Logical Approaches to the Complexity of Search Problems: Proof Complexity, Quantified Propositional Calculus, and Bounded Arithmetic

by

Tsuyoshi Morioka

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

Abstract

Logical Approaches to the Complexity of Search Problems:
Proof Complexity, Quantified Propositional Calculus,
and Bounded Arithmetic

Tsuyoshi Morioka
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2005

For every binary predicate R, there is a *search problem* Q_R for finding, given x, any y such that $R(x, y)$ holds. Q_R is said to be *total* if every instance x has a solution y, that is, $(\forall x)(\exists y)R(x, y)$ holds. Total search problems are commonplace in computer science, and studying their complexity is therefore an important endeavour. In this dissertation, we present links between the complexity of solving Q_R and the difficulty of proving the totality of Q_R in the three logical formalisms: propositional calculus, quantified propositional calculus (QPC), and theories of bounded arithmetic. These links allow logical approaches to the complexity of search problems.

We show several links between the complexity of a type-2 total search problem Q_R, where R is represented by a first-order existential sentence Φ, and the lengths of proofs of the propositional translations of Φ in bounded-depth Frege systems and the Nullstellensatz proof system. In particular, we prove the first direct links between reducibilities among type-2 search problems and lengths of propositional proofs. Based on this and the results on propositional proof complexity, we obtain a number of relative separations among the so-called NP-search classes such as Polynomial Local Search ($\mathbf{PLS}$). Some of the relative separations we obtain are new.

Let H be a QPC proof system and $j \geq 1$. We define the Σ^q_j-witnessing problem for H to be: given an H-proof of a prenex Σ^q_j-formula A, and a truth assignment to the free variables of A,

find a witness for the outermost existential quantifiers of A. These witnessing problems provide a tangible link between the proof lengths in QPC and the complexity of search problems, and we consider them for various parameters. We also introduce and study the new QPC proof systems G_0^* and G_0, and prove that the Σ_1^q-witnessing problem for each is complete for NC^1-search problems. Our proof involves proving the TC^0-versions of Gentzen's midsequent theorem and Herbrand's theorem.

We introduce a second-order theory VNC^1 of bounded arithmetic, and show that the Σ_1^B-definable functions of VNC^1 are precisely the NC^1-functions. We describe simple translations of every VNC^1-proof into a family of polynomial-size G_0^*-proofs. From this and similar translation theorems for other bounded arithmetic theories, we obtain the hardness of the Σ_j^q-witnessing problem for H for various H and $j \geq 1$.

Acknowledgements

First of all, my deepest gratitude goes to my advisor Stephen Cook, who was the reason I came to the University of Toronto. It has been always inspiring to witness firsthand how a great mind like his operates. Under his guidance, I always felt that I was free to pursue my own curiosity, knowing that he would never let me go astray. I will miss discussions with him, not only on research but also on our shared interests in music.

I thank the members of my Ph.D committee: Stephen Cook, Charles Rackoff, Alasdair Urquhart, and Toniann Pitassi. I'm grateful for their helpful comments, suggestions, and questions that they made in the various checkpoints in my Ph.D process. They were such a pleasant and supportive committee.

Also I would like to thank Jan Krajíček, who did the external appraisal. His comments and questions helped me improve the quality of this thesis.

I am lucky to have been surrounded by so many grad student colleagues, and I thank them for their friendship and support. They are too many to be named here, but I'd like to thank my close academic colleagues Josh Buresh-Oppenheim, Antonina Kolokolova, Alan Skelley, and Phuong The Nguyen.

I have a lot of good memories of my life in Toronto, and many of my fondest memories are of the times that I spent with Ping-Ying Angel Ho. I thank her for lots of laughs she gave me and for being a very good reason for me not to spend too much time at my desk.

Finally, I would like to thank my parents, Takeshi and Fumiko Morioka. Throughout my life they encouraged me follow my own interests, and they respected and supported whatever decisions I made on the way. Life is full of ups and downs, but their faith in me gives me strength to continue on whatever path I have chosen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations and Background

Search Problems and Logic

Complexity theory is the study of the hardness of various problems, measured in terms of the required amount of resources (time, space, etc.) on various models of computation. Three decades of fruitful research followed Cook's introduction of NP-completeness in 1971 [Coo71], and since then numerous computational problems have been shown to be complete for, or at least classified into, various complexity classes. In particular, a great number of combinatorial problems that arise naturally in practical settings are shown to be NP-complete.

An interesting aspect of complexity theory is its almost exclusive focus on *decision problems*, or the problems of deciding whether the input has a certain property, although problems often arise naturally as *search problems*, or the problems of finding an object with a certain property. As a result, search problems have been commonly studied indirectly using their equivalent decision counterparts. For example, the complexity of finding a 3-colouring of a graph (if one exists) is studied indirectly via the problem of deciding if the input graph is 3-colourable. A justification for this indirect approach has been that these search and decision problems are polynomially equivalent, i.e., they are polynomial-time Turing reducible to each

other. Note that functions are search problems every instance of which has a unique solution.

However, the past research on the complexity of search problems, in particular *total search problems*, has shown that this indirect approach is not completely satisfactory for two reasons. Here a search problem is said to be total if every instance of it has a solution. For example, the 3-colouring search problem above is not total.

First, the complexity of a total search problem depends on subtle structures of the problem that are lost when they are transformed to decision problems. For example, Krentel showed in [Kre88] that the problem of computing the cost of an optimal Traveling Salesman tour of a given instance is complete for $\mathbf{FP}^{\mathbf{NP}}$ while the problem of computing the size of a maximum clique of a graph is complete for $\mathbf{FP}^{\mathbf{NP}}[O(\log n)]$. (See Section 2.1.3 for the definition of these classes.) Since Krentel proves that $\mathbf{FP}^{\mathbf{NP}}$ properly contains $\mathbf{FP}^{\mathbf{NP}}[O(\log n)]$ unless the polynomial-time hierarchy $\mathbf{PH}$ collapses, the Traveling Salesman search problem apparently is harder than the clique search problem, although their decision counterparts are both complete for $\mathbf{NP}$.

The second reason why we need a direct approach to search problems is simple: some total search problems may not have polynomially-equivalent decision counterparts. Evidence in this direction is obtained by Beame et. al [BCE$^+$98], who demonstrate specific relativized search problems that are not polynomially-equivalent to any decision problem. The result of Beame et. al implies that the complete problems for classes such as Polynomial Local Search ($\mathbf{PLS}$) of [JPY88], and Polynomial Pigeonhole Principle ($\mathbf{PPP}$) and Polynomial Parity Argument ($\mathbf{PPA}$) of [Pap94b] are unlikely to have equivalent decision problems. We will discuss these classes in more depth below, but for the moment it suffices to note that they contain a number of natural, practical problems that arise in computer science and mathematics. Examples include optimization problems such as the problem of finding a Traveling Salesman tour that is locally optimal with respect to the 2-OPT heuristic, and problems in game theory such as finding a Nash equilibrium given payoff matrices for two players.

The study of total search problems is a fruitful research area that is in need of new ap-

proaches and techniques, and the way total search problems is defined gives rise to the following logical approach to its complexity. If Q is a total search problem, then the statement of Q's totality 'every instance x of Q has a solution y' is a true assertion, and therefore it can be formulated and proved in suitable formal systems. Now we can ask the following question: in a given formal system, how hard is it to prove the totality of Q?

The main theme of this dissertation is to link the complexity of total search problems Q and the hardness of proving its totality in the following three logical formalisms: propositional calculus, quantified propositional calculus, and theories of bounded arithmetic.

Proof Complexity

Proofs are fundamental objects in mathematics, and proof complexity is the quantitative study of proofs. The proof complexity of a statement is measured as the length of the shortest proof of the statement in a given proof system. In the seemingly simple case of propositional logic, the study of proof complexity is already highly nontrivial: in the seminal work of Cook and Reckhow [CR77], it is shown that there exists a super proof system, i.e., a proof system in which every tautology has a polynomial-size proof, if and only if $\mathbf{NP} = \mathbf{coNP}$. Hence, the question of the existence of a super proof system is a restatement of a fundamental open problem of complexity theory. Proof complexity research is also important because it sheds light on the efficiency of various automatic theorem provers and algorithms for the propositional satisfiability problem.

A number of proof systems have been introduced and studied, and in this dissertation we will work with Gentzen's sequent calculus PK, the bounded-depth version of PK, and the algebraic proof system Nullstellensatz. More information on these systems as well as proof complexity in general is found in [BP98, Kra95, Urq95, Bus98c].

In a certain context, a PK-proof can be thought of as a branching program (or a decision tree if the proof is tree-like) for solving a search problem, and this correspondence between propositional proofs and computations has been successfully used in obtaining lower bounds

for various subsystems of PK. This correspondence is most clearly exploited in [Rii01] with respect to tree-like resolution and in [Kra01] for the system $R^*(\log)$ (also see [Kra95]). For bounded-depth PK, this idea is carried out in more sophisticated ways in [PBI93, KPW95, BP96].

Part of our work explores this proof-computation correspondence in the context of type-2 computation, and as a result we are able to deduce via the existing results in proof complexity research a number of relative separations among search classes. In particular, our work is the first to explicitly link reducibilities among search problems with relative hardness of proving the totality of the search problems in propositional proof systems. We will say more on this below.

Bounded Arithmetic

Theories of bounded arithmetic are logical theories of arithmetic with bounds on their reasoning power. The most intensely studied are Buss's first-order theories [Bus86, Bus98a]

$$S_2^1 \subseteq T_2^1 \subseteq S_2^2 \subseteq T_2^2 \subseteq \dots$$

and $S_2 = \bigcup_{i \geq 1} S_2^i$. The theory T_2^i is essentially the theory of arithmetic with induction allowed only on $\mathbf{\Sigma}_i^p$-predicates, and S_2^i is obtained by restricting T_2^i so that, instead of induction, only length induction is allowed.

The main motivation for studying these theories is their close connections to complexity classes, and the notion of *definability* provides a particularly strong link. A search problem is Σ_i^b-definable in theory T if and only if its totality is represented by a $\forall\Sigma_i^b$-sentence and T proves this sentence. For example, a search problem is Σ_1^b-definable in S_2^1 if and only if it is solvable in polynomial-time [Bus86], and a search problem is similarly definable in T_2^1 if and only if it is many-one reducible to a problem in $\mathbf{PLS}$ [BK94, CK98, Mor01]. Also definable search problems are a crucial tool in obtaining a number of important results for bounded arithmetic, such as conservativity of theories, separations of relativized theories, and unprovability of certain combinatorial principles. From a complexity theorist's point of view, bounded arithmetic

allows one to study complexity classes using techniques of mathematical logic, which include proof-theoretic methods and model-theoretic methods. In particular, Chiari and Krajíček studied various total search problems in the context of bounded arithmetic [CK98].

It turns out that Buss's theories are also connected to proof complexity in the following way: every bounded theorem Φ of the relativized theory $S_2(\mathcal{L})$ translates into a family $Trans(\Phi)$ of tautologies with quasipolynomial-size bounded-depth PK-proofs [PW85, Kra95]. Thus, bounded-depth PK is a nonuniform analogue of $S_2(\mathcal{L})$, and the unprovability of Φ follows from an exponential lower bound on the proof size of $Trans(\Phi)$ in bounded-depth PK.

Quantified Propositional Calculus

Quantified propositional calculus (QPC) is obtained by introducing quantifiers into propositional calculus, where $(\exists x)A(x)$ is equivalent to $A(\mathsf{T}) \vee (A(\mathsf{F})$ and $(\forall x)A(x)$ is equivalent to $A(\mathsf{T}) \wedge A(\mathsf{F})$. Unlike first-order logic, which is strictly more expressive than propositional calculus, QPC is no more expressive than propositional calculus in the sense that, for every QPC formula A, there is a propositional formula that is logically equivalent to A.

However, in QPC we can express certain properties in a more compact way than in propositional calculus. For example, assuming an appropriate uniformity condition, every $\mathbf{PSPACE}$-predicate is represented by a polynomial-size family of QPC formulas [Pap94a], while no $\mathbf{PSPACE}$-complete predicate is representable by a polynomial-size family of propositional formulas. The latter fact follows from the characterization of $\mathbf{NC}^1$ by polynomial-size families of propositional formulas [BIS90] and the fact that $\mathbf{NC}^1 \subsetneq \mathbf{PSPACE}$.

Thus, an advantage of QPC is that it has enough expressive power to represent everything in $\mathbf{PSPACE}$ in a concise way witout losing the syntactic and semantic simplicity of propositional calculus. The following is an example QPC formula, which asserts that a QPC formula $A(\vec{p})$ has a truth value x:

$$(\exists x)[(x \wedge A(\vec{p})) \vee (\neg x \wedge \neg A(\vec{p}))]$$

QPC formulas are also known as QBF (Quantified Boolean Formulas), and efficiency of

decision procedures for the satisfiability of QBFs is an important issue in various research areas such as formal verification, planning, reasoning about knowledge, and there has been much effort in designing and implementing QBF solvers that can be used in practice [BST03]. The study of QPC proof complexity is relevant to such effort.

Since the set of valid QPC formulas is complete for **PSPACE** [Pap94a], the proof complexity of QPC is closely related to **PSPACE**. Krajíček and Pudlák [KP90, Kra95] introduced the Gentzen-style sequent calculus proof system G for QPC, together with the hierarchy of fragments

$$G_1^* \leq_p G_1 \leq_p G_2^* \leq_p G_2 \leq_p \ldots$$

where $\leq_p$ denotes p-simulation. The results of Krajíček and Pudlák demonstrate that these QPC proof systems are nonuniform analogue of Buss's theories. Aside from Pollett's work that introduces the QPC proof systems L_i^* that correspond to the theories R_2^i [Pol97], we are not aware of any other work on the QPC proof complexity and/or its connection to complexity theory and bounded arithmetic.

1.2 Our Work

This dissertation is concerned with search problems, proof complexity, quantified propositional calculus, and bounded arithmetic, and how they are connected with each other. Figure 1.1 graphically depicts the relationship among these four research areas. This dissertation is organized into two Parts: Part I is about the triangle on the right in Figure 1.1, that is, it is on the connections between search problems and propositional proof complexity, both of which are in turn related to bounded arithmetic. Part II is on the left triangle, namely, the connections among QPC, search problems, and bounded arithmetic. In Figure 1.1, the thick arrows linking search problem with propositional proof complexity and quantified propositional calculus indicate that we are particularly interested in these connections.

The organization of this dissertation is as follows. Part I consists of Chapters 3 and 4. Part

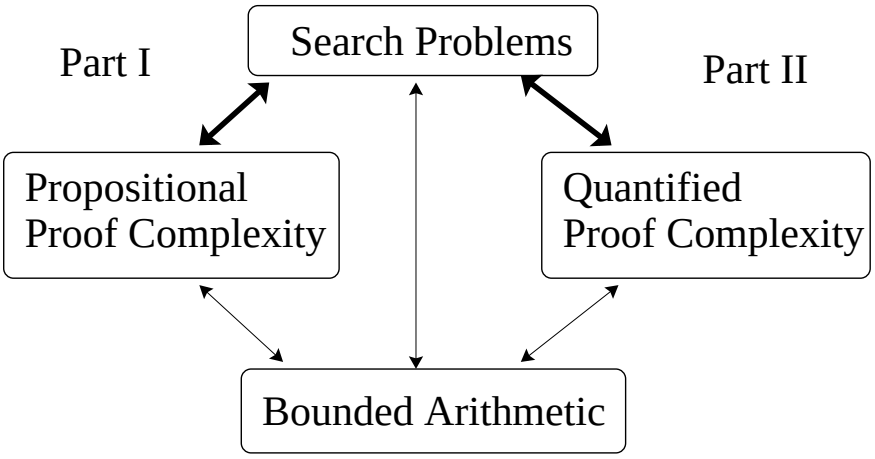

Figure 1.1: Overview of this dissertation. The thick arrows indicate that we are particularly interested in these connections.

II begins with Chapter 5 and end with Chapter 9. Chapter 2 is a preliminary chapter in which we present a number of basic definitions and facts that we use in the subsequent chapters. The following are brief expositions on the contents of each Part.

1.2.1 Part I

Most of the results in Chapter 3 are obtained in our joint work with J. Buresh-Oppenheim, and the results of Chapter 4 are extensions of our M.Sc results [Mor01]. These have been reported in a preliminary form in our joint paper [BOM04].

In the papers [JPY88, Pap94b], total search problems are classified according to the combinatorial principle in the finite domain that guarantees the totality of the problems. These classes contain numerous natural problems, some of which are complete. For example, *Polynomial Local Search* ($\mathbf{PLS}$), which is the class of problems efficiently solvable by local search heuristics, is characterized by the iteration principle "every finite dag has a sink"; and *Polynomial Pigeonhole Principle* ($\mathbf{PPP}$), which has relevance to cryptographic hash functions, corresponds to the pigeonhole principle "there is no injective mapping from $[n+1]$ to $[n]$." The class *Polynomial Parity Argument* ($\mathbf{PPA}$) is defined by the parity principle "there is no perfect

matching in an odd-sized graph" and contains the problems of finding various economic equilibria. And its variants **PPAD** and **PPADS** are defined in a similar manner (**PPAD** was called **PSK** in [Pap94b], and it is given this name in [BCE+98]).

Beame, Cook, Edmonds, Impagliazzo, and Pitassi [BCE+98] reformulate the search classes in terms of type-2 search problems, or search problems whose input contains not only numbers and strings (type-0 objects) but also functions and relations (type-1 objects) that are accessed as oracles. This type-2 approach results in much cleaner definitions of the original type-1 search classes: each class essentially becomes a collection of the type-1 instances of a single type-2 problem, and hence the relationship among these classes can be studied through the corresponding type-2 search problems. In many cases we can obtain unconditional separations of type-2 search problems, which imply oracle separations of the corresponding type-1 search classes. Many such relative containments and separations among the above five search classes are obtained via the type-2 methods in [BCE+98, Mor01].

Since an unrelativized separation of any two of the above search classes implies $\mathbf{P} \neq \mathbf{NP}$, such relative separations are currently the best results we can hope for. However, we would like to argue that this particular type of oracle result is more meaningful than your garden-variety oracle result, whose relevance has been repeatedly brought into question (starting with [BGS75]). This intuition comes from two sources: (i) Each of these type-2 separations implies that the *generic oracle* separates the corresponding type-1 classes ([CIY97]). While generic oracles ([BI87]) are not infallible ([FS88]), they capture the intuition that an "arbitrary" oracle should not affect the equality of two classes; (ii) Oracle separations of complexity classes are usually obtained by exploiting the difference in the ways these classes access the oracle, and hence these results are better understood as separations of oracle access methods and not of complexity classes [HCC+92]. This way we can make more sense of 'oracle separations' between classes that are actually equal, such as **PSPACE** and **IP** [FS88]. On the other hand, since all the relativized search classes in this dissertation access an oracle in the same way (via deterministic polynomial-time machines), oracle separations of those classes may better reflect

the unrelativized world.

In Chapter 3, we extend the framework of [BCE$^+$98] into the following systematic method of defining type-2 search problems from combinatorial principles. Let Φ be a first-order existential sentence over an arbitrary language such that Φ holds in every finite structure, and define Q_Φ to be the corresponding type-2 search problem of finding a witness to Φ in a finite structure given as the input. For example, the type-2 problem **PIGEON**, which characterizes the class $\mathbf{PPP}$, arises from the sentence

$$(\forall x)[\alpha(x) \neq 0] \supset (\exists x, y)[x \neq y \wedge \alpha(x) = \alpha(y)],$$

which states that, if 0 is not in the image of α, then there must exist two elements that are mapped to the same element by α; this is the injective pigeonhole principle, which holds in every finite structure.

Formulated as above, it is clear that studying the complexity of a type-2 search problem Q_Φ amounts to the study of the 'computational power' of the combinatorial principle Φ, which is an interesting mathematical endeavour in its own right. In addition to the pigeonhole principle above, we present the $\exists$-sentences that give rise to the type-2 problems $\mathbf{LONELY}$, $\mathbf{OntoPIGEON}$, and $\mathbf{ITERATION}$, which characterize the classes $\mathbf{PPA}$, $\mathbf{PPAD}$, and $\mathbf{PLS}$, respectively. We also formulate a new type-2 problem $\mathbf{WeakPIGEON}$ which corresponds to the weak pigeonhole principle.

We describe how to translate an $\exists$-sentence Φ into a family $Trans(Q_\Phi)$ of tautologies of depth 2. This is the translation due to Paris and Wilkie [PW85] which translates every bounded theorem of the relativized theory $S_2(\mathcal{L})$ into a family of tautologies with quasipolynomial-size bounded-depth PK-proofs.

The main results of Chapter 3 connect the complexity of Q_Φ and the proof complexity of $Trans(Q_\Phi)$. We first obtain two such results for bounded-depth PK. The first and simpler result (Theorem 3.21) is the following: if Q_Φ is solvable in polynomial-time, then $Trans(Q_\Phi)$ has quasipolynomial-size PK-proofs of depth 2. This result formalizes the intuitive idea that

an algorithm that solves Q_Φ is a highly constructive proof of Φ, and this is done by showing how to turn an algorithm for Q_Φ into a depth-2 PK-proof of $Trans(Q_\Phi)$. This result could be used to obtain a lower bound on the time complexity of Q_Φ via a lower bound on $Trans(Q_\Phi)$ in depth-2 PK; however, such a proof complexity lower bound is usually much harder to obtain than the complexity lower bound for Q_Φ. This may be more useful in obtaining upper bound on the proof lengths of $Trans(Q_\Phi)$: it now suffices to demonstrate an algorithm that solves Q_Φ within an appropriate time bound.

The second and more important result for bounded-depth PK is as follows: if $Q_\Phi \leq_m Q_\Psi$, then $Trans(Q_\Phi)$ has quasipolynomial-size depth-3 PK-proofs in which substitution instances of $Trans(Q_\Psi)$ are allowed as nonlogical axioms. Note that the conclusion essentially states that the proof complexity of $Trans(Q_\Phi)$ in depth-3 PK does not exceed that of $Trans(Q_\Psi)$, and the proof of this result is a formalization of the intuitive idea that a many-one reduction from Q_Φ to Q_Ψ is itself a constructive proof that Ψ implies Φ. We first prove this result with a technical assumption that Q_Ψ has the *instance extension property* (Theorem 3.23); however, we prove that this assumption is not really needed (Theorem 3.24). The instance extension property is introduced and discussed in Section 3.3.

Similarly, we prove two results for Nullstellensatz. The first result converts a polynomial-time algorithm for Q_Φ into polylogarithmic-degree Nullstellensatz refutations of $\neg Trans(Q_\Psi)$ (Theorem 3.28), and the second result constructs Nullstellensatz derivations of $\neg Trans(Q_\Phi)$ from a substitution instance of $\neg Trans(Q_\Psi)$ (Theorem 3.30). Again we prove the second result with the assumption of the instance extension property of Q_Ψ; however, for Nullstellensatz, we do not know whether this assumption can be removed from Theorem 3.30. Note that the proof of Theorem 3.30 is a generalization of a technique that Beame et. al used in [BCE+98] to separate $\mathbf{LONELY}$ and $\mathbf{PIGEON}$ using a Nullstellensatz degree lower bound.

As corollaries to the above results, we obtain relative separations of search classes that were previously unknown, such as $\mathbf{PLS}^G \not\subseteq \mathbf{PPA}^G$ with G a generic oracle. Our result generalizes the proof techniques of Beame et al. and hence it provides alternative proofs for most of

their results via the proof complexity separations. Moreover, since the combinatorial principle characterizing $\mathbf{PPA}$ has low-complexity proofs in Nullstellensatz, it follows that the totality of every $\mathbf{PPA}$ problem has a low-complexity proof. This is interesting because $\mathbf{PPA}$ contains the witnessing problems for the fixed point theorems of Brouwer, Nash, and Kakutani [Pap94b].

There has been much work (for example, [LTT89, Aar04]) on the efficiency of local search, whose primary goal is to obtain lower bounds on the number of times a local search heuristics has to be invoked. In our context, this amounts to obtaining a lower bound on the number of times f has to be accessed for solving the type-2 problem $\mathbf{ITERATION}(f, 1^n)$, and, by Theorems 3.28, we can obtain such a lower bound from the degree lower bound for the iteration principle (i.e., the housesitting principle) in Nullstellensatz [CEI96, Bus98c]. We do not yet know how such a lower bound compares with the known lower bounds.

The Nash problem is formulated as follows. Given two integer matrices that represent payoffs for a two-player game, find a Nash equilibrium. It is known to be in $\mathbf{PPAD}$ [Pap94b], but not known to be complete for any class. Papadimitriou calls the Nash problem 'a most fundamental computational problem whose complexity is wide open' [Pap01], and there have been attempts to obtain good bounds on this problem; for example, see [SvS04]. From our results, a new approach to this problem emerges: namely, formulating the totality of the Nash problem as a family of tautologies, and show a lower bound on the size of their depth-2 PK-proofs. By our Theorem 3.21, such a proof complexity lower bound translates into a lower bound on the deterministic time complexity of Nash. Note that, since Nash is in $\mathbf{PPAD}$ and since $\mathbf{OntoPIGEON}$ is easy for Nullstellensatz, the totality of Nash has low-degree proofs in Nullstellensatz.

Aside from one type-2 separation that Beame et. al obtain via Nullstellensatz degree lower bounds [BCE$^+$98], we are not aware of any work explicitly linking reductions among search problems with the lengths of propositional proofs. However, Buss in [Bus03] obtains upper bounds and lower bounds on the proof complexity of various combinatorial principles by describing transformations of a proof of one principle into that of another principle. His proof

transformations amount to simple reductions between the search problems corresponding to the principles.

