREF(A)'s Turing functions to encode $\text{ExPrf}^*_{\text{UNION(A)}}(s, y, h)$ as a Δ_0^+ formula. That is, this formula can be encoded using solely ISREF(A)'s non-Turing functions of *Integer Subtraction, Division, Maximum*(x, y), *Logarithm*(x), *Count*(x, y), *Predecessor*(x), *Root*(x, y), and *Bit*(x, y) (all of which were defined in Section 3). We omitted proving such a stronger version of Theorem A.1 in this Appendix because it was not needed by any of our main theorems in Sections 4-9, and because its actual proof is quite long. (One of the many topics discussed in our other articles [34, 36] is a description of how these more terse encodings can be designed.)

Remark A.4. The identical techniques that enabled us to encode ISREF(A)'s Group-3 axioms in this section are also applicable to encoding the Group-3 axioms for $\text{ISREF}^R(A)$ and $\text{ISREF}^*(A)$.

References

[1] A. Aho, J. Hopcroft and J. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1994.

[2] G. Boolos, The Unprovability of Consistency: An Essay on Modal Logic Cambridge University Press, 1979.

[3] A. Bezboruah and J. Shepherdson, "Gödel's Second Incompleteness Theorem for Q", J of Symbolic Logic 41 (1976), 503-512.

[4] S. Buss, "Polynomial Hierarchy and Fragments of Bounded", 17th ACM Sym on Theory of Comp (1985) pp. 285-290

[5] S. Buss, Bounded Arithmetic, Princeton Ph. D. dissertation published in Proof Theory Lecture Notes, Vol. 3, published by Bibliopolic (1986).

[6] S. Cook, "The Complexity of Theorem Proving Procedures", 3-rd ACM Symp. On Theory of Comp 1971, pp. 151-158.

[7] S. Feferman, "Arithmetization of Metamathematics in a General Setting", Fund Math 49 (1960) pp. 35-92.

[8] S. Feferman, "Finitary Inductively Presented Logics', in Proceedings of Logic Colloquium 88, North Holland Publishing House (1989) pp. 191-220.

[9] H. Friedman, "On The Consistency, Completeness and Correctness Problems", Ohio State Univ (unpublished manuscript). See also Theorem 13 of [25] for the statement of Friedman's theorem.

[10] D. Hilbert and B. Bernays, Grundlager der Mathematik Volume 1 (1934) and Volume 2 (1939), Springer.

[11] P. Hájek, "On the Interpretatability of Theories Containing Arithmetic (II)", Comm. Math. Univ. Carol. 22 (1981) pp.595-594.

[12] P. Hájek and P. Pudlák, Metamathematics of First Order Arithmetic, Springer-Verlag (1991).

[13] R. Jeroslow, "Consistency Statements in Formal Mathematics", Fundamentae Mathematicae 72 (1971) pp. 17-40.

[14] S. Kleene, "On the Notation of Ordinal Numbers", Journal of Symbolic Logic 3 (1938), pp. 150-156.

[15] J. Krájicek, "A Note on the Proofs of Falsehoods", Arch Math Logik 26 (1987) pp. 169-176.

[16] J. Krájicek and P. Pudlák, "Propositional Proof Systems, The Consistency of First-Order Theories and The Complexity of Computation", J. of Symb Logic 54 (1989) PP. 1063-1079.

[17] G. Kreisel, "A Survey of Proof Theory, Part I" in Journal of Symbolic Logic 33 (1968) pp. 321-388 and "Part II" in Proceedings of Second Scandinavian Logic Symposium (1971) North Holland Press (with Fenstad ed.), Amsterdam

[18] G. Kreisel and G. Takeuti, "Formally self-referential propositions for cut-free classical analysis and related systems", Dissertations Mathematica 118, 1974 pp. 1 -55.

[19] M. Löb, A Solution to a Problem by Leon Henkin, Journal of Symbolic Logic 20 (1955) pp. 115-118.