Local search is a widely used approach to various optimization problems. The type-2 problem **ITERATION** captures the power of local search in the sense that, if type-1 search problem Q is many-one reducible to **ITERATION**, then Q can be formulated as a local search problem, and if $Q \not\leq_m$ **ITERATION**, then there is no efficient local search heuristic for Q. In Chapter 4, we present a sufficient condition for a type-2 search problem to be nonreducible to **ITERATION**, which is useful in recognizing search problems for which local search is unlikely to be useful.

All the known type-2 separations of the form $Q \not\leq_m$ **ITERATION** in [Mor01] and Chapter 3 follow from this as corollaries. This 'separation criterion' is a slightly stronger variant of Riis's 'independence criterion' for the relativized theory $S^2_2(L)$ of bounded arithmetic [Rii93], and it also generalizes the main result of our M.Sc. thesis [Mor01] (Theorem 3.8) and other relative separations involving the I-problem (the iteration problem) in [CK98].

More formally, the main result of Chapter 4 is stated as follows: if Φ is a combinatorial principle that does not involve the ordering relation, and if Φ fails in an infinite structure, then Q_Φ is not many-one reducible to **ITERATION**. Since all the type-2 search problems we introduced in Chapter 3 satisfy the conditions of the above theorem, it follows that none of them is many-one reducible to **ITERATION**, and we obtain the oracle separations of the corresponding search classes, suggesting that efficient local search heuristics do not exist for the problems in these search classes.

We point out that Krajíček has a result (Theorem 4.3) that is similar to but incomparable with ours. Krajíček's result is used to obtain an alternative proof of Riis's independence criterion [Kra95], and we show that our result implies it in a similar way.

1.2.2 Part II

The main topics of Chapters in Part II are QPC and its link to bounded arithmetic via definable search problems. Some of the results of this Part have appeared in a preliminary form in our joint paper with Stephen Cook [CM04]. This dissertation contains a number of results that we have obtained since then, such as the ones in Sections 6.3, 7.4, and 8.3. The following are brief descriptions of each Chapter and the results that appear in it.

Chapter 5 is an exposition on QPC and its proof complexity. It begins with the basic definitions of QPC and the QPC sequent calculus systems G, G_i, and G_i^* of Krajíček and Pudlák. Here G_i is G restricted so that only Σ_i^q or Π_i^q formulas can occur in proofs, where a formula is in Σ_i^q if it has a prenex form with at most $i-1$ alternations of quantifiers, beginning with $\exists$, and dually for Π_i^q. G_i^* is G_i restricted to tree-like proofs. The systems are related to the polynomial hierarchy (**PH**) in that the decision problem for truth of Σ_i^q sentences is complete for the level $\mathbf{\Sigma_i^P}$ of **PH**, and similarly for Π_i^q and $\mathbf{\Pi_i^P}$.

We modify the definitions of G_i and G_i^* by allowing arbitrary QPC formulas in proofs, but restricting cut formulas to be Σ_i^q and restricting the target formulas in $\exists$-right and $\forall$-left rules to be quantifier-free. Since the modifications result in two versions of G_i and G_i^*, ones under the original definitions of Krajíček and Pudlák and the others under our modified definitions, the reader may become concerned that this results in confusion. We argue, in Section 5.1, (i) that this does not result in any confusion, and (ii) that the modification enables us to ask interesting questions that did not arise under the original definitions. As a justification for (i), we prove that the modified systems are polynomially equivalent to the original for proving $\Sigma_i^q \cup \Pi_i^q$ formulas.

A justification for (ii) and also a major advantage of the modification is that all G_i and G_i^* for $i \geq 0$ are complete systems for proving all valid QPC formulas. In particular, we obtain G_0 and G_0^* as new and interesting QPC systems, which are polynomially equivalent to Frege systems when proving quantifier-free theorems. In fact, we prove that every valid QPC formula A has a G_0^*-proof of size doubly exponential in $|A|$, and if A is a Σ_i^q-sentence then G_i^* has a

proof of A of size single exponential in $|A|$ (Theorem 5.9). Also this modification allows us to ask questions about the complexity of the QPC witnessing problems for various parameter values (see below), many of which couldn't be asked under the original definitions. We also argue that the modified systems are easy to work with: for example, in the modified systems, the quantifier introduction rules always increase quantifier complexity, while this is not the case in the original systems.

In intuitive terms, the restriction on the complexity of cut formulas amounts to a restriction on the complexity of lemmas that are allowed in proofs, and the difference between tree-like proofs and dag proofs amounts to what kind of induction can be simulated in the system. It turns out that simulation of induction requires dag proofs while tree-like proofs can simulate only length induction. We give more explanations in Section 5.2.

In Section 5.3, we prove several theorems on the QPC proof complexity. One useful result is that every G_i^*-proof can be converted, in polynomial-time, into another G_i^*-proof in which every cut formula is prenex Σ_i^q. This result turns out to be crucial for obtaining the completeness of the Σ_{i+1}^q-witnessing problem for G_i^* in Section 6.3. Surprisingly, the question whether the cut formulas of G_i can be restricted to prenex Σ_i^q without decreasing the power of G_i is connected to the question whether $O(\log n)$ witness queries are as powerful as polynomially many witness queries. A more formal statement is proven in Section 6.3 (Theorem 6.10).

In Section 5.4, we show that much of the proof theory of the QPC sequent calculus can be carried out in the complexity class (**Dlogtime**-uniform) $\mathbf{TC}^0$, which apparently is much smaller than $\mathbf{P}$. In particular, we show that extracting a propositional 'Herbrand disjunction' of a G_0-proof π is in $\mathbf{TC}^0$. Using this fact, we prove the $\mathbf{TC}^0$-version of Gentzen's midsequent theorem for G_0^*. Note that it is not clear if Gentzen's original proof of his midsequent theorem can be carried out in a class below $\mathbf{P}$. We use these facts to prove that G_0 and G_0^* are polynomially equivalent with respect to prenex Σ_1^q-formulas. Note that a similar p-equivalence between G_1 and G_1^* implies that $\mathbf{PLS}$ is contained in $\mathbf{FP}$, which is not believed to be true.

Krajíček and Pudlák [KP90] show that, for every $i \geq 1$, every Σ_i^q-theorem of T_2^i translates

into a family of Σ_i^q-formulas with polynomial-size G_i-proofs, and Krajíček [Kra95] proves a similar statement for S_2^i and G_i^*. Because of the peculiarities of the translation used, the proofs of these statements do not generalize to the translation of bounded theorems with more than $i-1$ quantifier alternations. We point out that, by going through the second-order theories $\mathbf{V}^i$ and $\mathbf{TV}^i$ of Cook that we present in Chapter 7, the above theorems nonetheless generalize to the translation of *any* bounded theorems of S_2^i and T_2^i.

In Chapter 6, we introduce a new kind of total search problem, the QPC witnessing problems. The QPC witnessing problems provide a tangible link between the complexity of search problems and the length of proofs in QPC. We obtain a number of results on the complexity of the QPC witnessing problems, and they are summarized in Chapter 9. Also see Table 9.2 on page 173. Some of the results are derived based on the close connection between QPC and bounded arithmetic [KP90, Kra95]. However, some of our results do not have any counterparts in bounded arithmetic; in fact, we will show in Chapter 8 that the connection between QPC and bounded arithmetic breaks down for proving formulas with a large number of quantifier alternations.

For $j \geq 1$, we define the Σ_j^q-witnessing problem for a QPC system H to be: given a prenex Σ_j^q-formula A, an H-proof of A, and a truth assignment to the free variables of A, find a witness for the outermost existential quantifiers in A. This is denoted as $Witness[H, \Sigma_j^q]$. Note that, for every H, $Witness[H, \Sigma_j^q]$ is trivially solvable in polynomial-time using an $\mathbf{\Sigma}_j^p$-oracle. Thus, the interesting question is how much the presence of an H-proof of A brings down the complexity of witnessing A. For example, consider the Σ_1^q-witnessing problem for a QPC proof system H. Regardless of H, we can deduce the trivial upper bound of $\mathbf{FP}^{\mathbf{NP}}$. But our results tell us more:

- $Witness[G_0^*, \Sigma_1^q]$ and $Witness[G_0, \Sigma_1^q]$ are complete for $\mathbf{FNC}^1$. (Theorem 6.5)
- $Witness[G_1^*, \Sigma_1^q]$ is complete for $\mathbf{FP}$. (Theorem 6.2)
- $Witness[G_1, \Sigma_1^q]$ is complete for $C(\mathbf{PLS})$. (Theorem 6.2)

Thus, when $H \in \{G_0^*, G_0, G_1^*, G_1\}$, the presence of an H-proof in the input dramatically decrease the complexity of witnessing the given Σ_1^q-formula A, and the amount of the decrease is larger if H is a weaker proof system. This shows that H-proofs of Σ_1^q-formulas contain information on how to construct a witness for the endformula A, and the weaker the system the more constructive the proofs.

In Section 6.1, we prove that, for $i \geq 1$, $Witness[G_i^*, \Sigma_i^q]$ and $Witness[G_i, \Sigma_i^q]$ are complete for $\mathbf{FP}^{\mathbf{\Sigma}_{i-1}^p}$ and $C(\mathbf{PLS})^{\mathbf{\Sigma}_{i-1}^p}$, respectively. In the proof, we make use of the results of Krajíček and Pudlák on the provability of the reflection principles for the QPC systems [KP90] and witnessing theorems for bounded arithmetic [Bus86, BK94].

Next, in Section 6.2, we show that both $Witness[G_0^*, \Sigma_1^q]$ and $Witness[G_0, \Sigma_1^q]$ are complete for the class **FNC**^{1} of (**Dlogtime**-uniform) **NC**^{1}-search problems. Our proof uses Buss's **NC**^{1} algorithm for the Boolean Formula Value Problem [Bus87, Bus93], as well as the fact that necessary manipulations of G_0-proofs can be done in **TC**^{0}.

Finally, in Section 6.3, we consider the complexity of $Witness[H_i, \Sigma_k^q]$ for $i \geq 0$ and $k > i$, where H_i is either G_i^* or G_i. More specifically, we show that, for every $i \geq 1$, $Witness[G_i^*, \Sigma_{i+1}^q]$ is complete for $\mathbf{FP}^{\mathbf{\Sigma}_i^p}[wit, O(\log n)]$, which matches the result on the Σ_{i+1}^b-definable search problems of S_2^i. We also show that, for every $i \geq 0$ and $k \geq i+2$, $Witness[G_i, \Sigma_k^q]$ is in $\mathbf{FP}^{\mathbf{\Sigma}_{k-1}^p}[wit, O(\log n)]$. We prove a hardness result for these witnessing problems in Chapter 8, but the hardness we obtain does not match this upper bound.

In Section 6.4, we consider propositional systems which allow threshold gates, and extend them to systems with propositional quantifiers. In particular we define for $d = 1, 2, \ldots$ the system $TG_0(d)$, which allows cuts only on quantifier-free formulas, and in which all quantifier-free formulas in a proof have depth $\leq d$. We prove that the $T\Sigma_1^q$-witnessing problem for $TG_0(d)$ can be solved by a **TC**^{0}-function, and conversely every **TC**^{0}-function is reducible to such a witnessing problem for some d. For the results in this Section, it is important that **TC**^{0} contains basic parsing operations for QPC sequent calculus, which we prove in Section 5.4.

Buss's first-order theories S_2^i and T_2^i of bounded arithmetic are closely related to $\mathbf{PH}$, and

they have been the objects of intense study since their introduction in Buss's 1985 dissertation published as [Bus86]. Buss's theories are preceded by the first-order theory $\mathbf{I\Delta}_0 + \Omega_1$ of Paris and Wilkie [PW81, WP87] and the equational theories PV of Cook [Coo75] for $\mathbf{P}$ and PRA of Dowd [Dow79], and followed by a number of new theories of bounded arithmetic corresponding to various complexity classes, such as Arai's **AID** [Ara00] for $\mathbf{NC}^1$, and theories for $\mathbf{NC}$, $\mathbf{NC}^1$, logspace, and nondeterministic logspace by Clote and Takeuti [CT92], among others. A notable aspect of these developments is a great variety of styles in which these theories are defined: there are equational, first-order, and second-order theories characterizing complexity classes in different ways. As a result of these developments, our knowledge of the connections between complexity classes and logic has greatly increased, but it is nontrivial to compare these theories because of their different flavours.

Based on Zambella's work on second-order bounded arithmetic theories [Zam96], Cook [Coo02] has introduced second-order theories $\mathbf{V}^0$ and $\mathbf{V}^1$ corresponding to $\mathbf{AC^0}$ and $\mathbf{P}$, respectively, and his work led to a framework in which second-order theories characterizing various complexity classes are developed in a unified way [CK03, NC04, Coo04]. The relatively simple syntax of this framework results in a simplification of both the description of the theories and the proofs of their properties, such as their connections to QPC which will be discussed in Chapter 8. Moreover, the unified syntax of these theories makes it easy to compare them.

The main result of Chapter 7 is the introduction of the second-order theory $\mathbf{VNC}^1$ for $\mathbf{NC}^1$, which is inspired by Arai's **AID**. We obtain $\mathbf{VNC}^1$ by augmenting the base theory $\mathbf{V}^0$ with a scheme Σ^B_0-$TreeRec$ for a tree recursion, and prove that a function is Σ^B_1-definable in $\mathbf{VNC}^1$ if and only if it is an $\mathbf{NC}^1$-function. We begin Chapter 7 with the presentation of Cook's theories $\mathbf{V}^0$, $\mathbf{V}^i$, and $\mathbf{TV}^i$ and their syntax.

In Section 7.4, we present alternative proofs for Pollett's result [Pol99] (Theorem 2.24 in this dissertation) characterizing the Σ^b_k-definable search problems of S^i_2 and T^i_2 for $i \geq 1$ and $k \geq i + 2$. Our proofs are simple and presented in a more general setting. As a result, we

obtain new characterizations of the Σ_k^B-definable search problems for all $k \geq 2$ of second-order theories $\mathbf{V}^0$, $\mathbf{VTC}^0$ of [NC04], $\mathbf{VNC}^1$, $\mathbf{V}^1$-Horn of [CK03], and $\mathbf{TV^0}$ of [Coo04] (also in Chapter 7 of this dissertation). In fact, this characterization applies to every theory T with $\mathbf{V}^0 \subseteq T \subseteq \mathbf{TV^0}$, and it has an interesting consequence on the provability of reflection principles in bounded arithmetic, which will be discussed in Chapter 8.

In Chapter 8 we discuss the QPC translation of Cook-style second-order theories. We begin with the presentation of a translation [Coo02] of second-order bounded arithmetic formulas into polynomial-size QPC formulas. Using this translation, we prove that any bounded theorem of $\mathbf{V}^i$ and $\mathbf{TV}^i$ translates into families of valid QPC formulas with polynomial-size G_i^*- and G_i-proofs, respectively. We also prove a translation of bounded theorems of $\mathbf{VNC}^1$ into polynomial-size G_0^*-proofs. These translation theorems generalize similar results for S_2^i and T_2^i by Krajíček and Pudlák [KP90], since our translation applies to any bounded theorem as opposed to Σ_i^B-theorems. This generality stems from the simple syntax of Cook's second-order framework.

Below are some of the results we obtain based on these translation theorems: for every $i \geq 1$ and $k \geq i+2$,

- $Witness[G_i, \Sigma_k^q]$ is hard for $\mathbf{FP}^{\mathbf{\Sigma}_{k-1}^p}[wit, O(1)]$;
- $Witness[G_i, \Sigma_k^q]$ cannot be complete for $\mathbf{FP}^{\mathbf{\Sigma}_{k-1}^p}[wit, O(1)]$ unless $\mathbf{PH}$ collapses; and
- the k-reflection principle for G_i is not provable in T_2^i unless $\mathbf{PH}$ collapses.

The results above show that the close relationship between T_2^i and G_i break down with respect to proving formulas with more than $i+1$-quantifier alternations, which we find surprising. The statements analogous to the above for S_2^i and G_i^* also hold.

In Chapter 9, we summarize our results on the QPC witnessing theorems and their definability in bounded arithmetic. We also provide two Tables (9.1 and 9.2 on pages 172 and 173, respectively) that summarize what is known about the complexity of QPC witnessing and the provability of the reflection principles.

Chapter 2

Preliminaries

Throughout this dissertation, we are concerned with the complexity of relations and functions on the natural numbers $\mathbb{N}$. Unless otherwise noted, we assume that natural numbers are represented in binary, and $|x|$ denotes the length of $x \in \mathbb{N}$.

Often we identify $\mathbb{N}$ with the set $\{0,1\}^*$ of finite binary strings, assuming appropriate schemata for encoding one in the other.

2.1 Complexity Theory

2.1.1 Complexity Classes

We are concerned with the complexity classes of relations, functions, and search problems (defined below) on $\mathbb{N}$. We assume that the reader is familiar with the following complexity classes

$$\mathbf{AC^0} \subsetneq \mathbf{TC}^0 \subseteq \mathbf{NC}^1 \subseteq \mathbf{P} \subseteq \mathbf{NP} = \mathbf{\Sigma}_1^p \subseteq \mathbf{\Sigma}_2^p \subseteq \ldots \subseteq \mathbf{\Sigma}_i^p \subseteq \ldots \mathbf{PH} \subseteq \mathbf{PSPACE}.$$

[Pap94a, Joh90] contain more information on these complexity classes. Note that, in the standard terminology, $\mathbf{AC^0}$, $\mathbf{TC}^0$, and $\mathbf{NC}^1$ are *nonuniform complexity classes*, defined in terms of nonuniform circuit families. Throughout this dissertation, we always work with the

Dlogtime-uniform versions of these classes. Note that **Dlogtime**-uniform $\mathbf{NC}^1$ coincides with $\mathbf{Alogtime}$, and **Dlogtime**-uniform $\mathbf{AC^0}$ is obtained from $\mathbf{Alogtime}$ by restricting the number of alternations to be $O(1)$. **Dlogtime**-uniform $\mathbf{TC}^0$ can be defined in a similar manner in terms of the *threshold Turing machine* of [PS88] (also see [All99]). From now on, we will simply write $\mathbf{NC}^1$, $\mathbf{AC^0}$, and $\mathbf{TC}^0$ to mean the **Dlogtime**-uniform of these classes.

2.1.2 Descriptive Characterization of $\mathbf{AC^0}$ and $\mathbf{TC}^0$

Both $\mathbf{AC^0}$ and $\mathbf{TC}^0$ have elegant, machine-independent characterizations based on descriptive complexity theory, and below we provide an informal presentation of this characterization. First, we fix the underlying first-order language $\mathcal{L}_{FO}$ to be

$$\mathcal{L}_{FO} = \{1, n, +, \cdot, \leq, Bit\},$$

where $0, 1, n$ are constant symbols, $+, \cdot$ are function symbols, and $\leq, Bit$ are predicate symbols. We refer to a first-order formula over $\mathcal{L}_{FO}$ as an *FO-formula*.

Structures that interprets FO-sentences are binary strings $X \in \{0,1\}^*$. More specifically, we interpret n to be $|X|$, the length of X, and we let the bound variables to range over $\{1, \ldots, n\}$, which is the set of indices of the string X. The symbols $+, \cdot, \leq$ assume the usual meaning, and $Bit(i)$ evaluates to true iff the ith bit of X is 1. It is clear that each FO-sentence ϕ defines the relation $R \subseteq \{0,1\}^*$ such that $X \in R$ iff $X \models \phi$, and we say that ϕ represents R. Finally, we fix an appropriate scheme for encoding k strings for $k \geq 1$ in one string so that an FO-sentence can represent a k-ary relation.

Theorem 2.1. *[BIS90, Imm99] A relation R is in $\mathbf{AC^0}$ iff it is representable by an FO-sentence.*

Note that there are other first-order languages $\mathcal{L}$ such that the first-order sentences over $\mathcal{L}$ characterize $\mathbf{AC^0}$. We only mention $\mathcal{L}_{FO}$ because of its obvious connection to the language of arithmetic.

FOM-formulas are obtained by introducing the *majority quantifier* M, where $Mx\phi(x)$ means that $\phi(x)$ is true for more than half of the possible x's.

Theorem 2.2. *[BIS90, Imm99] A relation R is in $\mathbf{TC}^0$ iff it is representable by an FOM-sentence.*

The following is an easy and useful fact: if $\phi(a)$ is an FOM-formula with one free variable a, then there exists another FOM-formula $\psi(b)$ such that, for every $k \in \mathbb{N}$, the sentence $\psi(k)$ holds iff there are k values of x such that $\phi(x)$ holds.

We will use the FOM characterization of $\mathbf{TC}^0$ in Section 5.4.1.

2.1.3 Complexity Theory of Functions and Search Problems

Let $f : \mathbb{N}^k \to \mathbb{N}$ be a k-ary function. Its *bit graph* R_f is defined as

$$R_f(x, i, b) \equiv \text{the } i\text{th bit of } f(x) \text{ is } b,$$

where $b \in \{0, 1\}$.

Definition 2.3. *Let $f : \mathbb{N}^k \to \mathbb{N}$ be a k-ary function. We say that f is a* polynomial-time function *if and only if both of the following are satisfied: (i) there is a polynomial p such that $|f(x)| \leq p(|x|)$ for all $x \in \mathbb{N}$; and (ii) the bit graph R_f is in $\mathbf{P}$.*

$\mathbf{NC}^1$-functions, $\mathbf{TC}^0$-functions, and $\mathbf{AC}^0$-functions are defined by replacing $\mathbf{P}$ in condition (ii) above with $\mathbf{NC}^1$, $\mathbf{TC}^0$, and $\mathbf{AC}^0$, respectively.

Let $R(\vec{x}, y)$ be an arbitrary $k+1$-ary relation and for each $\vec{x} \in \mathbb{N}^k$ define

$$Q_R(\vec{x}) = \{y : R(\vec{x}, y)\}.$$

The *search problem* Q_R is the following problem: given $\vec{x} \in \mathbb{N}^k$, find any $y \in Q_R(\vec{x})$. We call any y with $y \in Q_R(\vec{x})$ a *solutions* for $Q_R(\vec{x})$. We call R a *defining relation* of Q_R, and often we do not indicate R in the subscript. Throughout this dissertation we are concerned with *total*

search problems, or search problems every instance of which has a solution. Note that Q_R is a total search problem if and only if $(\forall x)(\exists y)R(x,y)$ holds. Note that every k-ary function is also a total search problem Q_R whose defining relation R is such that, for every $\vec{x}$, there is a unique y with $R(\vec{x},y)$.

Let f be a k-ary function and Q_R be a search problem defined by a $k+1$-ary relation R. We say that f *solves* Q_R if the following holds: for every $\vec{x} \in \mathbb{N}^k$, $f(\vec{x}) \in Q_R(\vec{x})$.

Definition 2.4. *We define* $\mathbf{FP}$ *to be the class of search problems* Q *for which there is a polynomial-time function that solves* Q. $\mathbf{FNC}^1$ *is the class of search problems that are solved by an* $\mathbf{NC}^1$*-function.*

Note that $\mathbf{FP}$ contains search problems whose defining relations are not polynomial-time. Here is a trivial example: $y \in Q_R(x)$ if and only if either (i) $y = 0$ or (ii) $y = 1$ and $K(x)$, where K is undecidable. However, the following is true: for every $Q \in \mathbf{FP}$, there exists a polynomial-time relation R' such that

$$R(x,y) \Rightarrow y \in Q(x)$$

for all $x, y \in \mathbb{N}$. Here $R'(x,y)$ is equivalent to the assertion that y is the output of a machine M on x, where M is a polynomial-time machine that solves Q.

Definition 2.5. *Let* Q_1 *and* Q_2 *be two total search problems. We say that* Q_1 *is* polynomial-time many-one reducible to Q_2*, and write* $Q_1 \leq_m Q_2$*, if and only if there are two polynomial-time functions* f *and* g *such that, whenever* $z \in Q_2(f(x))$, $g(z) \in Q_1(x)$.

Many-one $\mathbf{AC^0}$*-reduction between two search problems is defined in the same way except that* f *and* g *are required to be* $\mathbf{AC^0}$*-functions.*

Note that, if both Q_1 and Q_2 are functions, then $Q_1 \leq_m Q_2$ if and only if $Q_1(x) = g(Q_1(f(x)))$ for some polynomial-time functions f and g.

Let $i \geq 1$. Every relation $A(x) \in \mathbf{\Sigma}_i^p$ is of the form $(\exists)[|y| \leq p(|x|) \wedge R(x,y)]$, where p is some polynomial and $R \in \mathbf{\Pi}_{i-1}^p$. A *witness query* to A is an oracle query '$A(q)$?' such that a

positive answer is accompanied by a witness, i.e., some y such that $R(q, y)$. When we want to emphasize that a query is not a witness query, we call it a *yes-no query*.

Let $A \in \mathbf{\Sigma}_i^p$ for some $i \geq 1$ and let M be a deterministic machine that asks witness queries to A. Note that there can be multiple computations of M on input x because the positive answers to a witness query are not unique in general. Let $Output_M(x)$ be the set of all outputs that M on x can produce. We say that M solves a search problem Q if $Output_M(x) \subseteq Q(x)$ for all x; that is, no matter how a witness query is answered, M on x always halts with a solution for $Q(x)$.

We say that a Turing machine M asks queries *nonadaptively* if M presents all its queries to an oracle before receiving any answer, and once M receives the answers it does not ask any more query.