[20] E. Mendelson, Introduction to Mathematical Logic, Wadsworth and Brooks/Cole Mathematics Series, 1987.

[21] E. Nelson, Predicative Arithmetic, Mathematical Notes, Princeton University Press, 1986.

[22] R.Parikh, "Existence and Feasibility in Arithmetic", Journal of Symbolic Logic 36 (1971), pp.494-508.

[23] J. Paris and C. Dimitracopoulos, "A Note on the Undefinability of Cuts", Journal of Symbolic Logic 48 (1983) pp. 564-569.

[24] P. Pudlák, "Cuts Consistency Statements and Interpretations", Journal of Symbolic Logic 50 (1985) pp.423-442.

[25] P. Pudlák, "On the Lengths of Proofs of Consistency", published in Kurt Gödel Soceity Proceedings (1996).

[26] H. Rogers, Theory of Recursive Functions and Effective Compatibility, McGraw Hill 1967, see especially pp. 186-188.

[27] C. Smorynski, "The Incompleteness Theorem", in Handbook on Mathematical Logic, pp. 821-866, 1983.

[28] R. Solovay, Private Communications (April 1994) generalizing Pudlák's Proposition 2.2 from [24] to establish that no axiom system can recognize Successor, Subtraction and Division as functions, prove all the Π_1 theorems of Arithmetic and verify its own Hilbert consistency. (The proof is based essentially on introducing a Definable Cut where locally Addition and Multiplication are total functions. Then apply Pudlák's Proposition 2.2 from [24] to show that the cut can not preclude a Hilbert-proof of 0=1.)

[29] R. Statman, "Herbrand's theorem and Gentzen's Notion of a Direct Proof", in Handbook on Mathematical Logic, North Holland Publishing House (1983) pp. 897-913.

[30] G. Takeuti, "On a Generalized Logical Calculus", Japan Journal on Mathematics 23 (1953) pp. 39-96.

[31] G. Takeuti, Proof Theory, Studies in Logic Volume 81, North Holland, 1987.

[32] A. Wilkie and J. Paris, "On the Scheme of Induction for Bounded Arithmetic", Annals Pure Applied Logic (35) 1987, pp. 261-302.

[33] D. Willard, "Self-Verifying Axiom Systems", in Proceedings of the Third Kurt Gödel Symposium (1993) pp. 325-336, published in Springer-Verlag LNCS (Vol. 713).

[34] D. Willard, "The Tangibility Reflection Principle for Self-Verifying Axiom Systems", in the Proceedings of the Fifth Kurt Gödel Symposium, scheduled to be published in August (1997) in Springer-Verlag LNCS.

[35] D. Willard, "The Either-Or Theorem for Introspective Turing Axiom Systems", manuscript (1996).

[36] D. Willard, "Self-Verifying Axiom Systems, the Incompleteness Theorem and Related Reflection Principles", manuscript (1997).

State University of New York at Albany
Department of Computer Science
Albany, New York 12222 (USA)

Email = dew@cs.albany.edu.

Selected Titles in This Series

(*Continued from the front of this publication*)

8 **Simon Gindikin, Editor,** Mathematical Methods of Analysis of Biopolymer Sequences

7 **Lyle A. McGeoch and Daniel D. Sleator, Editors,** On-Line Algorithms

6 **Jacob E. Goodman, Richard Pollack, and William Steiger, Editors,** Discrete and Computational Geometry: Papers from the DIMACS Special Year

5 **Frank Hwang, Fred Roberts, and Clyde Monma, Editors,** Reliability of Computer and Communication Networks

4 **Peter Gritzmann and Bernd Sturmfels, Editors,** Applied Geometry and Discrete Mathematics, The Victor Klee Festschrift

3 **E. M. Clarke and R. P. Kurshan, Editors,** Computer-Aided Verification '90

2 **Joan Feigenbaum and Michael Merritt, Editors,** Distributed Computing and Cryptography

1 **William Cook and Paul D. Seymour, Editors,** Polyhedral Combinatorics