Definition 2.6. *Let $i \geq 1$.*

- $\mathbf{FP}^{\mathbf{\Sigma}_i^p}[wit]$ *is the class of search problems Q for which there exists a Turing machine that solves Q in polynomial time using a witness oracle $A \in \mathbf{\Sigma}_i^p$.*
- $\mathbf{FP}^{\mathbf{\Sigma}_i^p}[wit, O(g(n))]$ *is defined similarly to $\mathbf{FP}^{\mathbf{\Sigma}_i^p}[wit]$ except that the Turing machines that solve the problems in this class are allowed to ask only $O(g(n))$ witness queries. Throughout this dissertation, $g(n)$ is either 1 or $\log n$.*
- $\mathbf{FP}_{||}^{\mathbf{\Sigma}_i^p}[wit]$ *is defined similarly to $\mathbf{FP}^{\mathbf{\Sigma}_i^p}[wit]$ except that the Turing machines that solve the problems in this class have to ask the witness queries nonadaptively. $\mathbf{FP}_{||}^{\mathbf{\Sigma}_i^p}[wit, O(g(n))]$ is defined similarly.*
- *For each class defined above, there is a class obtained by removing 'wit', i.e., by requiring that all queries be* yes-no queries *instead of witness queries. $\mathbf{FP}^{\mathbf{\Sigma}_i^p}[O(\log n)]$ is an example of such a class.*

For more information on the search classes of the above form with yes-no queries, see [Kre88, BH88, Wag90, JT95, JT97]. The search classes defined using witness queries are used in [BKT93, Kra93, Kra95, Pol99] in the context of bounded arithmetic; see Section 2.3.

The following are some of the useful facts about the search classes that we introduced.

Theorem 2.7. *Let C be $\mathbf{\Sigma}_i^p$ for some $i \geq 1$. The following hold:*
(i) $\mathbf{FP}^C = \mathbf{FP}^C[wit]$.
(ii) If $Q \in \mathbf{FP}^C[wit, O(\log n)]$*, then Q is solvable by a polynomial-time machine that asks* $O(\log n)$ *yes-no queries followed by one witness query.*
(iii) $\mathbf{P}^C[O(f(n))]$ *is exactly the boolean functions in* $\mathbf{FP}^C[wit, O(f(n))]$.
(iv) $\mathbf{FP}^C[wit, O(\log n)] = \mathbf{FP}_{||}^C[wit]$.
(v) $\mathbf{FP}^C[wit, O(1)] = \mathbf{FP}_{||}^C[wit, O(1)]$.

Proof. (Ideas) (i) easily follows from the self-reducibility of $\mathbf{\Sigma}_i^p$. (ii) is shown by Krajíček in [Kra93, Kra95]. (iii) is an easy fact. Both (iv) and (v) follow from the following fact:

$$\mathbf{FP}^C[wit, f(n)] = \mathbf{FP}_{||}^C[wit, 2^{f(n)}] \quad \text{for any } f \in O(\log n)$$

The above equality can be proven in a completely analogous manner to the following well-known fact:

$$\mathbf{P}^C[f(n)] = \mathbf{P}_{||}^C[2^{f(n)}] \quad \text{for any } f \in O(\log n)$$

Proofs of the above equality appear in [BH88, Pap94a]. □

Interestingly, the yes-no version of item (iv) above is not known to hold, that is, we do not know whether $\mathbf{FP}^C[O(\log n)]$ is equal to $\mathbf{FP}_{||}^C$, where C is $\mathbf{\Sigma}_i^p$ for $i \geq 1$. Selman [Sel94] proves that, if these classes are equal for $C = \mathbf{NP}$, then $\mathbf{NP} = \mathbf{RP}$ and $\mathbf{P} = \mathbf{FewP}$.

2.2 Propositional Calculus and Proof Complexity

2.2.1 Basic Definitions

Throughout this dissertation we let T and F denote the truth values *True* and *False*, respectively. Let $\{p_i : i \in \mathbb{N}\}$ be the set of propositional variables.

Definition 2.8. Propositional formulas*, or simply formulas, are defined inductively as follows. (1) The atomic formulas are* (T)*,* (F)*, and* (p_i) *for every* $i \in \mathbb{N}$*. (2) If* ϕ *and* ψ *are formulas, then so are* $(\phi \wedge \psi)$*,* $(\phi \vee \psi)$*, and* $(\neg\phi)$*.*

Often we do now write all the parentheses. Moreover, although the connectives $\vee$ and $\wedge$ have fan-in 2, we often use the unbounded fan-in connectives such as $(A_1 \vee \ldots \vee A_k)$ and $\bigvee_{i=1}^{k} A_i$ as abbreviations for fully parenthesized formulas over connectives of fan-in 2, and this simplifies our presentation with no loss of generality. Finally, we write $A \supset B$ as an abbreviation of $\neg A \vee B$. We never use $\rightarrow$ for implication and reserve it for the arrow sign in sequent calculus (see below).

Definition 2.9. *([CR77]) Let* $TAUT$ *be the set of tautologies. A polytime computable function* Q *that maps* $\{0,1\}^*$ *onto* $TAUT$ *is called a* proof system*, and we say that* π *is a* Q*-proof of* $Q(\pi)$*.*

Let Q be a proof system and let π be a Q-proof. The *size* of π, denoted as $|\pi|$, is the total number of symbols in π, including the bits of the subscripts of the variables.

The following, fundamental theorem of propositional proof complexity was proven by Cook and Reckhow, connecting the question of proof lengths to open problems of complexity theory:

Theorem 2.10. *([CR77]) There exists a proof system in which every tautology* A *has a proof of size polynomial in* $|A|$ *if and only if* $\mathbf{NP} = \mathbf{coNP}$*.*

Cook and Reckhow also introduced the notion of *p-simulation*, which is a means to compare proof systems in terms of their power.

Definition 2.11. *([CR77]) Let* Q_1 *and* Q_2 *be proof systems. We say that* Q_2 p-simulates Q_1 *iff there exists a polytime function* f *satisfying the following: if* π_1 *is a* Q_1*-proof of* A*, then* $f(\pi_1)$ *is a* Q_2*-proof of* A*. We say that* Q_1 *and* Q_2 *are* p-equivalent *iff they p-simulate each other.*

Definition 2.12. *Let $f : \mathbb{N} \to \mathbb{N}$. Let Q be a proof system and let $\Phi = \{\Phi_n\}_{n\in\mathbb{N}}$ be a family of tautologies. We say that Φ has* Q-proofs of size f *if Φ_n for every $n \in \mathbb{N}$ has a Q-proof of size at most $f(|\Phi_n|)$. If $f \in n^{O(1)}$ then Φ is said to have polynomial-size Q-proofs, and if $f \in 2^{polylog(n)}$ then we say that Φ has* quasipolynomial-size Q-proofs.

2.2.2 Sequent Calculus PK

The sequent calculus LK is introduced by Gentzen in his 1935 paper [Gen35], and it is considered one of the most elegant formal systems for first-order logic. In this dissertation, we mostly work with the propositional fragment of LK and its bounded-depth variants, and this subsection is devoted to the presentation of these proof systems.

A fundamental object in the propositional sequent calculus PK is a *sequent* of the form

$$A_1, \ldots, A_k \to B_1, \ldots, B_l$$

where $A_1, \ldots, A_k$ and $B_1, \ldots, B_l$ are all propositional formulas, and the intended meaning of this sequent is

$$A_1 \wedge \ldots \wedge A_k \supset B_1 \vee \ldots \vee B_l.$$

Any sequence of formulas is called a *cedent*. The *antecedent* and *succedent* refer to the sequences of formulas appearing in the right and left of $\to$ in a sequent, respectively. The symbols Γ and Δ are used to denote arbitrary cedents.

Below we describe the inference rules of PK. Each rule has a *lower sequent* and one or two *upper sequents* separated by a horizontal line, indicating that the lower sequent follows from the upper sequent(s). First, PK has *structural rules*:

$$\text{weakening-left} : \frac{\Gamma \to \Delta}{A, \Gamma \to \Delta} \qquad \text{weakening-right} : \frac{\Gamma \to \Delta}{\Gamma \to \Delta, A}$$

$$\text{contraction-left} : \frac{A, A, \Gamma \to \Delta}{A, \Gamma \to \Delta} \qquad \text{contraction-right} : \frac{\Gamma \to \Delta, A, A}{\Gamma \to \Delta, A}$$

$$\text{exchange-left} : \frac{\Gamma_1, A, B, \Gamma_2 \to \Delta}{\Gamma_1, B, A, \Gamma_2 \to \Delta} \quad \text{exchange-right} : \frac{\Gamma \to \Delta_1, A, B, \Delta_2}{\Gamma \to \Delta_1, B, A, \Delta_2}$$

The contraction and exchange rules allow us to treat cedents as sets of formulas.

We introduce the *propositional rules* below. There are two rules for each connective, introducing the connective in either side of the arrow sign.

$$\neg\text{-left} : \frac{\Gamma \to \Delta, A}{\neg A, \Gamma \to \Delta} \qquad \neg\text{-right} : \frac{A, \Gamma \to \Delta}{\Gamma \to \Delta, \neg A}$$

$$\vee\text{-left} : \frac{A, \Gamma \to \Delta \quad B, \Gamma \to \Delta}{A \vee B, \Gamma \to \Delta} \qquad \vee\text{-right} : \frac{\Gamma \to \Delta, A, B}{\Gamma \to \Delta, A \vee B}$$

$$\wedge\text{-left} : \frac{A, B, \Gamma \to \Delta}{A \wedge B, \Gamma \to \Delta} \qquad \wedge\text{-right} : \frac{\Gamma \to \Delta, A \quad \Gamma \to \Delta, B}{\Gamma \to \Delta, A \wedge B}$$

Finally, the following is the *cut rule*. Note that this is the only rule that allows us to remove a formula from the existing sequents.

$$\text{Cut:} \frac{A, \Gamma \to \Delta \quad \Gamma \to \Delta, A}{\Gamma \to \Delta}$$

There are three kinds of *logical axioms* in PK: (i) $\mathsf{F} \to$, (ii) $\to \mathsf{T}$, or (iii) $A \to A$ for any propositional formula A. These sequents are trivially valid.

A PK*-proof* is a directed acyclic graph whose nodes are sequents and there is a directed edge from an upper sequent of an inference step to its lower sequent. The sequents that have no incoming edges are called *initial sequents*, and in a PK-proof the initial sequents must be logical axioms. The following is a more formal definition.

Definition 2.13. *A* PK-proof *is a sequence* $S_1, \ldots, S_k$ *of sequents such that either* S_i *is a logical axiom or* S_i *is derived from at most two preceding sequents by one of the inference rules.* S_k *is called the* endsequent. *If the endsequent is* $\to A$ *for* A *a formula, then we say that this is a* PK*-proof of* A.

Let $\Phi = \{A_1, \ldots, A_k\}$ *be a set of formulas. A* PK*-derivation of* B *from* Φ *is a* PK*-proof of* B *except that sequents of the form* $\to A_j$ *for* $A_j \in \Phi$ *are allowed to appear as initial sequents.*

PK is a complete formal system in the sense that formula A is valid if and only A has a PK-proof. PK is also derivationally complete, which means that $A_1 \wedge \ldots \wedge A_k \supset B$ is valid if and only if B has a PK-derivation from $\Phi = \{A_1, \ldots, A_k\}$.

For each inference rule, the *principal formulas* are the formulas in the lower sequent to which the rule is applied. For example, the principal formula of $\wedge$-right above is $A \wedge B$. The exchange rules are the only rules with two principal formulas, and cut has no principal formula. The *auxiliary formulas* of a rule are the formulas in the upper sequents to which the rule is applied. For example, the auxiliary formulas of $\wedge$-right are A and B. The weakening rules do not have any auxiliary formula.

Let π be a PK-proof. For each inference step in π, we say that the lower sequent is the *immediate descendant* of the upper sequent(s). The descendant relation is the transitive closure of the immediate descendent relation. The ancestor relation among the sequents of π is defined similarly. We also define the descendent-ancestor relation on the *formulas* occurring in π as follows. For each inference step in π, its principal formula is defined to be the descendent of the auxiliary formulas, and the descendent relation of formulas of π is taken to be the transitive closure of it. The ancestor relation for formulas is defined similarly. We emphasize that the ancestor-descendent relations for formulas and sequents are two distinct relations that should not be confused. Let S_1, S_2 be two sequents of a proof π and let A_1 and A_2 be formulas occurring in S_1 and S_2, respectively. Note that, S_1 is an ancestor of S_2 whenever A_1 is an ancestor of A_2. However, the opposite is not true in general and, moreover, even if S_1 is an ancestor of S_2, it is possible that S_1 does not contain any ancestor of A_2.

Treelike PK is PK with the restriction that every proof can be drawn as a tree; more specifically, a tree-like proof is a proof in which each sequent is used as an upper sequent of an inference step at most once. Krajíček showed that tree-like PK is as powerful as PK:

Theorem 2.14. *[Kra95] Treelike PK p-simulates PK.*

The *depth* of a formula is defined as follows. First, if a formula is either atomic or a literal, then its depth is zero. Let ϕ be a formula in which at least one binary connective occurs. Push the negations within ϕ inward by applying De Morgan's law so that every negation is placed in front of an atomic formula; call the result ϕ'. Let k be the maximum number of times the connectives change between $\vee$ and $\wedge$ in any path from the root to a leaf in the tree form of ϕ.

Define the depth of ϕ to be $k+1$. Thus, clauses have depth one, and CNF formulas have depth two.

Definition 2.15. *Let $f : \mathbb{N} \to \mathbb{N}$ and $d \in \mathbb{N}$, and Let $\{\Phi_n\}_{n\in\mathbb{N}}$ be a family of tautologies. We say that the family has* depth-d PK-proofs *of size f if, for every $n \in \mathbb{N}$, PK has a proof of Φ_n of size at most $f(|\Phi_n|)$ in which every formula has depth at most d. We say that the family has* bounded-depth PK-proofs *if it has depth-d PK-proofs for some $d \in \mathbb{N}$.*

There have been a number of papers on bounded-depth PK. See [BB03, Kra95, MPW00, BP96, KPW95, PBI93] for more information.

2.3 First-order Theories of Bounded Arithmetic

Many of the definitions and results presented in this subsection are from [Bus86, Kra95, Bus98a]. Let

$$\mathcal{L}_A = \{0, S, +, \cdot, \lfloor\frac{x}{2}\rfloor, |x|, \#, \leq\}$$

be the *language of bounded arithmetic*, where 0 is a constant, S is the successor function, $|x| = \lceil \log_2 x \rceil$ denotes the binary length of x, and $x\#y = 2^{|x|\cdot|y|}$ is the smash function. Note that for every term $t(a)$ in the language $\mathcal{L}_A$, there exists a constant c such that $|t(x)| \in O(|x|^c)$.

We use $(\exists x \leq t)\phi$ as an abbreviation for $(\exists x)[x \leq t \wedge \phi]$ and call existential quantifiers of this form *bounded*. Similarly, a *bounded universal quantifier* is of the form $(\forall x \leq t)\phi$ and it abbreviates $(\forall x)[x \leq t \supset \phi]$. A quantifier is said to be *sharply bounded* if it is of the form $(\exists x \leq |t|)$ or $(\forall x \leq |t|)$. A formula is said to be *bounded* if all of its quantifiers are bounded. A formula is said to be *sharply bounded* if all of its quantifiers are sharply bounded.

The sets $\Sigma_0^b = \Pi_0^b$ are the sets of formulas in which all quantifiers are sharply bounded. For $i \geq 1$, Σ_i^b and Π_i^b are defined inductively to be the smallest sets of formulas satisfying the following conditions:

(1) $\Sigma_{i-1}^b \cup \Pi_{i-1}^b \subseteq \Sigma_i^b \cap \Pi_i^b$;

(2) Both Σ_i^b and Π_i^b are closed under $\vee$, $\wedge$, and sharply bounded quantifiers;

(3) For every A, if $A \in \Pi_i^b$ then $\neg A \in \Sigma_i^b$; and if $A \in \Sigma_i^b$ then $\neg A \in \Pi_i^b$; and

(4) Σ_i^b and Π_i^b are closed under existential bounded quantifiers and universal bounded quantifiers, respectively.

Informally, for $i \geq 1$, formula A is in Σ_i^b iff A has a prenex form A' such that either A' has at most $i-2$ alternations of bounded quantifiers or A' has exactly $i-1$ alternations of bounded quantifiers with the outermost bounded quantifier $\exists$. Similarly for Π_i^b except that, if A' has exactly $i-1$ alternations of bounded quantifiers, the outermost one must be $\forall$. The classes Σ_i^b and Π_i^b of bounded formulas are useful because they represent the classes $\mathbf{\Sigma}_i^p$ and $\mathbf{\Pi}_i^p$ at the ith level of $\mathbf{PH}$, respectively; see Theorem 2.17 below.

Strictly bounded formulas are bounded formulas that starts with a sequence of bounded quantifiers followed by a sharply bounded formula.

Definition 2.16. *The classes of* strictly bounded formulas *are defined inductively as follows. First, let $\hat{\Sigma}_0^b$ be the class of sharply bounded formulas and let $\hat{\Pi}_0^b = \hat{\Sigma}_0^b$. For $i \geq 1$, $\hat{\Sigma}_i^b$ is the smallest class that contains $\hat{\Sigma}_{i-1}^b \cup \hat{\Pi}_{i-1}^b$ closed under bounded existential quantification. Similarly $\hat{\Pi}_i^b$ is the class containing $\hat{\Sigma}_{i-1}^b \cup \hat{\Pi}_{i-1}^b$ closed under bounded universal quantification.*

Let $\underline{\mathbb{N}}$ be the standard model of arithmetic, in which nonlogical symbols of $\mathcal{L}_A$ assume their standard interpretation. We say that a formula $\phi(a_1, \ldots, a_k)$ *represents* a relation $R(x_1, \ldots, x_k)$ if and only if

$$\underline{\mathbb{N}} \models \phi(s_1, \ldots, s_k) \iff R(n_1, \ldots, n_k),$$

where, for each $i \in [1, k]$, $n_i \in \mathbb{N}$ and s_i is a numeral representing n_i.

Theorem 2.17. *[Bus86] A predicate R is in $\mathbf{\Sigma}_i^p$ if and only if there exists a Σ_i^b formula ϕ that represents R.*

Let $BASIC$ is the set of axioms that define the meaning of the nonlogical symbols in L_A .

Definition 2.18. *Let Φ be a set of formulas. The Φ-LIND axioms are the formulas*

$$A(0) \wedge (\forall x)[A(x) \supset A(Sx)] \supset (\forall x)A(|x|)$$

for all formulas $A \in \Phi$. Similarly, Φ-IND axioms are the formulas

$$A(0) \wedge (\forall x)[A(x) \supset A(Sx)] \supset (\forall x)A(x)$$

for all $A \in \Phi$.

For $i \geq 0$, S_2^i is the theory axiomatized by $BASIC$ axioms plus Σ_i^b-LIND, and T_2^i is the theory axiomatized by $BASIC$ plus Σ_i^b-IND. Finally, $S_2 = \bigcup_{i \geq 0} S_2^i$.

In light of Theorem 2.17, the theory T_2^1 is essentially the theory of arithmetic with induction allowed only on $\mathbf{NP}$ predicates and S_2^1 is obtained by restricting T_2^1 so that, instead of induction, only length induction on $\mathbf{NP}$ predicates are allowed. For $i \geq 1$, T_2^i and S_2^i are theories with induction and length induction on $\mathbf{\Sigma}_i^p$-predicates, respectively. The main motivation for studying these theories is their close connections to complexity classes. For example, as we will state below, S_2^1 proves the totality of every search problems in $\mathbf{FP}$ and therefore $\hat{\Sigma}_1^b$-define them.

The theories S_2^i and T_2^i are related to each other in the following way [Bus86]:

$$S_2^1 \subseteq T_2^1 \subseteq S_2^2 \subseteq T_2^2 \subseteq S_2^3 \subseteq \ldots. \tag{2.1}$$

The above hierarchy of theories is referred to as the S_2 hierarchy. It is a fundamental open problem whether the S_2 hierarchy extends infinitely or collapses at a finite level. This question is related to whether the polynomial-time hierarchy collapses at a finite level:

Theorem 2.19. *[KPT91]) For every $i \geq 1$, if $T_2^i = S_2^{i+1}$, then $\mathbf{\Sigma}_{i+1}^p \subseteq \mathbf{\Delta}_{\mathbf{i+1}}^{\mathbf{p}}/poly$, which implies $\mathbf{PH} = \mathbf{\Sigma}_i^p$.*

Buss [Bus95] and Zambella [Zam96] proved stronger connections between the collapse of the S_2 hierarchy and that of $\mathbf{PH}$.

Although no significant progress has been made in deciding whether the S_2 hierarchy collapses, the following conservativity relation is known to hold:

Theorem 2.20. *([Bus90]) For all $i \geq 1$, S_2^{i+1} is Σ_{i+1}^b-conservative over T_2^i.*

Currently no conservativity of T_2^i over S_2^i is known. It is believed by some that these theories are separated by Σ_1^b-statements.

We use the following definition of Σ_i^b-definability of search problems. Buss, Krajíček, and Takeuti uses the same definition in [BKT93].

Definition 2.21. *Let Q be a search problem. We say Q is $\hat{\Sigma}_i^b$-definable in theory T if the following two conditions are met: (i) there is a $\hat{\Sigma}_i^b$-formula $\phi(\vec{x}, y)$ with all free variables indicated such that*

$$\underline{\mathbb{N}} \models (\forall \vec{x})(\forall y)[\phi(\vec{x}, y) \supset y \in Q(\vec{x})],$$

and (ii) T proves $(\forall \vec{x})(\exists y)\phi(\vec{x}, y)$. We call $\phi(\vec{x}, y)$ a $\hat{\Sigma}_i^b$-defining formula *of Q.*

We like to make two remarks on the above definition of definability. First, note that we require that the formula ϕ be strict. This is irrelevant if T is either S_2^i or T_2^i for some $i \geq 1$ and Q is being Σ_i^b-defined in T, since these theories prove that every Σ_i^b-formula is equivalent to a $\hat{\Sigma}_i^b$-formula. However, it is not known whether S_2^i proves the equivalence of Σ_{i+1}^b-formulas with $\hat{\Sigma}_{i+1}^b$-formulas.

Second, when Q_R is Σ_i^b-definable in T, we do not require that a Σ_i^b-defining formula $\phi(\vec{x}, y)$ represent the defining relation R; instead, $\phi(\vec{x}, y)$ represents a relation R' such that, for all $x, y \in \mathbb{N}$, $R'(x, y)$ implies $R(x, y)$. This is consistent with the way we define search classes in Section 2.1.3. It is possible to restate the results of this dissertation using the notion of definability that requires

$$\underline{\mathbb{N}} \models (\forall \vec{x})(\forall y)[\phi(\vec{x}, y) \leftrightarrow y \in Q(\vec{x})],$$

which demands that a defining formula of Q represent its defining relation. In fact, we developed such a framework in [Mor01] by introducing the notions of *exact solvability*, *exact reducibility*, and *exact definability*. However, we decided in favour of Definition 2.21 since it is simper and since there is no loss of generality.

Below we state the known facts about the complexity of various definable search problems of bounded arithmetic. We use the notation $C(\mathbf{PLS})$ to denote

$$C(\mathbf{PLS}) = \{Q : Q \leq_m Q' \text{ for some } Q' \in \mathbf{PLS}\},$$

where $\mathbf{PLS}$ is the class defined in [JPY88] that captures local search problems; we will define $\mathbf{PLS}$ more formally in Chapter 3. We showed in [Mor01] that $\mathbf{PLS}$ is not closed under $\leq_m$ unless $\mathbf{PLS} \subseteq \mathbf{FP}$, which is not believed to be true. Thus, $C(\mathbf{PLS})$ is likely to properly contain $\mathbf{PLS}$.

Theorem 2.22. *Let $i \geq 1$.*
(i) ([Bus86]) A search problem Q is $\hat{\Sigma}^b_i$-definable in S^i_2 if and only if $Q \in \mathbf{FP}^{\mathbf{\Sigma}^p_{i-1}}$.
(ii) ([BK94, CK98, Mor01]) A search problem Q is $\hat{\Sigma}^b_i$-definable in T^i_2 if and only if $Q \in C(\mathbf{PLS})^{\mathbf{\Sigma}^p_{i-1}}$.

Theorem 2.23. *Let $i \geq 1$.*
(i) ([Kra93]) A search problem Q is $\hat{\Sigma}^b_{i+1}$-definable in S^i_2 if and only if $Q \in \mathbf{FP}^{\mathbf{\Sigma}^p_i}[wit, O(\log n)]$.
(ii) ([Bus90]) A search problem Q is $\hat{\Sigma}^b_{i+1}$-definable in T^i_2 if and only if $Q \in \mathbf{FP}^{\mathbf{\Sigma}^p_i}$.

Note that the Σ^b_{i+1}-conservativity of S^{i+1}_2 over T^i_2 (Theorem 2.20) implies that these two theories $\hat{\Sigma}^b_k$-define the same class of search problems for every $k \leq i+1$.

Theorem 2.24. *([Pol99]) Let $i \geq 1$ and let T denote either S^i_2 or T^i_2. For every $k \geq i+2$, the following holds: a search problem Q is $\hat{\Sigma}^b_k$-definable in T if and only if $Q \in \mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(1)]$.*

Table 2.1 on page 34 summarizes the three Theorems above.

Pollett's result (Theorem 2.24) state that, for every $i \geq 1$ and $k \geq i+2$, S^i_2 and T^i_2 $\hat{\Sigma}^b_k$-define the same class of search problems. This does not mean that the two theory has the same $\hat{\Sigma}^b_k$-theorems, since search problems may be defined by different formulas in the two theories.

Σ_5^b	$\mathbf{FP}^{\mathbf{\Sigma}_4^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_4^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_4^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_4^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_4^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_4^p}[wit, O(1)]$
Σ_4^b	$\mathbf{FP}^{\mathbf{\Sigma}_3^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_3^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_3^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_3^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_3^p}[wit, O(\log n)]$	$\mathbf{FP}^{\mathbf{\Sigma}_3^p}$
Σ_3^b	$\mathbf{FP}^{\mathbf{\Sigma}_2^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_2^p}[wit, O(1)]$	$\mathbf{FP}^{\mathbf{\Sigma}_2^p}[wit, O(\log n)]$	$\mathbf{FP}^{\mathbf{\Sigma}_2^p}$	$\mathbf{FP}^{\mathbf{\Sigma}_2^q}$	$C(\mathbf{PLS})^{\mathbf{\Sigma}_2^p}$
Σ_2^b	$\mathbf{FP}^{\mathbf{NP}}[wit, O(\log n)]$	$\mathbf{FP}^{\mathbf{NP}}$	$\mathbf{FP}^{\mathbf{NP}}$	$C(\mathbf{PLS})^{\mathbf{NP}}$	$C(\mathbf{PLS})^{\mathbf{NP}}$	??
Σ_1^b	$\mathbf{FP}$	$C(\mathbf{PLS})$	$C(\mathbf{PLS})$	??	??	??
	S_2^1	T_2^1	S_2^2	T_2^2	S_2^3	T_2^3

Table 2.1: The complexity of the $\hat{\Sigma}_i^b$-definable search problems in S_2^j and T_2^j for $i, j \in \{1, 2, 3, 4\}$.

Part I

Type-2 Search Problems

Chapter 3

Type-2 Search Problems and Proof Complexity

3.1 Type-2 Search Problems

In the papers [JPY88, Pap94b], total search problems are classified according to the combinatorial principle in the finite domain that guarantees the totality of the problems. These classes contain numerous natural problems, some of which are complete. For example, *Polynomial Local Search* ($\mathbf{PLS}$), which is the class of problems efficiently solvable by local search heuristics, is characterized by the iteration principle "every finite dag has a sink"; and *Polynomial Pigeonhole Principle* ($\mathbf{PPP}$), which has relevance to cryptographic hash functions, corresponds to the pigeonhole principle "there is no injective mapping from $[n+1]$ to $[n]$." The class *Polynomial Parity Argument* ($\mathbf{PPA}$) is defined by the parity principle "there is no perfect matching in an odd-sized graph" and contains the problems of finding various economic equilibria. And its variants $\mathbf{PPAD}$ and $\mathbf{PPADS}$ are defined in a similar manner ($\mathbf{PPAD}$ was called **PSK** in [Pap94b], and it is given this name in [BCE+98]).

Beame et al. [BCE+98] generalize the notion of search problem so that the instances of search problem Q consist not only of strings, which are type-0 objects, but also functions and

relations, which are type-1 objects. More formally, let R be a type-2 relation with arguments $(\alpha_1, \ldots, \alpha_k, x, y)$, where x and y are strings and α_i for each $i \in [1, k]$ is either a string function or a string relation. R defines a type-2 search problem Q_R in the usual way.

The complexity of type-2 relation, functions, and search problems is measured with respect to a Turing machine that receives the type-0 arguments on its input tape and is allowed to access the type-1 arguments as oracles [Tow90]. In particular, a type-2 function $F(\alpha_1, \ldots, \alpha_k, x)$ is said to be polynomial-time computable if it is computed by a deterministic Turing machine in time polynomial in $|x|$ with oracle access to $\alpha_1, \ldots, \alpha_k$.

Let Q be a type-2 search problem. Q can be used as an oracle in the following way. A Turing machine M presents a query to Q in the form $(\beta_1, \ldots, \beta_l, y)$, where each of $\beta_1, \ldots, \beta_l$ is a polynomial-time function or relation, and we assume that these are encoded as polynomial-size circuits that compute the corresponding functions or relations. In the next step M receives in its answer tape some z that is a solution for $Q(\beta_1, \ldots, \beta_k, y)$.

Let Q_1 and Q_2 be two type-2 search problems. We say Q_1 is *Turing reducible to* Q_2 and write $Q_1 \leq_T Q_2$ iff there exists an oracle Turing machine M that, given an instance $(\alpha_1, \ldots, \alpha_k, x)$ of Q_1, outputs some $z \in Q_1(\alpha_1, \ldots, \alpha_k, x)$ in polynomial-time using $\alpha_1, \ldots, \alpha_k$ and Q_2 as oracles, where each query to Q_2 is of the form $(\beta_1, \ldots, \beta_l, y)$ with $m \in n^{O(1)}$ and with each β_i for each $1 \leq i \leq l$, a function or a relation that is polynomial-time computable using $\alpha_1, \ldots, \alpha_k$ as oracles.

Q_1 is *many-one reducible to* Q_2, written $Q_1 \leq_m Q_2$, if $Q_1 \leq_T Q_2$ by an oracle Turing machine that asks at most one query to Q_2. We write $Q_1 \equiv_m Q_2$ if Q_1 and Q_2 are many-one reducible to each other. Equivalently, $Q_1 \leq_m Q_2$ iff there exist two type-2 functions f, g such that

$$g(w) \in Q_1(\alpha, \ldots, \alpha_k, x) \text{ whenever } f(\alpha_1, \ldots, \alpha_k, x) = (\beta_1, \ldots, \beta_l, y) \wedge w \in Q_2(\beta_1, \ldots, \beta_l, y),$$

where f, g, $\beta_1, \ldots, \beta_l$ are all polynomial-time computable with oracle access to α. This definition of many-one reduction between type-2 search problems is closer to that of type-1 search

$$\begin{array}{ccc} \text{Instances of } Q_1 & f & \text{Instances of } Q_2 \\ (\alpha_1, \ldots, \alpha_k, x) & \longrightarrow & (\beta_1, \ldots, \beta_l, y) \\ & g & \\ \text{Solutions of } Q_1(\alpha_1, \ldots, \alpha_k, x) & \longleftarrow & \text{Solutions of } Q_2(\beta_1, \ldots, \beta_l, y) \end{array}$$

Figure 3.1: A many-one reduction from Q_1 to Q_2. Note that f, g and β are all polytime computable with oracle access to α.

problems, introduced in Section 2.1.3 as Definition 2.5. Figure 3.1 is a schematic view of a many-one reduction from Q_1 to Q_2.

When Q_1 is type-1, then Q_1 is treated as a type-2 problem with no type-1 arguments.

Definition 3.1. *([BCE+98]) Let Q be a type-2 search problem. Then $C(Q)$ is defined as*

$$C(Q) = \{Q' : Q' \text{ is type-1 and } Q' \leq_m Q\}$$

Thus, $Q' \in C(Q)$ means that Q' is polynomial-time solvable by finding a solution for a type-1 instance of Q that is obtained by fixing the type-1 arguments of Q to be polynomial-time predicates and functions.

For a type-2 search problem Q and an oracle A, $C(Q)^A$ is the class of type-1 search problems Q' such that $Q' \leq_m Q$ in a world in which A can be accessed at unit cost.

Theorem 3.2. *([CIY97]) Let Q_1 and Q_2 be type-2 total search problems whose defining relations are polynomial-time. The following are equivalent: (i) $Q_1 \leq_m Q_2$; (ii) for all oracles A, $C(Q_1)^A \subseteq C(Q_2)^A$; and (iii) there exists a generic oracle G such that $C(Q_1)^G \subseteq C(Q_2)^G$.*

3.2 Combinatorial Principles and Search Problems

Beame et al. [BCE+98] introduce several type-2 search problems that correspond to the combinatorial principles that characterize the search classes of [Pap94b]. We extend their approach

into a systematic method of defining type-2 search problems from combinatorial principles that are represented as sentences of first-order logic with equality.

3.2.1 Defining a Type-2 Search Problem from an Existential Sentence

Let $\mathcal{L}$ be an arbitrary first-order language and let Φ be a sentence over $\mathcal{L}$ of the form

$$\Phi \equiv (\exists x_1 \dots \exists x_k)\phi(x_1, \dots, x_k) \tag{3.1}$$

for some quantifier-free ϕ. Let us call such sentences $\exists$-sentences. As usual, we assume that the equality symbol $=$ is in Φ even though we do not explicitly include it in $\mathcal{L}$. Φ is interpreted in a structure $\mathcal{M}$ which defines the universe of discourse and the meaning of constants, functions, and relations of $\mathcal{L}$. Some symbols of $\mathcal{L}$ may be designated as *built-in symbols* with which we associate predetermined interpretation. We require that $=$ be interpreted as true equality, and therefore $=$ is a built-in predicate. A symbol (function, predicate, or constant) of $\mathcal{L}$ that is not built-in is called an *input symbol*, since it will become a type-1 input to the corresponding type-2 search problem.

Definition 3.3. *Define a* canonical structure *to be a structure such that (1) the universe of discourse is $V_n = \{0, \dots, 2^n - 1\}$ for some $n \geq 1$; and (2) every* built-in *symbol of $\mathcal{L}$ assumes the predetermined interpretation. We abuse the notation and write V_n to denote the canonical structure with the set V_n the universe of discourse.*

The requirement that a canonical structure be of size 2^n for some n is not essential and it is there to simplify our presentation. We could have done without it but it would require a slightly more complicated definition of the witnessing problem Q_Φ (see below). The only built-in symbols we use in this dissertation are $=$, $\leq$, and 0, which we interpreted as equality, the standard ordering of numbers, and $0 \in \mathbb{N}$, respectively.

Assume that Φ holds in every canonical structure. Then the corresponding witness problem is the following: given a canonical structure V_n, find a tuple $\langle v_1, \dots, v_k \rangle \in (V_n)^k$ such that

$\phi(v_1, \ldots, v_k)$ holds in V_n. We formulate the witness problem as the type-2 search problem Q_Φ whose type-0 argument x specifies the universe of discourse $V_{|x|}$ and each of whose type-1 arguments corresponds to an input symbol of $\mathcal{L}$. Built-in symbols are not part of the type-1 arguments, since their interpretations in $V_{|x|}$ are already fixed. Finally, since only the length of x is used to define $V_{|x|}$, we assume without loss of generality that the type-0 argument of Q_Φ is always of the form 1^n for $n \geq 1$.

The following is an easy upper bound on the complexity of any type-2 total search problem Q_Φ, where Φ is an $\exists$-sentence.

Lemma 3.4. *If Q_Φ is a total type-2 search problem defined by an $\exists$-sentence Φ, then Q_Φ is in type-2* $\mathbf{FP}^{\mathbf{NP}}$.

Proof. Let $(\alpha_1, \ldots, \alpha_k, 1^n)$ be an instance of Q_Φ. A solution for $Q_\Phi(\alpha_1, \ldots, \alpha_k, 1^n)$ is found by binary search, asking an $\mathbf{NP}$ query "does there exist $v > m$ such that $v \in Q_\Phi(\alpha_1, \ldots, \alpha_k, 1^n)$?" for various $m \in V_n$. □

3.2.2 The Five Combinatorial Principles

In this Subsection, we introduce five combinatorial principles that are represented by $\exists$-sentences. They are of particular interest in our study of search problems because four of the five characterize the search classes $\mathbf{PPP}$, $\mathbf{PPA}$, $\mathbf{PPAD}$, and $\mathbf{PLS}$, respectively (see Definition 3.6 below), and the fifth one also captures certain natural search problems; furthermore, a great deal is known about the proof complexity of the propositional translation of each of the five principles. Later, we will derive a number of separations among the corresponding type-2 search problems via our main results and the existing proof complexity results. For readability we present these principles in implicational form; it is easy to see that all of them are $\exists$-sentences. Moreover, all of them hold in every canonical structure; in fact, all of them except $\mathbf{LONELY}$ hold in every finite structure, and $\mathbf{LONELY}$ holds in every finite structure of even size.

(1) $(\forall x)[x = f(f(x))] \wedge f(0) = 0 \supset (\exists x)[x \neq 0 \wedge x = f(x)]$

This sentence states that, if every element is either *lonely* (i.e., is matched with itself) or matched with a unique partner, and if 0 is lonely, then there exists another lonely element. This is essentially the parity principle 'no odd-sized graph has a perfect matching', and it holds in every structure whose size is even; therefore, it holds in every canonical structure. **LONELY** is the corresponding search problem.

(2) $(\forall x)[f(x) \neq 0] \supset (\exists x, y)[x \neq y \wedge f(x) = f(y)]$

This states that if 0 is not in the image of f, then there exist two distinct elements that are mapped to the same element by f; this is the injective, functional pigeonhole principle, and the corresponding search problem is **PIGEON**.

(3) $(\exists x_1, y_1, x_2, y_2)[(x_1 \neq x_2 \vee y_1 \neq y_2) \wedge f(x_1, y_1) = f(x_2, y_2)]$

Let V be the universe of a structure that interprets this sentence. Since f is a binary function, it is a mapping from $V \times V$ to V. This sentence states that f is not an injective mapping, and it holds in any finite universe V. This is a weak pigeonhole principle, which is similar to (2) but the domain size is the square of the range size. We call the corresponding problem **WeakPIGEON**.

(4) $f(0) > 0 \wedge (\forall x)[f(x) \geq x] \supset (\exists x)[x < f(x) \wedge f(x) = f(f(x))]$

This is the iteration principle of [BK94, CK98], and we call the corresponding type-2 problem **ITERATION**. It states that, if f is nondecreasing and $f(0) > 0$, then there exist x such that $f(x) > x$ and $f(x) = f(f(x))$. Note that it contains a built-in ordering $\leq$. In graph-theoretic terms, the iteration principle states that every dag $G = (V, E)$ with at least one edge whose vertices are ordered so that $(u, v) \in E$ implies $u < v$ has a sink.

(5) $(\forall x)[f(x) \neq 0] \supset (\exists x)[x \neq 0 \wedge x \neq f(g(x))] \vee (\exists x)[x \neq g(f(x))]$

This is another weakening of the pigeonhole principle of (2). It states that, if 0 is not in the image of f, then g cannot be the inverse of f, and therefore f is not a bijection between $[N]$ and $[N-1]$. This principle is commonly known as the functional onto-pigeonhole principle, and the corresponding search problem is $\mathbf{OntoPIGEON}$.

Here we present yet another type-2 problem, $\mathbf{Source.Or.Sink}$ (**SOS**), and show that $\mathbf{OntoPIGEON} \equiv_m \mathbf{SOS}$. **SOS** is originally introduced by Beame et. al [BCE+98] to capture the class $\mathbf{PPAD}$. Its type-1 arguments are two unary functions $succ$ and $pred$. An instance $(succ, pred, 1^n)$ of $\mathbf{SOS}$ defines a directed graph $G = (V_n, E)$, where $(u, v) \in E$ if and only if all of the following hold: $u \neq v$, $succ(u) = v$, and $pred(v) = u$. A *source* is a node with indegree 0 and outdegree 1, and a *sink* is a node with indegree 1 and outdegree 0. Note that both the maximum indegree and the maximum outdegree of G are 1. $\mathbf{SOS}$ is defined by the following combinatorial principle, which we state informally:

$$0 \text{ is a source} \supset (\exists x)[x \text{ is either a source other than } 0 \text{ or a sink}].$$

Lemma 3.5. $\mathbf{OntoPIGEON} \equiv_m \mathbf{SOS}$.

Proof. We first show $\mathbf{OntoPIGEON} \leq_m \mathbf{SOS}$. Given an instance $(f, g, 1^n)$ of $\mathbf{OntoPIGEON}$, set $succ'(v) = f(v)$ and $pred'(v) = g(v)$ for every $v \in V_n$. We show that every solution for $\mathbf{SOS}(succ' pred' 1^n)$ corresponds to a solution for $\mathbf{OntoPIGEON}(f, g, 1^n)$. Let G' be the directed graph specified by $(succ', pred', 1^n)$. Three cases arise. In the first case, node 0 is not a source in G'. If node 0 has an incoming edge, then we have $f(g(0)) = 0$ and thus $g(0)$ is a witness to $\mathbf{OntoPIGEON}$. Otherwise node 0 has no outgoing edge and therefore $0 \neq g(f(0))$. In this case, 0 is a witness. The second case is when there is some $v \in V_n$ such that $v \neq 0$ and v is a source. Since the indegree of v is 0, we have $v \neq 0 \wedge v \neq f(g(v))$, making v a witness for $\mathbf{OntoPIGEON}$. In the third case, assume that $v \in V_n$ is a sink of G'. Since the outdegree of v is 0, we have $v \neq g(f(v))$, so v is a witness to $\mathbf{OntoPIGEON}$.

Next, we show that $\mathbf{SOS} \leq_m \mathbf{OntoPIGEON}$. Let $(succ, pred, 1^n)$ be an instance of $\mathbf{SOS}$, and let G be the directed graph defined by this instance. We define an instance $(f', g', 1^n)$

of $\mathbf{OntoPIGEON}$ as follows. For each $v \in V_n$, if v is isolated in G, then we set $f'(v) = g'(v) = v$; otherwise, we set $f'(v) = succ(v)$ and $g'(v) = pred(v)$. It suffices to show that each witness to the $\mathbf{OntoPIGEON}$ principle gives rise to a witness to the $\mathbf{SOS}$ principle. Three cases arise. In the first case, we have some $v \in V_n$ such that $f'(v) = 0$. Then either node 0 is not a source or v is a sink. The second case is when we have $v \neq 0$ such that $v \neq f'(g'(v))$. Then v is a source in G, since it is not isolated and its indegree is zero. In the third case, we have $v \neq g'(f'(v))$. Then v is a sink in G since v is not isolated and its outdegree is zero. □

3.2.3 Search Classes and Combinatorial Principles

Now we are ready to formulate the search classes of [JPY88, Pap94b] in terms of the type-2 search problems. Although we give below the type-2 formulation of these classes as a definition, what we really mean is that these type-2 formulations are equivalent to the original definitions of these search classes in [JPY88, Pap94b]. In each of the definitions below we take an intersection with $\mathbf{TFNP}$, where $\mathbf{TFNP}$ is the class of total search problems whose defining relations are polynomial-time [MP91, Pap94a]. We take the intersection for a technical reason that $\mathbf{PPA}$, $\mathbf{PPP}$, $\mathbf{PPAD}$, and $\mathbf{PLS}$ are all defined to be subclasses of $\mathbf{TFNP}$ [JPY88, Pap94b, BCE+98]; however, dropping the intersection will not invalidate any results on these classes.

Definition 3.6.

(i) $\mathbf{PPA} = C(\mathbf{LONELY}) \cap \mathbf{TFNP}$. $\mathbf{PPA}$ *stands for* Polynomial Parity Argument.

(ii) $\mathbf{PPP} = C(\mathbf{PIGEON}) \cap \mathbf{TFNP}$. $\mathbf{PPP}$ *stands for* Polynomial Pigeonhole Principle.

(iii) $\mathbf{PPAD} = C(\mathbf{OntoPIGEON}) \cap \mathbf{TFNP}$. $\mathbf{PPAD}$ *stands for* $\mathbf{PPA}$ on Directed graphs.

(iv) $\mathbf{PLS} = C(\mathbf{ITERATION}) \cap \mathbf{TFNP}$. $\mathbf{PLS}$ *stands for* Polynomial Local Search.

(i) and (ii) above are from [BCE+98]. $\mathbf{PPA}$ contains the problems of finding various economic equilibria, some of which are complete. $\mathbf{PPP}$ has relevance in the study of cryptographic hash functions. We believe that the class $C(\mathbf{WeakPIGEON})$ is also relevant to cryp-

tographic hash functions, since $C(\mathbf{WeakPIGEON})$ contains the following problem: given a polynomial-time hash function f that maps N^2 elements to N elements, find a collision. As far as we know, this class has not been given a proper name.

As we have already stated, Beame et. al uses the problem **SOS** to characterize $\mathbf{PPAD}$ while in item (iii) above we use $\mathbf{OntoPIGEON}$. Note that, by Lemma 3.5, $C(\mathbf{SOS}) = C(\mathbf{OntoPIGEON})$. $\mathbf{PPAD}$ contains the problem of finding a Nash equilibrium given pay-off matrices of two player game [Pap94b], which Papadimitriou calls 'a most fundamental computational problem whose complexity is wide open' [Pap01]. Note that the Nash problem is *not* known to be complete for $\mathbf{PPAD}$ [Pap94b].

$\mathbf{PLS}$ is defined to capture the complexity of optimization problems for which efficient local-search heuristics exist. The close connection between $\mathbf{PLS}$ and the iteration principle is stated in the context of bounded arithmetic by Chiari and Krajíček [CK98], based on the work by Buss and Krajíček [BK94]. Item (iv) above is stated in [Mor01], where its equivalence to the original definition of $\mathbf{PLS}$ in [JPY88] is proven explicitly. For more information on these classes, see [JPY88, Yan97] for $\mathbf{PLS}$ and [Pap94b] for the other classes.

Theorem 3.7. *([BCE+98]) The following hold:*
(i) $\mathbf{OntoPIGEON} \leq_m \mathbf{LONELY}$*;*
(ii) $\mathbf{OntoPIGEON} \leq_m \mathbf{PIGEON}$*; and*
(iii) $\mathbf{LONELY}$ *and* $\mathbf{PIGEON}$ *are incomparable, i.e., neither is many-one reducible to the other.*

The above result completely characterizes the relationship among relativized versions of $\mathbf{PPAD}$, $\mathbf{PPA}$, and $\mathbf{PPP}$ via Theorem 3.2. Item (iii) of Theorem 3.7 also follows from our main results below (see Section 3.7). Figure 3.2 depicts the relationships among $\mathbf{PPAD}$, $\mathbf{PPA}$, and $\mathbf{PPP}$ in a generic relativized world. Note that all containments in this figure are proper.

$\mathbf{PLS}$ is not discussed in [BCE+98], and progress for resolving the relative complexity of $\mathbf{PLS}$ is made in [Mor01]:

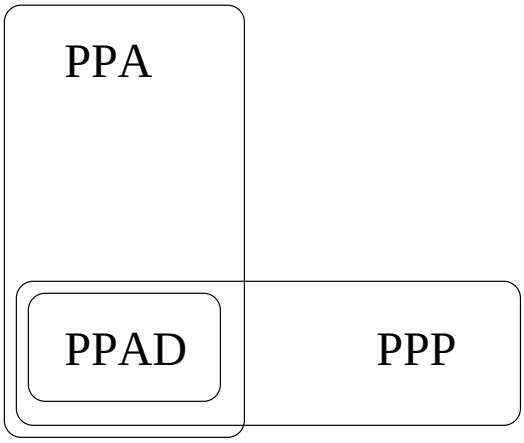

Figure 3.2: The relationships among $\mathbf{PPAD}$, $\mathbf{PPA}$, and $\mathbf{PPP}$ in a generic relativized world [BCE+98].

Theorem 3.8. *([Mor01])* $\mathbf{OntoPIGEON}$ *is not many-one reducible to* $\mathbf{ITERATION}$.

Thus, $\mathbf{PLS}$ contains none of $\mathbf{PPP}$, $\mathbf{PPA}$, and $\mathbf{PPAD}$ in a generic relativized world. We will show two different proofs of the above result: one is based on proof complexity lower bounds (Corollary 3.34) and the other is by the 'separation criterion' for $\mathbf{ITERATION}$ (Theorem 4.1). It was still unresolved in [Mor01] whether $\mathbf{PLS}$ is contained in any of the other classes, and we will present below a partial solution to this question. Note that we still cannot draw $\mathbf{PLS}$ in Figure 3.2, since the relative complexity of $\mathbf{PLS}$ is still not completely resolved.

There is a type-2 problem defined in [BCE+98] that we do not discuss in this dissertation. The problem is called **SINK**, and the motivation for studying this problem is that it characterizes the class $\mathbf{PPADS}$, which is called **PSK** in [Pap94b]. $\mathbf{SINK}$ is defined similarly to $\mathbf{SOS}$ as follows. Its instances are of the form $(succ, pred, 1^n)$, and as in $\mathbf{SOS}$ these instances define directed graphs with indegree and outdegree at most one. $\mathbf{SINK}$ is defined to be the problem of witnessing the following principle:

$$0 \text{ is a source} \supset (\exists x)[x \text{ is a sink}].$$

Thus, $\mathbf{SINK}$ is a stronger (i.e., more difficult) variant of $\mathbf{SOS}$. In fact, Beame et. al prove that

$\mathbf{SOS} \leq_m \mathbf{SINK}$ and $\mathbf{SINK} \not\leq_m \mathbf{SOS}$, and that $\mathbf{SINK} \leq_m \mathbf{PIGEON}$ and $\mathbf{PIGEON} \not\leq_m \mathbf{SINK}$.

The reason why we do not deal with $\mathbf{SINK}$ is that, as far as we know, not much is known about the proof complexity of the tautology family corresponding to the $\mathbf{SINK}$ principle, and thus currently we are not able to derive separations that involve $\mathbf{SINK}$ via our main results.

Finally, let us point out that

$$\mathbf{SINK} \equiv_m \mathbf{OntoPIGEON}^*$$

where $\mathbf{OntoPIGEON}^*$ is defined by the $\exists$-sentence

$$(\forall x)[f(x) \neq 0] \supset (\exists x)[x \neq g(f(x))],$$

which is obtained from the $\mathbf{OntoPIGEON}$ principle by dropping the condition $(\exists x)[x \neq 0 \wedge x \neq f(g(x))]$. The above equivalence is proven in a way completely analogous to Lemma 3.5.

3.3 The Instance Extension Property

In this Section, we introduce the *instance extension property* and show that, if type-2 search problem Q has this property, then any polynomial-time many-one reduction to Q has a simple form. This property turns out to be important in connecting the complexity of search problems and the proof complexity of tautologies in the Nullstellensatz proof system.

Let Q_Φ be a type-2 search problem defined by an $\exists$-sentence Φ. Intuitively, we say that Q_Φ has the *instance extension property* if every instance of Q_Φ can be extended to an arbitrarily large instance so that the solutions in the larger instance correspond to the solutions in the original instance.

Definition 3.9. *Let $Q_\Phi(\alpha_1, \ldots, \alpha_k, 1^n)$ be a type-2 search problem defined by an $\exists$-sentence Φ. We say that Q_Φ has the* instance extension property *iff the following holds: for every*

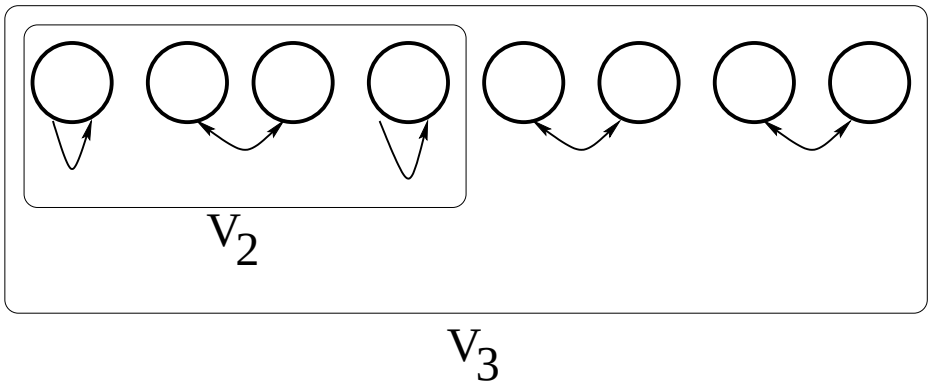

Figure 3.3: Extending an instance V_2 of **LONELY** to V_3.

polynomial p such that $(\forall n)p(n) \geq n$, there exist functions $f, \beta_1, \ldots, \beta_k$ such that, if $w \in Q_\Phi(\beta_1, \ldots, \beta_k, 1^{p(n)})$, then $f(w) \in Q_\Phi(\alpha_1, \ldots, \alpha_k, 1^n)$, where f and β_i for each $1 \leq i \leq k$ are polynomial-time computable using α_i as an oracle.

Lemma 3.10. **LONELY**, **PIGEON**, **ITERATION**, *and* **OntoPIGEON** *have the instance extension property.*

Proof. Let p be a polynomial such that $p(n) > n$ for every $n \geq 1$. An instance $(f, 1^n)$ of **LONELY** can be extended to $(f', 1^{p(n)})$ as follows. For each $v \in V_{p(n)}$, if $v \in V_n$, then $f'(v) = f(v)$; otherwise, if v is even then $f'(v) = v + 1$ and if v is odd then $f'(v) = v - 1$. Every lonely node of $V_{p(n)}$ is in V_n, since every node outside V_n is matched with some other node. Figure 3.3 shows how an instance V_2 of **LONELY** is extended to an instance V_3.

Instances $(f, 1^n)$ of **PIGEON** and **ITERATION** are all extended into larger instances $(f', 1^{p(n)})$ in the following way: $f'(v) = f(v)$ if $v \in V_n$, and $f'(v) = v$ otherwise. For **OntoPIGEON**, we also set $g'(v) = g(v)$ for $v \in V_n$ and $g'(v) = v$ otherwise. □

We do not know whether **WeakPIGEON** has the instance extension property.

The following result shows that, if Ψ has the instance extension property, then every many-one reduction to Q_Ψ is of a relatively simple form.

Lemma 3.11. *Let Q_1 and Q_2 be two total search problems defined by some $\exists$-sentences. Assume that $Q_1 \leq_m Q_2$. If Q_2 has the instance extension property, then Q_1 is many-one reducible to Q_2 by an oracle Turing machine M that does not ask any query to α before asking a query to Q_2.*

Proof. For simplicity, assume that both Q_1 and Q_2 take one unary function as their type-1 argument. We write α and β to denote the type-1 argument of Q_1 and Q_2, respectively. Let $Q_1 \leq_m Q_2$ by a reduction M which, given $(\alpha, 1^n)$, first asks polynomially many queries to α before producing a query $q = (\beta, 1^m)$ to Q_2. Then there exists a polynomial p such that $m \leq p(n)$ for all n and all possible computations of M. We assume without loss of generality that $p(n) > n$.

Define another reduction M' as follows. Given $(\alpha, 1^n)$, it asks a query $r = (\beta', 1^{p(n)})$ to Q_2 without asking any query to α, where β' is computed by a polynomial-time oracle machine $M_{\beta'}$ as follows. Given $v \in V_{p(n)}$, $M_{\beta'}$ first simulates M on $(\alpha, 1^n)$ until it composes a Q_2-query $q = (\beta, 1^m)$. By the instance extension property of Q_2, q can be extended to a larger instance $q' = (\gamma, 1^{p(n)})$. Finally, $M_{\beta'}$ computes $\gamma(v) \in V_{p(n)}$ and set $\beta'(v) = \gamma(v)$. Essentially what is happening here is that the query r is simulating q', which is an extension of q, and q is the query that M would have asked given $(\alpha, 1^n)$.

Note that a solution z' for $Q_2(r)$ is basically a solution for $Q_2(q')$. The function f of the instance extension property extracts from z a solution z for $Q_2(r)$. Then M' can simulate the computation of M after M receives an answer to its Q_2-query. □

Assume that $Q_1 \leq_m Q_2$ and Q_2 has the instance extension property. The above Lemma states that there is a reduction from Q_1 to Q_2 such that the function f in Figure 3.1 on page 38 *does not ask any query to* α.

3.4 Propositional Translations of Type-2 Problems

Let Φ be an $\exists$-sentence over language $\mathcal{L}$. We say that Φ is *basic* if the following conditions are met: it is in prenex form; its quantifier-free part is in DNF; and it contains no nesting of symbols of $\mathcal{L}$. In particular, if Φ is basic, then every atomic formula in Φ is of the form $R(\vec{x})$, $y = f(\vec{x})$, or $x = y$, where R is a predicate symbol and f is a function symbol.

Let Q_Φ be a type-2 search problem defined by a basic $\exists$-sentence Φ of the form

$$\Phi =_{syn} (\exists x_1 \ldots \exists x_k)\phi(x_1, \ldots, x_k).$$

For each $n \geq 1$, we define a propositional tautology $Trans(Q_\Phi, n)$ asserting that Φ holds in every canonical structure V_n. It is the result of a standard translation of Φ into propositional formulas due to Paris and Wilkie [PW85]. The following is a more detailed description of the translation. There will be a set of variables in $Trans(Q_\Phi, n)$ for each type-1 argument α for Q_Φ. If α is an m-ary relation, then, for each m-tuple $\vec{v}$ in the domain of α, there is a propositional variable $X^\alpha_{\vec{v}}$, which is intended to assert that $\alpha(\vec{v})$ is true. If α is a function, we add propositional variables for the relation $graph(\alpha)$: that is, for each $\vec{v}$ in the domain of α and each w in the range V_n of α, we add $X^\alpha_{\vec{v},w}$, which asserts that $\alpha(\vec{v}) = w$.

Let $\alpha_1, \ldots, \alpha_m$ be the function symbols of $\mathcal{L}$ that are not built-in. Then $Trans(Q_\Phi, n)$ is defined as

$$Trans(Q_\Phi, n) =_{syn} (F_{Def(\Phi)} \wedge F_{SingleDef(\Phi)}) \supset F_\Phi$$

which intuitively asserts that Φ holds in V_n if $\alpha_1, \ldots, \alpha_m$ are total and well-defined. First, F_Φ is of the following form:

$$F_\Phi =_{syn} \bigvee_{x_1,\ldots,x_k \in V_n} \phi'(x_1, \ldots, x_k), \tag{3.2}$$

where ϕ' is in DNF with each atomic formula replaced by its corresponding propositional variable or propositional constants. If an atom either contains any built-in predicate or function, or of the form $x = y$, then it is replaced with either $true$ or $false$, depending on their truth value in the canonical structure V_n.

$F_{Def(\Phi)}$ is a CNF formula stating that $\alpha_1, \ldots, \alpha_m$ are total; more specifically, $F_{Def(\Phi)}$ is the following CNF formula:

$$F_{Def(\Phi)} = \bigwedge_{i\in[1,m]} \bigwedge_{\vec{v}\in dom(\alpha_i)} \bigvee_{w\in V_n} X^{\alpha_i}_{\vec{v},w}$$

We denote by $def^{\alpha_i}(\vec{v})$ the clause $\bigvee_{w\in V_n} X^{\alpha_i}_{\vec{v},w}$. Similarly, $F_{SingleDef(\Phi)}$ is the following CNF formula:

$$\bigwedge_{i\in[1,m]} \bigwedge_{\vec{v}\in dom(\alpha_i)} \bigwedge_{w\neq w'} \neg X^{\alpha_i}_{\vec{v},w} \vee \neg X^{\alpha_i}_{\vec{v},w'}.$$

$F_{SingleDef(\Phi)}$ asserts that every function of $\mathcal{L}$ are well-defined.

We define $Trans(Q_\Phi)$ as the family

$$Trans(Q_\Phi) = \{Trans(Q_\Phi, n)\}_{n\geq 1}.$$

In the next section, we will connect the complexity of Q_Φ with the proof complexity of $Trans(Q_\Phi)$.

Finally, we describe how $Trans(Q_\Phi)$ is defined for Φ that is not basic.

Lemma 3.12. *For every $\exists$-sentence Φ, there is another $\exists$-sentence ϕ' over the same language such that (i) Φ' is basic, and (ii) $Q_\Phi \equiv_m Q_{\Phi'}$.*

Proof. If Φ is not basic, we construct a basic Φ' as follows. Pick any atomic subformula ϕ of Φ with nesting of symbols, say $y = f(g(x))$. Replace ϕ with $(\exists z)[z = g(x) \wedge y = f(z)]$ if ϕ is unnegated and with $(\forall z)[z = g(x) \supset y = f(z)]$ if ϕ is negated. Treat the other cases $f(x) = g(y)$ and $R(f(x))$ in a similar way, and repeat this process until all nestings of functions and predicates are removed. Then make the whole sentence prenex with the quantifier-free part in DNF, and call the resulting $\exists$-sentence Φ'. It is clear that $Q_\Phi \equiv_m Q_{\Phi'}$. □

If Φ is not basic, then we define $Trans(Q_\Phi) = Trans(Q_{\Phi'})$ for some Φ' that is constructed as in the above proof.

Below we sum up in the form of a lemma some of the most important properties of the propositional translation that we introduced.

Lemma 3.13. *Let Φ be an $\exists$-sentence over an arbitrary language. The following hold. (i) The number of variables in $Trans(Q_\Phi, n)$ is polynomial in $N = 2^n$. (ii) The size of $Trans(Q_\Phi, n)$ is polynomial in N. (iii) The depth of $Trans(Q_\Phi, n)$ is two.*

Translation Example : Here we describe $Trans(\mathbf{PIGEON}, n)$ as an example of how all this translation works, where $n \geq 1$ is arbitrary. First, we transform the $\mathbf{PIGEON}$ principle on page 41 into basic form, and the result is

$$(\exists x, x', z)[f(x) = 0 \vee (x \neq x' \wedge f(x) = z \wedge f(x') = z)]. \tag{3.3}$$

For each pair $(u, v) \in (V_n)^2$, we have a variable $X^f_{u,v}$ asserting that $f(u) = v$. Let $\phi(x, x', z)$ denote the quantifier-free part of (3.3). Then $Trans(\mathbf{PIGEON}, n)$ is the following DNF formula:

$$\underbrace{\bigvee_{i \in V_n} X^f_{i,0} \vee \bigvee_{i \neq j} \bigvee_{k \in V_n} X^f_{i,k} \wedge X^f_{j,k}}_{(1)} \vee \underbrace{\bigvee_{i \in V_n} \bigwedge_{j \in V_n} \neg X^f_{i,j}}_{(2)} \vee \underbrace{\bigvee_{i \in V_n} \bigvee_{j \neq j'} X^f_{i,j} \wedge X^f_{i,j'}}_{(3)}$$

where (1) is $F_{\mathbf{PIGEON}}$, (2) is the negation of $F_{Def(\mathbf{PIGEON})}$, and (3) is the negation of $F_{SingleDef(\mathbf{PIGEON})}$. This is a standard formulation of the functional pigeonhole principle.

As the above Example shows, $Trans(\mathbf{PIGEON})$ is essentially the *functional pigeonhole principle*, and the proof complexity of the functional pigeonhole principle has been intensely studied; see [PBI93, KPW95, BCE+98, Raz98]. Similarly, $Trans(\mathbf{LONELY})$ corresponds to the parity principle PAR_n of [BP96] in the sense that they have the same proof complexity (up to an application of a polynomial), and $Trans(\mathbf{ITERATION})$ is essentially the *housesitting principle* of [CEI96, Bus98c]. $Trans(\mathbf{WeakPIGEON})$ similarly corresponds to the weak pigeonhole principle whose proof complexity is fairly well known. Finally, we will later show that $Trans(\mathbf{OntoPIGEON})$ has a very low proof complexity in Nullstellensatz (Lemma 3.33).

3.5 Search Trees and Reduction

Let $\Phi = (\exists \vec{x})\phi(\vec{x})$ with $\phi(\vec{x})$ quantifier-free be an $\exists$-sentence over an arbitrary language $\mathcal{L}$. Assume for simplicity that the only input symbol of $\mathcal{L}$ is a unary function α.

Let M be a Turing machine that solves Q_Φ in time $t(n)$. For each n, M gives rise to a *search tree* T_n, which encodes all possible computations of M on $(\alpha, 1^n)$ in terms of α. More specifically, each internal node of T_n denotes a query to α, and this node has an outgoing edge for each of N possible ways the query can be answered. Each path P of T_n corresponds to a computation of M, and the leaf of P is labeled with the output of M in the corresponding computation.

There is a natural correspondence between P and a partial function $\pi_P : V_n \to V_n$ with $|dom(\pi_P)| \leq t(n)$. Let P be a path of T_n and assume that it is labeled with a tuple $\vec{v}$ of elements of V_n. That the tree T_n solves $Q_\Phi(\alpha, 1^n)$ is more formally stated as follows: for every path P of T_n, its leaf label $\vec{v}$ is a solution for $Q_\Phi(\alpha, 1^n)$ specified by π_P, where the notion of π_P specifying a solution is defined as below:

Definition 3.14. *Let $\mathcal{L}$ be a language of which a unary function α is the only input symbol. Let $\Phi = (\exists \vec{x})\phi(\vec{x})$ be an $\exists$-sentence over $\mathcal{L}$. Let $n \geq 1$ be arbitrary and assume that $\pi_n : V_n \to V_n$ be a partial function. We say that π_n* specifies a solution $\vec{v}$ for $Q_\Phi(\alpha, 1^n)$ *iff $\vec{v}$ is a tuple of elements of V_n such that, for every total extension α^* of π_n, the sentence $\phi(\vec{v})$ holds in the structure (V_n, α^*). We also say that π_n* specifies a witness $\vec{v}$ for Φ in V_n.

More generally, if the input symbols of $\mathcal{L}$ are functions $\alpha^1, \ldots, \alpha^k$, then we say that partial functions $\pi_n^1 \ldots, \pi_n^k$ specify a solution $\vec{v}$ for $Q_\Phi(\alpha^1, \ldots, \alpha^k, 1^n)$ iff the following holds: if, for each $i \in [1, k]$, α_n^{i*} is a total extension of π_n^i, then $\phi(\vec{x})$ holds in the structure $(V_n, \alpha_n^{1*}, \ldots, \alpha_n^{k*})$. If $\mathcal{L}$ contains relations, we use their characteristic functions and apply the same definition above.

The *height* of T_n is defined to be the length of the longest paths from the root to a leaf node, and the *size* of T_n is the number of nodes of T_n. From now on, the size and height of T_n are

measured in terms of the size $N = 2^n$ of the underlying set V_n. Thus, in general, if M runs in time $t(n)$ then the height of T_n is $\leq t(\log N)$ and its size is at most $N^{t(\log N)}$. In particular, if M runs in polynomial time, then T_n is of height polylogarithmic in N and size quasipolynomial in N.

Let Q_Φ and Q_Ψ be type-2 search problems defined by $\exists$-sentences Φ and Ψ, and assume that the input symbols of Q_Φ and Q_Ψ are functions $\alpha_1, \ldots, \alpha_k$ and $\beta_1, \ldots, \beta_l$, respectively. Assume that $Q_\Phi \leq_m Q_\Psi$ by a reduction M. Often it becomes useful to split M into two polytime oracle Turing machines M_1 and M_2 in the following way: M_1 is identical to M except that it terminates when M has produced a query $q = (\beta_1, \ldots, \beta_l, 1^m)$ to Q_Ψ; and M_2 simulates the computation of M after it has received an answer w to the query q. More intuitively, M_1 and M_2 are computing f and g of Figure 3.1 on page 38, respectively. Note that M_2's behaviour depends on a computation of M_1, since obviously M's behaviour after the Q_Ψ-query depends on its own behaviour before the query. Thus, in order to specify the computations of M_2 in terms of α, the following need to be fixed: (i) the original inputs to M; (ii) a computation path of M_1 and a query q to Q_Ψ which is asked at the end of the computation; and (iii) w such that $w \in Q_\Psi(q)$.

Let $T_n^{M_1}$ be a search tree encoding the computations of M_1 on $(\alpha_1, \ldots, \alpha_k, 1^n)$; note that its leaves are labeled with queries to Q_Φ. Let P be an arbitrary path of $T_n^{M_1}$ and q be its leaf label. Then for every $w \in Q_\Psi(q)$ we define the search tree $T_n^{M_2}[P, w]$ describing all possible computations of M_2 given P and w. Since M on any computation path outputs a solution for Q_Φ, the following condition is satisfied: If P_1 is a path in $T_n^{M_1}$ ending with a query q to Q_Φ, and if P_2 is a path in $T_n^{M_2}(P_1, w)$ with $w \in Q_\Psi(q)$, then π_{P_1} and π_{P_2} together specify a witness to Φ in V_n.

Note that, if Ψ has the instance extension property, then, by Lemma 3.11, $T_n^{M_1}$ consists of a single leaf node and therefore every path P in $T_n^{M_2}(w)$ specifies a solution for Q_Φ if q is the query that M asks on 1^n and $w \in Q_\Psi(q)$.

Finally, recall that, if the reduction M asks a query $(\beta_1, \ldots, \beta_l, 1^m)$ to Q_Ψ, then $m \leq p(n)$

for some polynomial p and each β_i is polynomial-time computable using oracles $\alpha_1, \ldots, \alpha_k$. For each $\vec{v} \in dom(\beta_i)$, we define $T_m^{\beta_i(\vec{v})}$ to be the search tree for the polynomial-time algorithm computing $\beta_i(\vec{v})$ in terms of $\alpha_1, \ldots, \alpha_k$. The leaves of $T_m^{\beta_i(\vec{v})}$ are labeled with $\beta_i(\vec{v}) = w$ for some $w \in ran(\beta_i)$.

3.6 Relationship between Search Problems and Proof Complexity

In this section we present our results connecting the complexity of Q_Φ and the proof complexity of $Trans(Q_\Phi)$. In order to simplify our presentation, we make the following assumption:

- Φ is over language $\mathcal{L}$ whose only input symbol is a unary function.

However, all the results of this section hold without this assumption.

3.6.1 Bounded-depth PK: Lemmas

Let Φ be an $\exists$-sentence and let M be a Turing machine solving Q_Φ in time $t(n)$. Assume for simplicity that a unary function α is the only input symbol of Φ. As described in the preceding section, M gives rise to the family $\{T_n\}_{n\geq 1}$. Consider the propositional variables $X_{v,w}^{\alpha}$ of $Trans(Q_\Phi)$. Each path P of T_n uniquely corresponds to the conjunction Π_P of variables such that a literal $X_{v,w}^{\alpha}$ appears in Π_P iff P contains an edge that sets $\alpha(v)$ to be w. We will abuse the notation and simply write P to denote both a path of T_n and its corresponding conjunction.

Definition 3.15. *Let $n \geq 1$ and T_n be a tree defined as above. We define $disj(T_n)$ to be the DNF formula $\bigvee_{P\in S} P$, where S is the set of all paths in T_n. The size of $disj(T_n)$ is polynomial in the size of T_n.*

The following lemmas show that certain important facts about the computations of M have short PK-proofs of depth 2. We will use the following three lemmas many times in the next

subsection. First, we show that depth-2 PK has short proofs of the fact that, if α is total, then at least one path in T_n is consistent with α.

Lemma 3.16. *Let $\mathcal{L}$, M, and $\{T_n\}_{n\geq 1}$ be defined as above. For each n, the sequent of the form*

$$F_{Def(\Phi)} \to disj(T_n)$$

has a depth-2 PK-proof whose size is polynomial in the size of the above sequent.

Proof. For every node k of T_n, define $Thru(k)$ to be the cedent of all paths of T_n that go through node k, and $Nodes(k)$ to be the cedent of all α-queries that are asked in the subtree rooted at k. Let R_k be the set of variables corresponding to the path from the root to k. We define S_k to be the sequent

$$S_k : \quad R_k, \bigwedge_{v\in Nodes(k)} def_n^\alpha(v) \to Thru(k).$$

It suffices to derive S_k with k the root node.

We argue that the sequent S_k for every node k of T_n has a desired depth-2 PK-proof; in fact, the derivation is essentially an upside-down copy of T_n itself. First, let k be a leaf node. Then S_k is

$$R_k \to Thru(k),$$

which has a trivial derivation. Now suppose that k is a nonleaf node labeled with, say, a query $\alpha(u)$. Node k has a child node k_i for each $i \in V_n$, which is reachable by the edge specifying $\alpha(u) = i$. Then S_{k_i} is of the following form:

$$X_{u,i}^\alpha, R_k, \bigwedge_{v\in Nodes(k_i)} def_n^\alpha(v) \to Thru(k_i)$$

By weakening and $\wedge$-left, we derive

$$X_{u,i}^\alpha, R_k, \bigwedge_{v\in (Nodes(k)\text{-}u)} def_n^\alpha(v) \to Thru(k). \tag{3.4}$$

Using the sequents (3.4) for every k_i, we derive

$$\bigvee_{i\in V_n} X_{u,i}^\alpha, R_k, \bigwedge_{v\in (Nodes(k)\text{-}u)} def_n^\alpha(v) \to Thru(k)$$

by $\vee$-left. Since $\bigvee_{i\in V_n} X_{u,i}^\alpha$ is $def_n^\alpha(u)$, we are done. □

The next lemma essentially shows that depth-2 PK has short proofs of the fact that, if α is well-defined, no two paths of T_n cannot be simultaneously consistent with α.

Lemma 3.17. *Let $\mathcal{L}$, M, and $\{T_n\}_{n\geq 1}$ be defined as above, and let $n \geq 1$ be arbitrary. Let S_1 and S_2 be two nonempty, disjoint sets of paths of T_n and define $D^i = \bigvee_{P\in S_i} P$ for $i \in \{1, 2\}$. Then the sequent of the form*

$$F_{SingleDef(\Phi)}, D^1, D^2 \rightarrow$$

has depth-2 PK-proof whose size is polynomial in the size of the sequent.

Proof. Let $D^1 = (P_1^1 \vee \ldots P_s^1)$, and let $D^2 = (P_1^2 \vee \ldots \vee P_t^2)$. Pick arbitrary paths P_i^1 and P_j^2 from D^1 and D^2, respectively. Then P_i^1 and P_j^2 must disagree on at least one query since they correspond to two different paths in the same tree. Assume that P_i^1 contains $\alpha(v_{ij}) = w_{ij}$ and that P_j^2, $\alpha(v_{ij}) = w'_{ij}$ with $w_{ij} \neq w'_{ij}$. Then both of the following have trivial derivations:

$$P_i^1, P_j^2 \longrightarrow X^{\alpha}_{v_{ij},w_{ij}} \qquad \text{and} \qquad P_i^1, P_j^2 \longrightarrow X^{\alpha}_{v_{ij},w'_{ij}}.$$

From these sequents we derive

$$\neg X^{\alpha}_{\vec{v}_{ij},w_{ij}} \vee \neg X^{\alpha}_{\vec{v}_{ij},w'_{ij}}, P_i^1, P_j^2 \longrightarrow \tag{3.5}$$

From the sequents (3.5) for every $i \in [1, s]$ and $j \in [1, t]$, we derive

$$\{\neg X^{\alpha}_{(\vec{v}_{ij}),w_{ij}} \vee \neg X^{\alpha}_{(\vec{v}_{ij}),w'_{ij}}\}_{i\in[1,s],j\in[1,t]}, D^1, D^2 \longrightarrow$$

by weakening and $\vee$-left. Using weakening and $\wedge$-left only, we derive

$$F_{SingleDef(\Phi)}, D^1, D^2 \longrightarrow$$

□

Note that, by definition, every path P of T_n is a conjunction of positive literals.

Definition 3.18. *Let P be a path of some search tree T_n, which is a conjunction of some positive literals. Then a* variant *of P is defined to be any conjunction P' of literals (positive*

or negative) satisfying the following: (i) P and P' contain an equal number of literals; (ii) all positive literals of P' are present in P; and (iii) for each negative literal $\neg X^{\alpha}_{v,w}$ of P', P contains a positive literal $X^{\alpha}_{v,w'}$ for some $w' \neq w$.

The following lemma states that, assuming that α is well-founded, bounded-depth PK can convert paths into any of their variants.

Lemma 3.19. *Let $n \geq 1$ and let T_n be defined as in Lemmas 3.16 and 3.17. Let P be an arbitrary path of T_n and P' be a variant of P. Then the sequent*

$$F_{SingleDef(\Phi)}, P \to P'$$

has depth-2 PK-proof whose size is polynomial in the size of the sequent.

Proof. Let P be a path of T_n and assume that P' is a variant of P obtained by substituting $\neg X^{\alpha}_{u_1,w_1}, \ldots, \neg X^{\alpha}_{u_k,w_k}$ for $X^{\alpha}_{u_1,w'_1}, \ldots, X^{\alpha}_{u_k,w'_k}$, respectively. For each $i \in [1,k]$,

$$F_{SingleDef(\Phi)}, X^{\alpha}_{u_i,w_i} \to \neg X^{\alpha}_{u,w'_i}$$

has short derivations. From these sequents, we derive

$$F_{SingleDef(\Phi)}, X^{\alpha}_{u_1,w'_1}, \ldots, X^{\alpha}_{u_k,w'_k} \to \neg X^{\alpha}_{u_1,w_1} \wedge \ldots \wedge \neg X^{\alpha}_{u_k,w_k}$$

by weakening and $\wedge$-right. It is easy to derive the desired sequent from the above. □

Let A and B be arbitrary conjunctions of literals. We say that A contains B if every literal of B occurs in A. Let P be a path of T_n and assume that $\vec{v}$ is the solution output at the end of the computation corresponding to P. Thus, for every total extension α^* of the partial function π_P, $\phi(\vec{v})$ is true in structure (V_n, α^*), where $\phi(\vec{x})$ is a quantifier-free formula such that $\Phi =_{syn} (\exists \vec{x})\phi(\vec{x})$. As in (3.2), let $\phi'(\vec{v})$ be the propositional DNF formula obtained by replacing the atomic formulas of $\phi(\vec{v})$ by either a logical constant or a variable in an appropriate way.

Lemma 3.20. *Let $n \geq 1$ and T_n, P, and $\phi'(\vec{v})$ be defined as the paragraph above. Then there is a variant P' of P that contains a term of $\phi'(\vec{v})$.*

Proof. Assume that $\phi'(\vec{v})$ is of the form

$$\phi'(\vec{v}) =_{syn} T_1 \vee \ldots \vee T_c$$

where $T_i(\vec{v})$ for each $i \in [1, c]$ is a term, i.e., a conjunction of literals.

Assume, for the sake of contradiction, that, for every $i \in [1, c]$, the term T_i is not contained in any variant of P. Thus, for every T_i, at least one of the following holds: (i) T_i contains a positive literal $X^{\alpha}_{v,w}$ that does not occur in P; and (ii) T_i contains a negative literal $\neg X^{\alpha}_{v,w}$ such that, for every $w' \in V_n$, $X^{\alpha}_{v,w'}$ does not occur in P. Define a total extension α^* of π_P as follows. For each T_i, if condition (i) holds, then $\alpha^*(v) = z$ for some $z \neq w$; and if condition (ii) holds, then $\alpha^*(v) = w$. The rest of α^* is defined in an arbitrary way. It is clear that $\vec{v}$ is not a solution in structure (V_n, α^*), and a contradiction is reached. □

3.6.2 Bounded-Depth PK: Results

We now present one of our main results that connect the complexity of search problems and the proof complexity of the corresponding combinatorial principles in bounded-depth PK. We first describe a simpler result relating the time complexity of Q_Φ and the proof complexity of $Trans(Q_\Phi)$, where Φ is an $\exists$-sentence that holds in every canonical structure. This is based on the intuitive idea that an algorithm that solves Q_Φ is a highly constructive proof of Φ. We formalize this idea by showing how to turn an algorithm for Q_Φ into a bounded-depth PK-proof of $Trans(Q_\Phi)$. This is not difficult, since we have already shown in the preceding section that bounded-depth PK has short proofs of some of the important properties of the search tree T_n.

Theorem 3.21. *Let Φ be an $\exists$-sentence, and let M be a deterministic Turing machine that solves Q_Φ in time $t(n)$. Then $Trans(Q_\Phi, n)$ has depth-2 PK-proofs whose size is polynomial in $max\{m_1, m_2\}$, where m_1 is the size of $Trans(Q_\Phi, n)$ and m_2 is the size of T_n.*

Proof. It suffices to prove that

$$F_{SingleDef(\Phi)}, F_{Def(\Phi)} \to F_\Phi \tag{3.6}$$

has a depth-2 PK proof of the desired size for every $n \geq 1$.

Let P be an arbitrary path of T_n. By Lemma 3.20, P has a variant P' that contains a term of F_Φ, and therefore we have a short derivation of

$$P' \to F_\Phi.$$

On the other hand,

$$F_{SingleDef(\Phi)}, P \to P'$$

has short PK-proofs by Lemma 3.19. From these two sequent, we derive by a cut on P' the sequent

$$F_{SingleDef(\Phi)}, P \to F_\Phi \tag{3.7}$$

Using the sequent (3.7) for each path P of T_n, we can derive

$$F_{SingleDef(\Phi)}, disj(T_n) \to F_\Phi \tag{3.8}$$

by $\vee$-left only. The desired sequent (3.6) follows by a cut from (3.8) and

$$F_{Def(\Phi)} \to disj(T_n)$$

which has short proofs by Lemma 3.16. □

Clearly, the assumption of Theorem 3.21 can be weakened to the existence of a nonuniform family of trees T_n solving Q_Φ.

Theorem 3.21 could be used to obtain a lower bound on the height (and, in turn, size) of T_n via a lower bound on $Trans(Q_\Phi)$ in depth-2 PK; however, it is often much harder to show a proof complexity lower bound than a lower bound for the complexity of Q_Φ. Theorem 3.21 may be more useful in obtaining upper bound on the proof lengths of $Trans(Q_\Phi)$: it now suffices to demonstrate an algorithm that solves Q_Φ.

Now we present the first main result of this Chapter. The intuitive idea behind this is that a many-one reduction from Q_Φ to Q_Ψ is itself a constructive proof that Ψ implies Φ, and we formalize this idea in bounded-depth PK. We adopt the following definition for comparing the proof complexity of different families of propositional formulas.

Definition 3.22. *Let $\mathcal{S}, \mathcal{R}$ be two families of propositional formulas. We say $\mathcal{S}$ has a quasipolynomial $BDPK$ reduction to $\mathcal{R}$, written $\mathcal{S} \leq_{BDPK} \mathcal{R}$, if $\mathcal{S}$ has quasipolynomial-size, bounded-depth PK-proofs in which any substitution instance of $\mathcal{R}$ is allowed as an initial sequent.*

If $\mathcal{S} \leq_{BDPK} \mathcal{R}$, then the existence of bounded-depth quasipolynomial-size PK-proofs of $\mathcal{R}$ implies that $\mathcal{S}$ also has quasipolynomial-size PK-proofs of bounded depth. We present a result connecting a many-one reduction between search problems and a quasipolynomial $BDPK$ reduction between combinatorial principles.

Theorem 3.23. *Let Φ and Ψ be two first-order $\exists$-sentences, and assume that Q_Ψ has the instance extension property. If $Q_\Phi \leq_m Q_\Psi$, then $Trans(Q_\Phi) \leq_{BDPK} Trans(Q_\Psi)$.*

Proof. For simplicity, we assume that the type-1 arguments of Q_Φ and Q_Ψ are unary functions α and β, respectively. A proof for a more general case is analogous. We use X with subscripts to denote the variables of $Trans(Q_\Phi)$ and Y with subscripts for the variables of $Trans(Q_\Psi)$.

Since Ψ satisfies the instance extension property, by Lemma 3.11, $Q_\Phi \leq_m Q_\Psi$ by a polynomial-time oracle Turing machine M whose query $q = (\beta, 1^m)$ only depends on n, the length of the string argument of Q_Φ.

Fix n arbitrarily. Recall the definition in Section 3.5 of the search tree $T_m^{\beta(v)}$ computing the value of $\beta(v)$. For each $w \in V_m$, we partition the paths of $T_m^{\beta(v)}$ into two sets S and S', where S consists of all the paths whose leaves are labeled $\beta(v) = w$, and S' contains all the other paths. We define $DNF(Y^\beta_{v,w}) = \bigvee_{P \in S} P$ and $DNF(\neg Y^\beta_{v,w}) = \bigvee_{P \in S'} P$.

We write $Trans(Q_\Psi, m)[Y/T]$ to denote the substitution instance of $Trans(Q_\Psi, m)$ in which the literals $Y^\beta_{v,w}$ and $\neg Y^\beta_{v,w}$ are replaced by the DNF formulas $DNF(Y^\beta_{v,w})$ and $DNF(\neg Y^\beta_{v,w})$, respectively. Note that, because of Lemmas 3.16 and 3.17, PK proves the equivalence of

$DNF(Y^{\beta}_{v,w})$ and $\neg DNF(\neg Y^{\beta}_{v,w})$ from $F_{Def(\Phi)}$ and $F_{SingleDef(\Phi)}$; thus, we will ignore the syntactic difference of the two.

We claim that PK has bounded-depth quasi-polynomial size proofs of

$$Trans(Q_\Psi, m)[Y/T] \rightarrow Trans(Q_\Phi, n) \tag{3.9}$$

from which the theorem follows. Note that this sequent has depth 3, and no formula of higher depth will appear in the proof. Denote $Trans(Q_\Phi, n)$ and $Trans(Q_\Psi, m)$ as

$$(F_{Def(\Phi)} \wedge F_{SingleDef(\Phi)}) \supset F_\Phi \text{ and } (F_{Def(\Psi)} \wedge F_{SingleDef(\Psi)}) \supset F_\Psi,$$

respectively. Then it suffices to show that the following three sequents have quasi-polynomial size depth-3 PK proofs:

$$\begin{array}{ll} (A) & F_{Def(\Phi)} \rightarrow F_{Def(\Psi)}[Y/T] \\ (B) & F_{SingleDef(\Phi)} \rightarrow F_{SingleDef(\Psi)}[Y/T] \\ (C) & F_{Def(\Phi)}, F_{SingleDef(\Phi)}, F_\Psi[Y/T] \rightarrow F_\Phi \end{array}$$

Below we show that each of the above sequents have bounded-depth PK proofs of appropriate size.

(A) *If α is total, then so is β.* Since $F_{Def(\Psi)}$ is the conjunction of $def^\beta(v)$ for every $v \in V_m$, it is enough to show that

$$F_{Def(\Phi)} \rightarrow def^\beta(v)[Y/T]$$

has a PK proof of an appropriate size and depth. This is immediate from Lemma 3.16, since $def^\beta(v)[Y/T]$ is simply $disj(T_m^{\beta(v)})$.

(B) *If α is well-defined, then β is well-defined.* Recall that $F_{SingleDef(\Psi)}$ is the conjunction of $\neg Y^\beta_{v,w} \vee \neg Y^\beta_{v,w'}$ for every $w, w' \in V_n$ with $w \neq w'$. Thus, it suffices to derive

$$F_{SingleDef(\Phi)}) \rightarrow \neg(Y^\beta_{v,w}[Y/T]), \neg(Y^\beta_{v,w'}[Y/T]). \tag{3.10}$$

This is immediate from Lemma 3.17, since $Y^\beta_{v,w}[Y/T]$ and $Y^\beta_{v,w'}[Y/T]$ are disjoint sets of paths of $T_m^{\beta(v)}$.

(C) *If we find a solution to Q_Ψ, then we can find one for Q_Φ.* Recall that Ψ is in DNF and therefore F_Ψ is also in DNF. Let C be a term of F_Ψ of the form $C =_{syn} \bigwedge_{i\in[1,t]} l_i$, where each literal l_i is either the variable $Y^\beta_{v_i,w_i}$ or its negation. Note that C is asserting that a specific tuple $\vec{z}$ is a witness to Ψ in V_m.

Our goal is to show that the following sequent has quasipolynomial-size, bounded-depth PK proofs:

$$F_{Def(\Phi)}, F_{SingleDef(\Phi)}, C[Y/T] \to F_\Phi \tag{3.11}$$

Note that sequent (C) is derivable from sequents (3.11) for every term C of F_Ψ by $\vee$-left.

For each $i \in [1,t]$, let $T_i = T_m^{\beta(v_i)}$. Define $\mathcal{P}$ to be the set of conjunctions

$$\mathcal{P} = \{(P_1 \wedge \cdots \wedge P_t) \mid P_1 \in T_1, \ldots, P_t \in T_t\},$$

where $P_i \in T_i$ means that P_i is a path in T_i. The sequent $\to \mathcal{P}$ is essentially $\to disj(T)$, where T is the tree that starts with T_1 whose leaves are replaced with copies of T_2 whose leaves are replaced with T_3, and so on. By Lemma 3.16, we have the sequent

$$F_{Def(\Phi)} \to \mathcal{P} \tag{3.12}$$

A path is *inconsistent* if it contains both $X^\alpha_{v,w}$ and $X^\alpha_{v,w'}$ for some v and $w \neq w'$. Our next goal is to remove all the inconsistent paths from the succedent of (3.12). This is done as follows. Let P be an inconsistent path. Then P can be written as $P' \wedge X^\alpha_{v,w} \wedge X^\alpha_{v,w'}$. By Lemma 3.19, we have

$$F_{SignleDef}, P' \wedge X^\alpha_{v,w} \wedge X^\alpha_{v,w'} \to P' \wedge X^\alpha_{v,w} \wedge (\neg X^\alpha_{v,w}).$$

Since $P' \wedge X^\alpha_{v,w} \wedge (\neg X^\alpha_{v,w}) \to$ has short proofs from logical axioms, we derive

$$F_{SignleDef}, P' \wedge X^\alpha_{v,w} \wedge X^\alpha_{v,w'} \to .$$

By a cut on $P = P' \wedge X^\alpha_{v,w} \wedge X^\alpha_{v,w'}$, we remove P from the succedent of (3.12). Thus, we have

$$F_{Def(\Phi)}, F_{SingleDef(\Phi)} \to \mathcal{P}' \tag{3.13}$$

where $\mathcal{P}'$ denotes the set of all consistent paths of T.

Recall that $C = \bigwedge_{i\in 1,t]} l_i$ and it is asserting that some $\vec{z}$ is a witness to Ψ. For every path $P = (P_1 \wedge \ldots \wedge P_t)$ of $\mathcal{P}'$, we say that P makes $\vec{z}$ a solution iff, for every $i \in [1,t]$, P_i witnesses l_i; that is, P_i is labeled with $\beta(v_i) = w_i$ if l_i is a positive literal $Y^{\beta}_{v_i,w_i}$ and $\beta(v_i) = w_i'$ for $w_i \neq w_i'$ if l_i is a negative literal $\neg Y^{\beta}_{v_i,w_i}$. Now we partition $\mathcal{P}'$ into two sets ${\mathcal{P}'}_{Sol}$ and ${\mathcal{P}'}_{NonSol}$, where $\mathcal{P}'_{Sol}$ consists of all the paths that makes $\vec{z}$ a solution and ${\mathcal{P}'}_{NonSol}$ contains the rest of the paths.

Let $P \in {\mathcal{P}'}_{Sol}$ and recall that $T^{M_2}_n[\vec{z}]$ is the search tree that encodes all possible computations of M after it receives $\vec{z} \in Q_\Psi(\beta, 1^m)$. Let R be an arbitrary path of $T^{M_2}_n[\vec{z}]$. If R is inconsistent with P, then the sequent $P, R \longrightarrow$ has trivial proofs. If R is consistent with P, then, by the definition of M_2, R specifies a solution for Q_Φ, and by the method of Lemma 3.19 and Theorem 3.21 we derive $F_{SingleDef(\Phi)}, P, R \rightarrow F_\Phi$. Thus, from these sequents for R, we derive

$$F_{SingleDef(\Phi)}, P, disj(T^{M_2}_n[w]) \rightarrow F_\Phi \tag{3.14}$$

by weakening and $\vee$-left. Since we have (3.14) for each $P \in {\mathcal{P}'}_{Sol}$, we can derive

$$F_{SingleDef(\Phi)}, \bigvee {\mathcal{P}'}_{Sol}, disj(T^{M_2}_n[w]) \rightarrow F_\Phi \tag{3.15}$$

where $\bigvee {\mathcal{P}'}_{Sol}$ is the disjunction of the paths in ${\mathcal{P}'}_{Sol}$. By Lemma 3.16 we have $F_{Def(\Phi)} \rightarrow disj(T^{M_2}_n[w])$. From this sequent and sequents (3.13) and (3.15), we derive

$$F_{Def(\Phi)}, F_{SingleDef(\Phi)} \rightarrow F_\Phi, {\mathcal{P}'}_{NonSol} \tag{3.16}$$

Let $P = (P_1 \wedge \ldots \wedge P_t)$ be a path in ${\mathcal{P}'}_{NonSol}$. Since P does not make $\vec{z}$ a solution, it follows that there is $i \in [1,t]$ such that P_i is inconsistent with every term in $l_i[Y/T]$. Thus, the sequent

$$P_i, l_i[Y/T] \rightarrow$$

has short proofs by repeated applications of $\vee$-left, and from this it is easy to derive

$$P_i, C[Y/T] \rightarrow$$

by weakening and $\wedge$-left. Since we have this sequent for every $P \in \mathcal{P}'_{NonSol}$, by repeated applications of $\vee$-left we can derive

$$\mathcal{P}'_{Nonsol}, C[Y/T] \rightarrow \tag{3.17}$$

Finally, a cut on $\mathcal{P}'_{Nonsol}$ using (3.16) and (3.17) produces (3.11). □

In Section 3.7, we obtain relative separations of search classes, one of which is previously unknown, via Theorem 3.23. We also discuss its connections to bounded arithmetic in Section 3.7.

Finally, we sketch how to prove a slightly stronger form of Theorem 3.23 which does not assume that Q_Ψ has the instance extension property.

Theorem 3.24. *Theorem 3.23 continues to hold without the assumptions that Q_Ψ has the instance extension property.*

Proof. Assume that $Q_\Phi \leq_m Q_\Psi$ by a reduction M that asks polynomially many queries to α before asking a query to Q_Ψ. Let $n \geq 1$ be arbitrary and let M_1 be a deterministic machine that simulates M until it produces a query to Q_Φ, and let $T_n^{M_1}$ be the corresponding search tree. Let $U_1, \ldots, U_k$ be all the paths of $T_n^{M_1}$, and for each $i \in [1, k]$, let $q_i = (\beta_i, 1^{m_i})$ be the corresponding Q_Ψ-query. Note that k is the number of paths of T_n, which is quasipolynomial in N. The query q_i gives rise to the tautology $Trans(Q_\Psi, m_i)[Y/T_i]$ that asserts the totality of $Q_\Psi(\beta_i, 1^{m_i})$. We claim that

$$Trans(Q_\Psi, n_1)[Y/T_1], \ldots, Trans(Q_\Psi, n_k)[Y/T_k] \rightarrow Trans(Q_\Phi, n) \tag{3.18}$$

has short PK proofs of bounded depth. Let us write $Trans(Q_\Psi, m_i)[Y/T]$ as

$$(F^i_{Def(\Psi)} \wedge F^i_{SingleDef(\Psi)}) \supset F^i_\Psi,$$

where the superscript i indicates the dependency on path U_i of $T_n^{M_1}$. It is not hard to see that sequent (3.18) has a short derivation from the following sequents:

(D_i) $F_{Def(\Phi)}, F_{SingleDef(\Phi)} \to F^i_{Def(\Psi)} \wedge F^i_{SingleDef(\Psi)}$ for each $i \in [1,k]$; and

(E) $F^1_\Psi, \ldots, F^k_\Psi, F_{Def(\Phi)}, F_{SingleDef(\Phi)} \to F_\Phi$.

The short proofs for (D_i)'s are already constructed in the proof of Theorem 3.23. (E) has the following derivation, where Δ denotes $\{F_{Def(\Phi)}, F_{SingleDef(\Phi)}\}$ and the double line abbreviates multiple applications of weakening followed by a cut:

$$\dfrac{\dfrac{U_1, F^1_\Psi, \Delta \to F_\Phi \qquad \dfrac{U_2, F^2_\Psi, \Delta \to F_\Phi \qquad \dfrac{\dfrac{\dfrac{U_m, F^k_\Psi, \Delta \to F_\Phi \qquad \dfrac{\Delta \to U_1, \ldots, U_k}{F^m_\Psi, \Delta \to F_\Phi, U_1, \ldots, U_k}\text{ weakening}}{F^m_\Psi, \Delta \to F_\Phi, U_1, \ldots, U_{k-1}}}{\vdots}}{F^2_\Psi, \ldots, F^m_\Psi, \Delta \to F_\Phi, U_1, U_2}}{F^2_\Psi, \ldots, F^m_\Psi, \Delta \to F_\Phi, U_1}}{F^1_\Psi, \ldots, F^k_\Psi, \Delta \to F_\Phi}$$

Note that the upper right sequent of the above derivations is $\Delta \to disj(T^{M_1}_n)$, for which we have bounded-depth PK proofs by Lemma 3.16. Thus, it remains to show that, for each $i \in [1,k]$, the sequent

$$U_i, F^i_\Psi, \Delta \to F_\Phi$$

has short proofs. This is done in a way analogous to the case (C) of the proof of Theorem 3.23. □

3.6.3 Nullstellensatz: Preliminaries

We saw above connections between the complexity of search problems and the proof lengths of combinatorial principles in bounded-depth PK. We present similar connections with respect to another proof system, Nullstellensatz.

Nullstellensatz is an algebraic proof system (actually, a refutation system) that operates on polynomials. Let F be a field and let $\vec{X}$ be a set of variables. Given polynomials $q_1, \ldots, q_m, p \in F[\vec{X}]$, a *Nullstellensatz derivation* of p from $q_1, \ldots, q_m$ is another set of polynomials $r_1, \ldots, r_m \in F[\vec{X}]$ such that

$$r_1 q_1 + \cdots + r_m q_m = p,$$

identically. *Nullstellensatz refutation* of $q_1, \ldots, q_m$ is a Nullstellensatz derivation of 1 from $q_1, \ldots, q_m$. The following shows why this is called a refutation:

Theorem 3.25. *Let F be an arbitrary field, and let $q_1, \ldots, q_m \in F[\vec{X}]$. Then the equations $q_1 = 0, \ldots, q_m = 0$ do not have a solution in the algebraic closure of F if and only if there is a Nullstellensatz refutation of $q_1, \ldots, q_m$.*

Via a standard translation of propositional formulas into polynomials (see below), Nullstellensatz is turned into a refutation system showing that the given propositional formula is unsatisfiable. The complexity measure of Nullstellensatz refutations is the *degree*, which is the maximum over the degrees of $r_i q_i$. See [CEI96, Raz98, Bus98c] for more information on Nullstellensatz. Nullstellensatz is derivationally sound and complete over any field F in the sense that p can be derived iff it is in the ideal generated by $q_1, \ldots, q_m$. Throughout this section, we work in an arbitrary field.

Let Φ be a basic $\exists$-sentence. We show how to construct a family of unsatisfiable sets of polynomials. Let $n \geq 1$ be arbitrary and consider the following unsatisfiable CNF formula expressing the negation of $Trans(Q_\Phi, n)$:

$$F_{Def(\Phi)} \wedge F_{SingleDef(\Phi)} \wedge F_{\neg\Phi},$$

where $F_{\neg\Phi}$ is the CNF formula obtained from $\neg F_\Phi$ by pushing negations inward.

We convert the above CNF formula into an unsatisfiable set $poly(\neg Q_\Phi, n)$ such that

$$poly(\neg Q_\Phi, n) = poly_{\neg\Phi} \cup poly_{Def(\Phi)} \cup poly_{SingleDef(\Phi)} \cup poly_{0/1}.$$

The negation sign in front of Q_Φ is meant to indicate that $poly(\neg Q_\Phi, n)$ is asserting that there is no solution for Q_Φ in V_n.

- The variables of $poly(\neg Q_\Phi, n)$ are the variables of $Trans(Q_\Phi, n)$.
- The set $poly_{\neg\Phi}$ contains a polynomial for each clause C of $F_{\neg\Phi}$ that is obtained in the following way: each literal of C forms a linear factor of the polynomial, where a positive literal X becomes a factor $(1 - X)$ and a negative literal $\neg X$ becomes a factor X.

- For each clause $def_n^{\alpha_i}(\vec{v})$ of $F_{Def(\Phi)}$, $poly_{Def(\Phi)}$ contains a polynomial

$$\sum_{w \in V_n} X^{\alpha}_{\vec{v},w} - 1$$

- Similarly, $poly_{SingleDef(\Phi)}$ consists of polynomials

$$X^{\alpha}_{\vec{v},w} X^{\alpha}_{\vec{v},w'}$$

  for each clause $\neg X^{\alpha}_{\vec{v},w} \vee \neg X^{\alpha}_{\vec{v},w'}$ of $F_{SingleDef(\Phi)}$.

- Finally, $poly_{0/1}$ is there to force each variable X to take on 0/1 values; i.e., there is a polynomial

$$X - X^2$$

  for each variable X.

We define $poly(\neg Q_\Phi)$ as

$$poly(\neg Q_\Phi) = \{poly(\neg Q_\Phi, n)\}_{n \geq 1}.$$

If Φ is not basic, we define $poly(\neg Q_\Phi)$ in the way analogous to $Trans(Q_\Phi)$. Note that the maximum degree of $poly(\neg Q_\Phi)$ is a constnat that depends on Φ.

3.6.4 Nullstellensatz: Results

Let $\mathcal{L}$, M, $\{T_n\}$ be as in section 3.6.1. For each path P in T_n, we form a monomial as follows: if the query $\alpha(v) = w$ appears in P, then the variable $X^{\alpha}_{v,w}$ appears in the monomial. Clearly, the degree of the monomial is the length of P. We will abuse notation and use P to refer to both the path and the monomial. We associate a polynomial, $poly(T_n)$ with the tree T_n—namely, the sum of the monomials for each path in T_n. The polynomial $poly(T_n)$ has degree equal to the height of the tree T_n.

Lemma 3.26. *Let $\mathcal{L}$, M, $\{T_n\}$ be as above. For each n, the polynomial $poly(T_n) - 1$ has a Nullstellensatz derivation from $poly_{Def}$ of degree at most the height of T_n.*

Proof. Fix n. We prove induction on the height of T_n that $poly(T_n)$ has a derivation of degree at most the height of T_n. If T_n is of height 1, then T_n has one internal node connected to N leaves. Let u be the query. Then

$$poly(T_n) - 1 = \sum_{w \in V_n} X^{\alpha}_{\vec{v},w} - 1,$$

which is a polynomial in $poly_{Def(\Phi)}$.

Suppose that T_n has height $k > 1$. Consider the tree T', the subtree of T_n where every path from the root is truncated at length $k-1$. By induction, $poly(T') - 1$ has a Nullstellensatz derivation of degree $k-1$. Consider any leaf l of T' that is not a leaf of T and assume it queries α on u in T_n. Let T'_l be the tree T' with every T_n-child of l added on. Then

$$poly(T'_l) - poly(T') = P_l(\sum_{w \in V_n} X^{\alpha}_{\vec{v},w} - 1),$$

where P_l is the path from the root to l. We know $poly(T_n) - poly(T')$ is just the sum of $poly(T'_l) - poly(T')$ for all such leaves l. Hence $poly(T_n)$ has a degree k Nullstellensatz derivation. □

Let P be a path (i.e., a monomial) of some search tree T_n specifying a solution $\vec{v}$ for Q_Φ, where $\Phi =_{syn} (\exists \vec{x})\phi(\vec{x})$. We define a *mutant* of P to be any polynomial of the form $P \cdot R$, where R is of the form $(1 - X^{\alpha}_{v_1,w'_1}) \cdots (1 - X^{\alpha}_{v_m,w'_m})$ such that, for each $i \in [1, m]$, the variable $X^{\alpha}_{v_i,w_i}$ is a factor of P with $w_i \neq w'_i$. We also define P to be a mutant of itself.

Let P' be a mutant of a path P. We say that P' contains a polynomial R if every linear factor of R is also a linear factor of P'.

Lemma 3.27. *Let P be a path of T_n specifying a solution $\vec{v}$ for Q_Φ, where Φ is defined as above. Then then P has a mutant P' that contains a polynomial from $poly_{\neg\Phi}$.*

Proof. This follows from the same argument as in the proof of the similar Lemma for bounded-depth PK (Lemma 3.20). □

The following is a Nullstellensatz analogue of Theorem 3.21, linking the time complexity of Q_Φ and the degree of Nullstellensatz refutations of $poly(\neg Q_\Phi)$.

Theorem 3.28. *Let Φ be an $\exists$-sentence, and let M be a deterministic Turing machine that solves Q_Φ in time $t(n)$. Then $poly(\neg Q_\Phi, n)$ has Nullstellensatz refutations of degree $t(\log N)$.*

Proof. (Sketch) This is proven analogously to Theorem 3.21 for depth-2 PK. By Lemma 3.26, there is a derivation of $poly(T_n) - 1$. Our goal is to cancel out each path P of $poly(T_n)$. By Lemma 3.27, each path P of $poly(T_n)$ has a mutant P' that contains a polynomial from $poly_{\neg\Phi}$, from which one instance of P' is easily derived.

It suffices to show how to derive another instance of P'. Assume that P' is of the form $P \cdot (1 - X_1) \cdots (1 - X_m)$, where variable X_i is a factor of P for each $i \in [1, m]$. It is easy to see that there exists a polynomial R derivable from $poly_{SingleDef(\Phi)}$ such that $P' = P - R$. □

Definition 3.29. *Let F be a field and let $\vec{X}$ and $\vec{Y}$ be infinite sets of variables. Let P_1 be an infinite family of finite subsets of $F[\vec{X}]$ and let P_2 be an infinite family of finite subsets of $F[\vec{Y}]$. We say that P_1 has a polylogarithmic Nullstellensatz reduction to P_2 and write $P_1 \leq_{HN} P_2$ if, for any $A \in P_1$ there is a $B \in P_2$ and a set of polynomials $\bar{f}_Y = \{f_Y\}_{Y \in \vec{Y}} \subset F[\vec{X}]$ such that each polynomial in $B[Y/f_Y]$ has a polylogarithmic-degree Nullstellensatz derivation from A, where $B[Y/f_Y]$ is the substitution instance of B in which every variable Y of B is replaced by a polynomial f_Y.*

If $\mathcal{S} \leq_{HN} \mathcal{R}$, then the existence of polylog-degree Nullstellensatz refutations of $\mathcal{R}$ implies that $\mathcal{S}$ also has polylog-degree refutations. We relate a many-one reduction between search problems and a polylogarithmic Nullstellensatz reduction between combinatorial principles encoded as sets of polynomials.

Theorem 3.30. *Let Φ and Ψ be two first-order $\exists$-sentences and assume that Q_Ψ has the instance extension property. If $Q_\Phi \leq_m Q_\Psi$, then $poly(Q_\Phi) \leq_{HN} poly(Q_\Psi)$ over any field.*

Proof. As usual, for simplicity, we assume that the type-1 arguments of Q_Φ and Q_Ψ are unary functions α and β, respectively. We use X with subscripts to denote the variables of $Trans(Q_\Phi)$ and Y with subscripts for the variables of $Trans(Q_\Psi)$.

Since Ψ satisfies the instance extension property, by Lemma 3.11, $Q_\Phi \leq_m Q_\Psi$ by a polynomial-time oracle Turing machine M whose query $q = (\beta_1, \ldots, \beta_{k'}, 1^m)$ only depends on 1^n, the string argument of Q_Φ. Fix n arbitrary. Recall that $poly(T_m^{\beta(v)})$ is the sum of all paths of the tree $T_m^{\beta(v)}$ computing the value of $\beta(v)$. For each $w \in V_m$, define $poly(Y_{v,w}^\beta)$ to be the sum of all paths of $T_m^{\beta(v)}$ asserting that $\beta(v) = w$, and define $poly(1 - Y_{v,w}^\beta)$ to be the sum of all other paths of $T_m^{\beta(v)}$. Thus,

$$poly(T_m^{\beta(v)}) = poly(Y_{v,w}^\beta) + poly(1 - Y_{v,w}^\beta).$$

Because of Lemma 3.26, there is an equivalence in Nullstellensatz between $1 - poly(Y_{v,w}^\beta)$ and $poly(1 - Y_{v,w}^\beta)$. Therefore, we will ignore the syntactic difference between the two. Define $poly(Q_\Psi, m)[Y/T]$ as the result of substituting for each variable $Y_{v,w}^\beta$ the polynomial $poly(Y_{v,w}^\beta)$.

Below we show that Nullstellensatz derives from $poly(\neg Q_\Phi, n)$ the substitution instance $poly(\neg Q_\Psi, m)[Y/T]$ and that the derivation is of degree polylogarithmic in N. It suffices to show that, for each polynomial τ in $poly(\neg Q_\Psi, m)$, its substitution instance $\tau[Y/T]$ has a polylogarithmic degree derivation from the polynomials of $poly(\neg Q_\Phi, n)$. Four cases arise.

(A) *If α is total, then so is β.* If τ is $poly_{Def(\Psi)}(\beta(v))$, then $\tau[Y/T]$ is $poly(T_m^{\beta(v)}) - 1$, which has a desired derivation by Lemma 3.26.

(B) *If α is well-defined, then so is β.* Let τ is in $poly_{SingleDef(\Psi)}$. Then τ is of the form $Y_{v,w}^\beta Y_{v,w'}^\beta$ for some $w \neq w'$. Every term t in $\tau[Y/T]$ is the product of two different paths in $T_m^{\beta(v)}$, so something in $poly_{SingleDef(\Phi)}$ is a factor of this term.

(C) *If we find a solution to Q_Ψ, then we can find one for Q_Φ.* This is analogous to the corresponding case in the proof of Theorem 3.23. Note that we handle mutants of paths in the way we did in the proof of Theorem 3.28.

(D) *The variables of $poly(\neg Q_\Psi)$ are boolean.* Let $\tau \in poly_{0/1}$ of the form $Y_{v,w}^\beta - Y_{\beta(v),w}^2$). But then $\tau[Y/T]$ is just $poly(Y_{v,w}^\beta)poly(1 - Y_{v,w}^\beta)$, so every term is the product to two different paths in the same tree. Hence, every term is derivable from something in $poly_{SingleDef(\Phi)}$.

It is clear that none of these four cases involves high degree polynomials in any way. □

We do not know how to prove Theorem 3.30 without assuming that Ψ has the instance extension property.

3.7 Search Problem Separations via Proof Complexity

First we show a number of proof complexity separations which, together with Theorems 3.23 and 3.30, imply separations of type-2 search problems. As we stated at the end of Section 3.4, the proof complexity of $Trans(\mathbf{PIGEON})$, $Trans(\mathbf{WeakPIGEON})$, $Trans(\mathbf{LONELY})$, and $Trans(\mathbf{WeakPIGEON})$ are known from the proof complexity research.

Lemma 3.31. *The following separations hold:*
(a) $Trans(\mathbf{PIGEON}) \not\leq_{BDPK} Trans(\mathbf{WeakPIGEON})$.
(b) $Trans(\mathbf{LONELY}) \not\leq_{BDPK} Trans(\mathbf{PIGEON})$.
(c) $Trans(\mathbf{PIGEON}) \not\leq_{BDPK} Trans(\mathbf{ITERATION})$.
(d) $Trans(\mathbf{LONELY}) \not\leq_{BDPK} Trans(\mathbf{ITERATION})$.

Proof. Maciel et. al [MPW00] show that $Trans(\mathbf{WeakPIGEON})$ has quasipolynomial-size bounded-depth PK-proofs, and Lemma 3.32 below shows that $Trans(\mathbf{ITERATION})$ has polynomial-size bounded-depth PK-proofs. It is shown in [PBI93, KPW95] that $Trans(\mathbf{PIGEON})$ requires exponential-size proofs in bounded-depth PK, and therefore (a) and (c) holds. Beame and Pitassi [BP96] prove (b), and their result also implies (d). □

Lemma 3.32. $Trans(\mathbf{ITERATION})$ *has depth-2* PK*-proofs of size polynomial in* N.

Proof. Fix arbitrary $n \geq 1$ and let $N = 2^n$. We show a resolution refutation of the negation of $Trans(\mathbf{ITERATION}, n)$, which is a CNF formula consisting of the following clauses:
(i) $\neg X_{0,0}$
(ii) $\neg X_{i,j}$ for all i, j such that $j < i$
(iii) $\neg X_{i,j} \vee \neg X_{j,j}$ for all i, j such that $i < j$
(iv) $\bigvee_{0 \leq j \leq N-1} X_{i,j}$ for every i

(v) $\neg X_{i,j} \vee \neg X_{i,k}$ for all i, j, k with $j \neq k$

For every $i \geq 1$, define A_i to be the clause $\bigvee_{j \geq i} \neg X_{j,j}$. A_1 is derivable from clauses (i), (iii), and (iv) for $i = 0$. Similarly, for every $i \geq 1$, the clause $X_{i,i} \vee A_{i+1}$ is derived using (ii), (iii), and (iv). Thus, for every $i \geq 2$, A_i is derived by resolving A_{i-1} and $X_{i-1,i-1} \vee A_i$ on $P_{i-1,i-1}$. Finally, the empty clause is derived from $A_n = \neg X_{N,N}$ and $X_{N,N}$, which is derived from (ii) and (iv).

It is easy to convert this resolution refutation into a bounded-depth PK-proof of roughly the same size. □

Lemma 3.33. *The following separations hold:*

(a) $poly(\neg\mathbf{PIGEON}) \not\leq_{HN} poly(\neg\mathbf{OntoPIGEON})$ *over any field* F.

(b) $poly(\neg\mathbf{ITERATION}) \not\leq_{HN} poly(\neg\mathbf{OntoPIGEON})$ *over any field* F.

(c) $poly(\neg\mathbf{PIGEON}) \not\leq_{HN} poly(\neg\mathbf{LONELY})$ *over any field* F *of characteristic* 2.

(d) $poly(\neg\mathbf{ITERATION}) \not\leq_{HN} poly(\neg\mathbf{LONELY})$ *over any field* F *of characteristic* 2.

Proof. [BCE+98, Raz98] prove that $poly(\mathbf{PIGEON})$ requires $\Omega(N)$-degree Nullstellensatz refutations over any field. [CEI96, Bus98c] prove the same for $poly(\mathbf{ITERATION})$ (they call the principle "housesitting").

On the other hand, $poly(\mathbf{OntoPIGEON})$ has constant-degree Nullstellensatz refutations over any field. We have the following polynomials (let X_{ij} say that pigeon i maps to hole j and let Y_{ij} say that hole i maps to pigeon j for $0 \leq i, j < N$):

(i) $(\sum_{j=0}^{N-1} X_{ij}) - 1$ for all i

(ii) $(\sum_{j=0}^{N-1} Y_{ij}) - 1$ for all $i \neq 0$

(iii) X_{i0} for all i

(iv) $X_{ij}(1 - Y_{ji})$ for any i, j

(v) $Y_{ij}(1 - X_{ji})$ for any i, j

(vi) $X_{ij}X_{ij'}$ for any $i, j \neq j'$

Begin by converting each Y_{ij} in (ii) to X_{ji} using (iv) and (v). Now sum up all polynomials in (i) and subtract all polynomials in (ii). What remains is $(\sum_{i=0}^{N} X_{i0}) + 1$. Now we can simply

cancel each X_{i0} using (iii).

Finally, $poly(\mathbf{LONELY})$ has constant-degree Nullstellensatz refutations over characteristic 2. We have the following polynomials (let X_{ij} say that node i maps to node j for $0 \leq i, j < N$):

(i) $X_{ij} - X_{ij}X_{ji}$ for all $i \neq j$

(ii) X_{ii} for all $i \neq 0$

(iii) $1 - X_{00}$

(iv) $\sum_{j=0}^{N-1} X_{ij}) - 1$ for any i

(v) $X_{ij}X_{ij'}$ for any $i, j \neq j'$

Begin by summing up all polynomials in (i), (ii) and (iv): this yields $(\sum_{i=1}^{N-1} X_{0i}) + 1$. If we add $X_{0j}X_{00} + X_{0j}(1 - X_{00})$ to this, we get simply 1. □

Below we state the relative separations of search classes that follow from Theorems 3.23 and 3.30 and Lemmas 3.31 and 3.33. As usual, the oracle separations of the search classes follow from the type-2 separations by Theorem 3.2.

Corollary 3.34. *The following separations hold (G is any generic oracle):*

(a) ([BCE+98]) $\mathbf{PIGEON} \not\leq_m \mathbf{LONELY}$ *and* $\mathbf{PPP}^G \not\subseteq \mathbf{PPA}^G$

(b) ([BCE+98]) $\mathbf{LONELY} \not\leq_m \mathbf{PIGEON}$ *and* $\mathbf{PPA}^G \not\subseteq \mathbf{PPP}^G$

(c) ([BCE+98]) $\mathbf{PIGEON} \not\leq_m \mathbf{OntoPIGEON}$ *and* $\mathbf{PPP}^G \not\subseteq \mathbf{PPAD}^G$

(d) [Mor01] $\mathbf{LONELY} \not\leq_m \mathbf{ITERATION}$ *and* $\mathbf{PPA}^G \not\subseteq \mathbf{PLS}^G$

(e) [Mor01] $\mathbf{PIGEON} \not\leq_m \mathbf{ITERATION}$ *and* $\mathbf{PPP}^G \not\subseteq \mathbf{PLS}^G$

(f)[Mor01] $\mathbf{OntoPIGEON} \not\leq_m \mathbf{ITERATION}$ *and* $\mathbf{PPAD}^G \not\subseteq \mathbf{PLS}^G$

(g) $\mathbf{PIGEON} \not\leq_m \mathbf{WeakPIGEON}$

(h) $\mathbf{ITERATION} \not\leq_m \mathbf{LONELY}$ *and* $\mathbf{PLS}^G \not\subseteq \mathbf{PPA}^G$

Thus, almost all known relative separations of search classes are obtained in Corollary 3.34 via proof complexity separations, and we obtain two previously unknown separations (g) and

(h). Note that we do not know whether **WeakPIGEON** has the instance extension property, and therefore for (h) the generalization of Theorem 3.23 to Theorem 3.24 is vital.

Theories of bounded arithmetic is closely related to computational complexity and proof complexity, and our results connecting these two areas naturally have a consequence on bounded arithmetic as well. For an $\exists$-sentence Φ, we denote by $\Phi^{<a}$ the $\Sigma^b_1(\mathcal{L})$-formula obtained by bounding all existential quantifiers in Φ by a free variable a. Note that a is the only free variable of $\Phi^{<a}$.

Theorem 3.35. *Let Ψ be an $\exists$-sentence over a language $\mathcal{L}$, which is disjoint from the language $\mathcal{L}_A$ of bounded arithmetic. If the relativized bounded arithmetic theory $S_2(\mathcal{L})$ proves $(\forall x)\Psi^{<x}$, then $\mathbf{PIGEON} \not\leq_m Q_\Psi$ and $\mathbf{LONELY} \not\leq_m Q_\Psi$. In fact, $Q_\Phi \not\leq_m Q_\Psi$ for any Φ such that every bounded-depth PK-proof of $Trans(\Phi)$ requires exponential size.*

Proof. The idea is that, if $S_2(\mathcal{L})$ proves $\forall x \Phi^{<x}$, then from the proof we can construct quasi-polynomial-size bounded-depth PK of $Trans(Q_\Phi)$ ([PW85, Kra95]). From Theorem 3.23 it follows that, if $Q_\Psi \leq_m Q_\Phi$, then $Trans(\Psi)$ has subexponential-size bounded-depth PK proofs, which contradicts the assumption. □

3.8 Remarks

We have obtained a number of search problem separations from proof complexity separations and our Theorems 3.23 and 3.30. Note that our proofs of these separations do not depend on the fact that the substitution instance of $Trans(Q_\Psi)$ and $poly(\neg Q_\Psi)$ are uniformly generated by a Turing machine that reduces Q_Φ to Q_Ψ. Hence, all the search problem separations in this paper hold to exclude reductions by nonuniform polynomial-size circuits. The same is true for the separations obtained in [BCE$^+$98, Mor01].

In the proof of Theorem 3.23, we constructed, from a many-one reduction from Q_Φ to Q_Ψ, 'small' depth-3 proofs of $Trans(\Phi)$ using $Trans(\Psi)$ as an axiom scheme. The depth-3 proofs

we construct are tree-like, and in fact they have depth 2.5 under the terminology of [MPW00], since every formula in the proofs we construct has polylogarithmic fan-in at the lowest level.

There has been much work (for example, [LTT89, Aar04]) on the efficiency of local search, whose primary goal is to obtain lower bounds on the number of times a local search heuristics has to be invoked. In our context, this is a pursuit of a lower bound on the number of times f has to be accessed for solving $\mathbf{ITERATION}(f, 1^n)$. Since Theorems 3.28 links the time complexity of $\mathbf{ITERATION}$ with the proof complexity of $Trans(\mathbf{ITERATION})$ (i.e., the housesitting principle) in Nullstellensatz, we can obtain such a lower bound [CEI96, Bus98c] from the degree lower bound for Nullstellensatz. We do not know how such a lower bound compares with the known lower bounds.

The Nash problem is formulated as follows. Given two integer matrices that represent payoffs for a two-player game, find a Nash equilibrium. It is known to be in $\mathbf{PPAD}$ [Pap94b], but not known to be complete for any class. Papadimitriou calls the Nash problem 'a most fundamental computational problem whose complexity is wide open' [Pap01], and there have been attempts to obtain good bounds on this problem; for example, [SvS04]. From our results, a new approach to this problem emerges: namely, formulating the totality of the Nash problem as a family of tautologies, and show a lower bound on the size of their depth-2 PK-proofs. By our Theorem 3.21, such a proof complexity lower bound translates into a lower bound on the deterministic time complexity of Nash. Note that, since Nash is in $\mathbf{PPAD}$ and since $\mathbf{OntoPIGEON}$ is easy for Nullstellensatz, the totality of Nash has low-degree proofs in Nullstellensatz.

All the separations we obtained in this paper are with respect to many-one reducibility. Since all the known separations from [BCE$^+$98, Mor01] are known to hold with respect to Turing reducibility, it is an interesting open problem to see if this stronger separation is obtainable directly from proof complexity separation.

As we remarked above, we do not know whether Theorem 3.30 holds without the assumption that Ψ has the instance extension property. Note that, without this assumption, basically

we need to show that the negation of (3.18) has small-degree Nullstellensatz refutations; However, we do not know how to express the negation of (3.18) as a set of polynomials of a small degree. It seems that Nullstellensatz is not sufficiently expressive for this case to go through.

We made progress toward resolving the relative complexity of $\mathbf{PLS}$ by showing $\mathbf{ITERATION} \not\leq_m \mathbf{LONELY}$. We are interested in knowing whether $\mathbf{ITERATION}$ is many-one reducible to $\mathbf{PIGEON}$ or not, which still remains open. One difficulty is that the iteration principle is easy for almost all proof systems except for Nullstellensatz, while the pigeonhole principle is hard for Nullstellensatz.

From Theorem 3.30 and the fact that $poly(\neg\mathbf{LONELY})$ has constant-degree Nullstellensatz refutations, it follows that the totality of every $\mathbf{PPA}$ problem has low-degree Nullstellensatz proofs. This indicates that the fixed point theorems of Brouwer, Nash, and Kakutani, whose corresponding search problems are in $\mathbf{PPA}$, have low-complexity proofs.

Aside from one type-2 separation that Beame et. al obtain via Nullstellensatz degree lower bounds [BCE+98], we do not know of any other work that explicitly link reductions among search problems with proof complexity. However, Buss in [Bus03] obtains upper and lower bounds on the proof complexity of various combinatorial principles by describing transformations of a proof of one principle into that of another principle. In our context, his proof transformations are in fact many-one reductions between the corresponding search problems.

Theorems 3.23 and 3.30 construct propositional refutations from reductions. Does the converse hold? Is it true that if the translation of a search problem has a simple PK or Nullstellensatz refutation, then the search problem is reducible to, say, $\mathbf{ITERATION}$ (which is easy for PK) or $\mathbf{LONELY}$ (which is easy for Nullstellensatz)? The following is one possible approach to obtain an upper bound on the complexity of Q_Φ from the existence of proofs for $Trans(Q_\Phi)$. It is well known that a resolution refutation of an unsatisfiable CNF formula F gives rise to a branching program B whose internal nodes query the variables of F such that B's output is a clause of F that is falsified by the given truth assignment. Thus, a resolution refutation for $\neg Trans(Q_\Phi, n)$ gives rise to a branching program solving Q_Φ on instances of

size n. However, this reasoning does not quite work because, in our context, the complexity of Q_Φ in measured in terms of n while the proof (or refutation) lengths of $\neg Trans(Q_\Phi, n)$ are measured in terms of $N = 2^n$. Thus, for example, a linear-size refutation of $\neg Trans(Q_\Phi, n)$ only gives rise to a branching program for Q_Φ whose size is exponential in n, which is not very useful.

However, we can extend the above line of reasoning by imposing a certain uniformity condition in the following way. Suppose that there is a resolution refutation π of $\neg Trans(Q_\Phi, n)$ and that the size of π is $N^k = 2^{kn}$ for some $k \in \mathbb{N}$. Now suppose that π is uniform in the sense that the following f_π is polynomial-time: given $i \in \mathbb{N}$, output $j, k \in \mathbb{N}$ and a variable X such that clause i of π is derived from clauses j, k by resolving X. It is easy to see that the existence of such π implies the existence of an algorithm for Q_Φ whose running time is polynomial in the length of the longest path in π; the idea is simulate the branching program corresponding to π, which is implicitly described by f_π. Note that clause i itself may not be computed in polynomial-time, since its size may be superpolynomial in n. This uniformity condition is similar, but not equivalent, to the notion of *implicit proofs* by Krajíček [Kra04]. It is an interesting open problem to explore the connections between algorithms for Q_Φ and proofs for $Trans(Q_\Phi)$ that have implicit descriptions.

Chapter 4

The Limitations of Local Search Heuristics

Local search is a widely used approach to various optimization problems, and $C(\mathbf{ITERATION})$ is essentially the class of search problems for which an efficient (i.e., polynomial-time) local search heuristic exists. Thus, if $Q \not\leq_m \mathbf{ITERATION}$, then $Q \notin C(\mathbf{PLS})$ and therefore there is no efficient local search heuristic for Q, i.e., Q cannot be formulated as a local search problem. See [JPY88, Yan97] for more information on local search and $\mathbf{PLS}$.

In this chapter we present a sufficient condition for a type-2 search problem Q to be nonreducible to $\mathbf{ITERATION}$, and such Q cannot be formulated as a local search problem. Moreover, it follows that $C(Q)^G \not\leq_m C(\mathbf{PLS})^G$ for generic oracles G and also that local search is unlikely to be useful for solving the complete problems of $C(Q)$. Thus, our result in this Chapter shows the limitations of local search heuristics, which have been quite successful in approximating various hard optimization problems [JPY88, Yan97]. This is motivated by the helpful comments on [Mor01] by Søren Riis.

It will be easy to see that all of $\mathbf{PIGEON}$, $\mathbf{LONELY}$, $\mathbf{OntoPIGEON}$, and $\mathbf{WeakPIGEON}$ $\mathbf{ITERATION}$ satisfy the sufficient condition and thus we obtain alternative proofs of Theorem 3.8 and items (d) and (e) of Corollary 3.34. This 'separation criterion' is a generalization

of the main result of our M.Sc thesis [Mor01], and it is also a stronger variant of Riis's independence criterion for the relativized theory $S_2^2(L)$ of bounded arithmetic [Rii93].

In [Mor01], we proved that $\mathbf{SOS} \not\leq_m \mathbf{ITERATION}$. Our proof depended on only one property of the $\mathbf{SOS}$ problem, and we realized that, as long as a type-2 search problem satisfies this property, we can easily show that $Q \not\leq_m \mathbf{ITERATION}$. The property of $\mathbf{SOS}$ is the following. Consider a two-player game between Solver and Spoiler. Solver is a type-2 Turing machine that, given $(succ, pred, 1^n)$, tries to find a solution for $\mathbf{SOS}$ by querying $succ$ and $pred$; Solver thus explores the underlying dag G, looking for either a source other than 0 or a sink. Spoiler's job is to answer all the queries by Solver while avoiding creating a solution for $\mathbf{SOS}$. It is easy to see that, if Solver is a polynomial-time machine, then Spoiler succeeds, i.e., $\mathbf{SOS}$ is not in type-2 $\mathbf{FP}$. But a stronger property holds: Spoiler still wins even if we give an advantage to Solver by fixing the values of $succ$ and $pred$ on polynomially many nodes, although we are not allowed to create any solution. This property, namely the existence of a winning strategy for Spoiler even if Solver is helped by fixing the values of the type-1 arguments of Q for polynomially many elements of V_n without creating a solution, is essentially what is needed to show that $Q \not\leq_m \mathbf{ITERATION}$.

It is not hard to see that all the type-2 problem we discussed in Chapter 3 except $\mathbf{ITERATION}$ has the above property, and hence none of them is many-one reducible to $\mathbf{ITERATION}$. Note that $\mathbf{ITERATION}$ itself does not have the property. Suppose that we help Solver by fixing, for each n, $f(2^n - 2) = 2^n - 1$. (Recall that $2^n - 1$ is the largest element in V_n.) Then Solver can simply query $f(2^n - 1)$: if Spoiler answers it with $2^n - 1$ then $2^n - 2$ is a solution, otherwise $2^n - 1$ itself becomes a solution.

The following is our strategy for proving the separation criterion for $\mathbf{ITERATION}$. We first show a sufficient condition C for a type-2 search problem Q to have the desirable property, which we will call the *safety property*. Then we prove that, if Φ has the safety property, then $Q \not\leq_m \mathbf{ITERATION}$. It then follows that the condition C is a sufficient condition for $Q \not\leq_m \mathbf{ITERATION}$.

4.1 A Separation Criterion for PLS

Theorem 4.1. *Let $\mathcal{L}$ be a first-order language such that $\mathcal{L}$ does not contain any built-in symbol other than $=$ and 0. Let Φ be an $\exists$-sentence over $\mathcal{L}$. If Φ fails in an infinite structure, then*

$$Q_\Phi \not\leq_m \textbf{ITERATION}.$$

Proof. For simplicity, we fix the language $\mathcal{L}$ to be $\mathcal{L} = \{0, \alpha\}$, where α is a unary function, and assume that Φ is an $\exists$-sentence over L of the form $(\exists\vec{x})\phi(\vec{x})$, and it is in basic form. The case for languages with arbitrarily many input symbols is analogous to the current case. Let $\mathcal{K} = (K, \alpha_K)$ be an infinite structure in which Φ fails.

For each $n \geq 1$, we call a partial function $\rho_n : V_n \to V_n$ a *restriction*. Let $\rho = \{\rho_n\}_n$ be a family of restrictions. The size of restriction ρ_n is $|dom(\rho_n)|$ and is written $|\rho_n|$. We say that $\{\rho_n\}_n$ is a *polysize family* if $|\rho_n| \in n^{O(1)}$. We say that ρ_n is *safe for* Φ if there exists a partial one-one mapping $h : V_n \to K$ such that (i) $dom(h) = dom(\rho_n) \cup im(\rho_n)$; and (ii) $\rho_n(v) = u$ implies $\alpha_K(h(v)) = h(u)$. Note that, if ρ_n is safe for Φ and $|\rho_n| << N$, then ρ_n *does not specify any solution for* Q_Φ in the sense of Definition 3.14 on page 3.14.

We claim that, if ρ_n is a safe restriction and $m + |\rho_n| << N$, then we can answer m queries to α consistently with ρ_n so that ρ_n augmented with the answers is still safe. We call this the *safety property* of Φ, and state it more formally as follows: If $\rho = \{\rho_n\}_n$ is a polysize family of safe restrictions, and if $\{T_n\}_n$ is a family of search trees of height polylogarithmic in N, then, for all sufficiently large n, T_n contains a path P such that ρ_n and π_P are consistent and $\rho_n \cup \pi_P$ is safe for Φ.

The above claim is proven as follows. Let n be sufficiently large so that $|\rho_n|$ plus the depth of T_n is less than, say, $\frac{|V_n|}{2}$. Let $h : V_n \to K$ be a mapping that witnesses the safety of ρ_n. We traverse T_n from the root to a leaf by answering the queries to α consistently with ρ_n so that, in the end, $\rho_n \cup \pi_P$ is safe for Φ, where P is the path that we have traversed. Let $\alpha(u)$ be the query that the current node asks. Two cases arise:

Case 1: $u \in dom(h)$. Let u_K and v_K denote $h(u) \in K$ and $\alpha_K(v_K)$, respectively. If $v_K \in$

$dom(h)$, then there is $v \in V_n$ with $h(v) = v_K$; we set $\alpha(u) = v$. If $v_K \notin dom(h)$, then choose an arbitrary $v' \in V_n$ and set $\alpha(u) = v'$ and $h(v') = v_K$.

Case 2: $u \notin dom(h)$. Then choose an arbitrary $v' \in V_n$ and set $\alpha(u) = v'$ and $h(v') = al_K(h(u))$.

In the above proof, choosing an arbitrary v' works because of the assumption that $\mathcal{L}$ does not have any built-in symbol other than 0 and $=$. If $\mathcal{L}$ has a built-in symbol, say $\leq$, then we must choose v' so that setting $\alpha(u) = v'$ is consistent with $\leq$ with respect to the elements in $dom(h)$. However, if there is no built-in symbol, then we are free to choose any $v' \in V_n$ with no fear of being inconsistent.

The safety property has been implicit in separation proofs in [Bus86, Kra95, BCE+98, CK98, Mor01].

Now assume for the sake of contradiction that $Q_\Phi \leq_m \mathbf{ITERATION}$. Since $\mathbf{ITERATION}$ has the instance extension property (Lemma 3.10), there is a reduction M from Q_Φ to $\mathbf{ITERATION}$ that does not ask any query before a query $(\beta, 1^m)$ to $\mathbf{ITERATION}$ (Lemma 3.11)..

Claim 4.2. *There exists a polysize family $\{\rho_n\}_n$ of restrictions such that, for sufficiently large n, the following hold: (1) ρ_n is safe for Φ; and (2) ρ_n contains the answers to all the queries to α and $\mathbf{ITERATION}$ made by M on $(\alpha, 1^n)$.*

Suppose Claim 4.2 holds and consider M on $(\alpha, 1^n)$ for n sufficiently large. We answer all the queries to α and $\mathbf{ITERATION}$ according to ρ_n asserted to exist by the Claim. At the end of its computation, M needs to output some v as a solution for Q_Φ on this instance, although no solution for Q_Φ has been specified. Hence, after M outputs some $\vec{v}$, we can construct a total extension α^* of ρ_n so that $\phi(\vec{v})$ is false in structure (V_n, α^*). This completes the proof of Theorem 4.1 from Claim 4.2.

It remains to prove Claim 4.2. Fix n sufficiently large and let $q = (\beta, 1^m)$ be the query that M asks to $\mathbf{ITERATION}$. We want to construct a safe restriction μ_1 so that a solution for $\mathbf{ITERATION}(\beta, 1^m)$ is specified. Recall that, for each $x \in V_m$, $T_m^{\beta(x)}$ is the search tree

corresponding to the computations of $\beta(x)$; we denote it as $B(x)$. We say a path P of $B(x)$ is *safe* if the corresponding restriction π_P is safe. For each $x \in V_m$, let $Safe_B(x)$ be the set of all safe paths of $B(x)$. Because of the safety property of Φ, $Safe_B(x)$ is nonempty for every $x \in V_m$. There are three cases to consider:

Case (i): $Safe_B(0)$ contains a path P with leaf label $\beta(0) = 0$. This path defines a solution for $\mathbf{ITERATION}$. We give the solution to M and set $\mu_1 := \pi_P$.

Case (ii): For some $x \in V_m$, $Safe_B(x)$ contains a path P with leaf label $\beta(x) = y$ for some $y < x$. This path also defines a solution for $\mathbf{ITERATION}$, so we give the solution to M and set $\mu_1 := \pi_P$.

Case (iii): The above two cases do not hold. Since the first case does not hold, every path in $Safe_B(0)$ corresponds to a computation of M_β with $\beta(0) > 0$. Similarly, since the second case does not hold, every path in $Safe_B(1^m)$ leads to $\beta(1^m) = 1^m$. Hence, by the least number principle, there exists $x \in V_m$ such that: (a) $Safe_B(x)$ contains a path P that leads to $\beta(x) = y$ for some $y > x$; and (b) for all $z > x$, every path in $Safe_B(z)$ leads to $\beta(z) = z$. Let $Good_B(y)$ as the set of paths P' of $B(y)$ such that $\pi_{P'}$ is consistent with π_P and $\pi_P \cup \pi_{P'}$ is safe for Φ. By the safety property of Φ, $Good_B(y)$ is not empty. Let P^* be any path in $Good_B(y)$. Set μ_1 to be $\pi_{P'} \cup \pi_{P^*}$ and return x to M as a solution for its $\mathbf{ITERATION}$-query. Note that x is a solution because $\beta(x) = y$ and $\beta(y) = y$. This concludes the construction of μ_1.

Let w be the answer to the query to $\mathbf{ITERATION}$ that is constructed as above. Recall that $T_n^{M_2}(w)$ is the search tree encoding all possible computations of M_2 in this case. By the safety property of Φ, there exists a path R in $T_n^{M_2}(w)$ such that π_R and μ_1 are consistent and $\pi_R \cup \mu_1$ is safe. Setting $\rho_n := \pi_R \cup \mu_1$ makes Claim 4.2 hold. □

Note that the conclusion of Theorem 4.1 implies that $C(Q_\Phi)^G \not\subseteq \mathbf{PLS}^G$ for any generic oracle G by Theorem 3.2.

The reader may be familiar with the following result of Krajíček (Theorem 11.3.1 of [Kra95]):

Theorem 4.3. *[Kra95] Let $\mathcal{L}$ be a first-order language with no function (built-in or not) and*

no built-in predicate. Let Φ be a $\exists\forall$-sentence over $\mathcal{L}$. If Φ fails in an infinite structure, then the type-2 problem Q_Φ is not in type-2 $\mathbf{FP^{NP}}$.

In this dissertation, we only work with type-2 search problems whose totality is represented by sentences using one or more function symbols, and therefore Theorem 4.3 does not apply to them. However, it is easy to formulate their totality by sentences over relational languages. The idea is to introduce the graph R_f for each function f. For example, the injective, functional pigeonhole principle can be stated as follows:

$$(\forall x)(\exists y)R_f(x,y) \wedge (\forall x)\neg R_f(x,0) \supset (\exists x)(\exists y)(\exists z)[x \neq y \wedge R_f(x,z) \wedge R_f(y,z)]$$

Note that the above sentence is not a $\exists$-sentence but a $\exists\forall$-sentence. Let $\mathbf{PIGEON}^R$ be the corresponding type-2 search problem. The superscript R stands for 'Relational language'.

Theorem 4.3 applies to $\mathbf{PIGEON}^R$ and we conclude that $\mathbf{PIGEON}^R$ is not in type-2 $\mathbf{FP^{NP}}$. However, Theorem 4.3 still does not say anything about the complexity of $\mathbf{PIGEON}$. The problem is that $\mathbf{PIGEON}$ and $\mathbf{PIGEON}^R$ are not equivalent, and in fact $\mathbf{PIGEON}^R$ is strictly harder than $\mathbf{PIGEON}$, since $\mathbf{PIGEON}$ is in type-2 $\mathbf{FP^{NP}}$ by Lemma 3.4. (Note that $\mathbf{PIGEON} \leq_m \mathbf{PIGEON}^R$ is obvious.) On the other hand, from our Theorem 4.1 we can deduce that $\mathbf{PIGEON}$ is not reducible to $\mathbf{ITERATION}$, a fact which does not follow from Theorem 4.3. Thus, Theorems 4.1 and 4.3 are incomparable.

4.2 Connections to Bounded Arithmetic

Riis proves in [Rii93] the following *independence criterion* for the relativized theory $S^2_2(\mathcal{L})$. Riis's proof is model-theoretic, and Krajíček shows a complexity-theoretic proof of Riis's result in [Kra95] using Theorem 4.3. Note that this independence criterion does not require $\mathcal{L}$ to be relational.

Although our Theorem 4.1 is incomparable to Theorem 4.3, Riis's result also follows from it.

Theorem 4.4. *[Rii93] Let $\mathcal{L}$ be a language that is disjoint with the language of bounded arithmetic, and let $\Phi = (\exists \vec{x})\phi(\vec{x})$ be a sentence over $\mathcal{L}$ of arbitrary quantifier-complexity. If Φ fails in an infinite structure, then the relativized bounded arithmetic theory $S^2_2(\mathcal{L})$ does not prove $\Phi^{<a}$.*

Proof. Krajíček has a proof of this theorem based on Theorem 4.3 in [Kra95]. Since our proof is almost identical to his, we only sketch the idea. Let Φ be of the form

$$\exists x_1 \forall y_1 \ldots \exists x_k \forall y_k \phi(x_1, y_1, \ldots, x_k, y_k)$$

with ϕ quantifier-free. Define a herbrandization Φ_H of Φ as

$$\exists x_1 \exists x_2 \ldots \exists x_k \phi(x_1, f_1(a, x_1), \ldots, x_k, f_k(a, x_1, \ldots, x_k)),$$

where $f_1, \ldots, f_k$ are new functions. Let $\mathcal{K}$ be an infinite structure in which Φ is false. By defining $f_1, \ldots, f_k$ appropriately, $\mathcal{K}$ can be extended to $\mathcal{K}'$ in which Φ_H fails; thus, Q_{Φ_H} is not reducible to **ITERATION** by Theorem 4.1.

Let $\mathcal{L}' = \mathcal{L} \cup \{f_1, \ldots, f_k\}$. Since **ITERATION** characterizes the $\Sigma^b_1(\mathcal{L}')$-consequences of $S^2_2(\mathcal{L}')$ ([CK98]), $S^2_2(\mathcal{L}')$ does not prove

$$\exists x_1 < a \exists x_2 < a \ldots \exists x_k < a$$

$$[f_1(a, x_1) < a \wedge \ldots \wedge f_k(a, x_1, \ldots, x_k) < a \supset$$

$$\phi(x_1, f_1(a, x_1), \ldots, x_k, f_k(a, x_1, \ldots, x_k))].$$

Let M be a model of $S^2_2(\mathcal{L}')$ in which the above formula fails. Then $\Phi^{<a}$ fails in this model. □

From Theorem 4.4 it follows that any type-2 search problem Q satisfying the assumptions of Theorem 4.1 is not $\hat{\Sigma}^b_1$-definable in the relativized $S^2_2(\mathcal{L})$. Then Q is not $\hat{\Sigma}^b_1$-definable in $T^1_2(\mathcal{L})$ because $S^2_2(\mathcal{L})$ is Σ^b_2-conservative over $T^1_2(\mathcal{L})$. The intuitive meaning of this is that $T^1_2(\mathcal{L})$ is unable to show that Q is reducible to **ITERATION**. However, $Q \not\leq_m$ **ITERATION** does not follow since there may be a many-one reduction which is not formalizable in $T^1_2(\mathcal{L})$. Thus, Theorem 4.1 is a stronger version of Theorem 4.4.

4.3 Remarks

In [Rii01] Riis proves that the assumption of our Theorem 4.1 implies that $\neg Trans(Q_\Phi, n)$ requires exponential-size refutations in tree-like resolution. Krajíček in [Kra01] proves a similar, stronger result for the refutation system $R^*(\log)$ instead of tree-like resolution, where $R^*(\log)$ is essentially tree-like PK with a restriction that every cut formula be a conjunction of k literals, where k is logarithmic in the size of $\neg Trans(Q_\Phi, n)$. In the terminology of [MPW00], $R^*(\log)$ is essentially tree-like PK with cuts only on depth-0.5 formulas.

It would be nice to prove Theorem 4.1 from proof complexity lower bounds, and Krajíček's result is an obvious candidate. However, Krajíček's result is not strong enough to derive Theorem 4.1. Below we explain why.

Let Φ be an $\exists$-sentence satisfying the assumptions of Theorem 4.1. Then Krajíček's result implies that $\neg Trans(Q_\Phi, n)$ requires exponential-size PK-refutations with cut restricted to depth-0.5 formulas. Theorem 4.1 follows if we succeed in constructing subexponential-size refutations of $\neg Trans(Q_\Phi, n)$ in $R^*(\log)$ by assuming that $Q_\Phi \leq \mathbf{ITERATION}$. However, there are two difficulties. By modifying the proof of Theorem 3.23, it is not hard to construct subexponential-size PK-derivation of a substitution instance $\neg Trans(\mathbf{ITERATION}, m)[Y/T]$ from $\neg Trans(Q_\Phi, n)$, where m and the substitution $[Y/T]$ depends on the reduction of Q_Φ to $\mathbf{ITERATION}$. The first difficulty is that this derivation already requires cuts on depth-1.5 formulas (i.e., DNFs with polylogarithmic size terms), and we do not know how the depth of cut formulas can be reduced to 0.5.

The second problem is that, even if we could derive $\neg Trans(\mathbf{ITERATION}, m)[Y/T]$ from $\neg Trans(Q_\Phi, n)$ in PK with cuts only on depth-0.5 formulas, we do not know how to construct subexponential-size refutation of $\neg Trans(\mathbf{ITERATION}, m)[Y/T]$ with cuts only on depth-0.5 formulas. If we apply the substitution $[Y/T]$ to the polynomial-size resolution refutation of $\neg Trans(\mathbf{ITERATION}, m)$ (see the proof of Lemma 3.32), each resolution step on variable Y becomes a cut on the DNF formula $DNF(Y)$, which has depth 1.5.

Note that the refutation version of Theorem 3.23 can be found in our joint paper [BOM04].

Part II

Quantifi ed Propositional Calculus and Bounded Arithmetic

Chapter 5

Quantified Propositional Calculus

5.1 Quantified Propositional Calculus: Basic Definitions

Quantified Propositional Calculus (QPC) is obtained by introducing quantifiers into propositional calculus, where $(\exists x)A(x)$ is equivalent to $A(\mathsf{T}) \vee A(\mathsf{F})$ and $(\forall x)A(x)$ is equivalent to $A(\mathsf{T}) \wedge A(\mathsf{F})$. An advantage of QPC is that it has enough expressive power to represent everything in $\mathbf{PSPACE}$ in a concise way without losing the syntactic and semantic simplicity of propositional calculus. The following is an example QPC formula, which asserts that a QPC formula $A(\vec{p})$ has a truth value x:

$$(\exists x)[(x \wedge A(\vec{p})) \vee (\neg x \wedge \neg A(\vec{p})] \tag{5.1}$$

QPC formulas are also known as QBF (Quantified Boolean Formulas), and efficiency of decision procedures for the satisfiability of QBF formulas is an important issue in various research areas such as formal verification, planning, reasoning about knowledge, and there has been much effort in designing and implementing QBF solvers that can be used in practice [BST03]. The study of QPC proof complexity is relevant to such effort.

Let $\{p_i : i \in \mathbb{N}\}$ and $\{x_i : i \in \mathbb{N}\}$ be the sets of p-variables and x-variables, respectively. We use the p-variables to denote free variables and the x-variables to denote bound variables. The following is a definition of the syntax of QPC below, which extends Definition 2.8 of

propositional calculus.

Definition 5.1. Formulas *are defined inductively as follows. (1) The atomic formulas are* (T), (F), *and* (p_i) *and* (x_i) *for every* $i \in \mathbb{N}$. *(2) If* ϕ *and* ψ *are formulas, then so are* $(\phi \wedge \psi)$, $(\phi \vee \psi)$, *and* $(\neg\phi)$. *(3) If* ϕ *is a formula, then for every* $i \in \mathbb{N}$, *both* $(\exists x_i \phi)$ *and* $(\forall x_i \phi)$ *are formulas.*

Often we do not write all the parentheses. Note that we parenthesize the atomic formulas since it somewhat simplifies parsing operations for QPC in Section 5.4.1. The semantics of QPC formulas is defined in an obvious way.

Definition 5.2. *A formula* A *is said to be* proper *and called a* QPC formula *iff every occurrence of an* x*-variable in* A *is bound.*

It is clear from the above that propositional formulas of Definition 2.8 are quantifier-free QPC formulas.

Both Σ_0^q and Π_0^q denote the set of propositional formulas. For $i \geq 1$, Σ_i^q is the set of QPC formulas that has a prenex form with at most $i-1$ quantifier alternations beginning with $\exists$, and Π_i^q is the dual of Σ_i^q. Note that $\Sigma_{i-1}^q \subseteq \Pi_i^q$ and $\Pi_{i-1}^q \subseteq \Sigma_i^q$ for all $i \geq 1$. For example, the formula (5.1) is in Σ_1^q iff $A(\vec{p})$ is quantifier-free; more generally, for every $i \geq 0$, it is in Σ_{i+1}^q if $A(\vec{p}) \in \Sigma_i^q \cup \Pi_i^q$. It is a well-known fact [Wra77, Joh90, Pap94a] that, for every $i \geq 1$, the evaluation problems for Σ_i^q-sentences and Π_i^q-sentences are complete for $\mathbf{\Sigma}_i^p$ and $\mathbf{\Pi}_i^p$, respectively.

The notions of QPC proof systems is obtained in a natural way the definition of propositional proof systems.

Definition 5.3. *Let* Q_1 *and* Q_2 *be QPC proof systems, and let* V *be a set of QPC formulas. We say that* Q_2 p-simulates Q_1 with respect to (w.r.t.) V *iff there is a polynomial-time function* f *such that, whenever* π *is a* Q_1*-proof of* $A \in V$, $f(\pi)$ *is a* Q_2*-proof of* A.

The following is an easy generalization of a fundamental theorem of propositional proof complexity by Cook and Reckhow [CR77], connecting the question of proof lengths to open problems of complexity theory:

Theorem 5.4. *(i) There exists a proof system Q in which every valid QPC formula A has a proof of size polynomial in $|A|$ iff $\mathbf{NP} = \mathbf{PSPACE}$. (ii) For every $i \geq 0$, there exists a proof system Q in which every valid Σ_i^q-formula A has a proof of size polynomial in $|A|$ iff $\mathbf{NP} = \mathbf{\Pi_{i+1}^p}$.*

Note that $\mathbf{NP} = \mathbf{\Pi_{i+1}^p}$ for some $i \geq 0$ if and only if $\mathbf{NP} = \mathbf{PH}$.

In [KP90], Krajíček and Pudlák introduced Gentzen-style sequent calculus systems for QPC which we call KPG, KPG_i, and KPG_i^*, for $i \geq 0$. Note that these systems are known without the prefix KP (for Krajíček-Pudlák); we added the prefix, since we are modifying these systems below and call them G, G_i, and G_i^*.

Definition 5.5. *([KP90]) KPG is obtained by augmenting PK with the following new inference rules:*

$$\exists\text{-left} : \frac{A(b), \Gamma \to \Delta}{\exists x A(x), \Gamma \to \Delta} \quad \exists\text{-right} : \frac{\Gamma \to \Delta, A(B)}{\Gamma \to \Delta, \exists x A(x)}$$

$$\forall\text{-left} : \frac{A(B), \Gamma \to \Delta}{\forall x A(x), \Gamma \to \Delta} \quad \forall\text{-right} : \frac{\Gamma \to \Delta, A(b)}{\Gamma \to \Delta, \forall x A(x)}$$

where b is an eigenvariable not occurring in the bottom sequent and B is any proper formula.

We also allow as an initial sequent any sequent of the form $A \to A$ for any QPC formula A.

We call B the *target* of the corresponding $\exists$-right or $\forall$-left step. For each of the above quantifier rules, the auxiliary formula is the formula that occurs only in the upper sequent (i.e., either $A(B)$ or $A(b)$) and the principal formula is the formula that appears only in the lower sequent (i.e., either $\forall x A(x)$ or $\exists x A(x)$).

Definition 5.6. *([KP90, Kra95]) For $i \geq 0$, KPG_i is obtained by requiring that all formulas in a KPG-proof be $\Sigma_i^q \cup \Pi_i^q$. KPG_i^* is KPG_i restricted to tree-like proofs. Note that KPG_0 and KPG_0^* are the propositional proof systems PK and tree-like PK, respectively.*

For example, Figure 5.1 is a KPG_1^*-proof of the formula (5.1) on page 87, where $A(\vec{p})$ is assumed to be quantifier-free. The proof is not a KPG_0^*-proof, since, although it is tree-like, it

$$\dfrac{\dfrac{\dfrac{\dfrac{\to \mathsf{T} \qquad A \to A}{A \to \mathsf{T} \wedge A}}{A \to (\mathsf{T} \wedge A) \vee (\neg \mathsf{T} \wedge \neg A)}}{A \to (\exists x)[(x \wedge A) \vee (\neg x \wedge \neg A)]}\ \exists\text{-right} \qquad \dfrac{\dfrac{\dfrac{\mathsf{F} \to \qquad A \to A}{\to \neg \mathsf{F} \wedge \neg A, A}}{\to (\mathsf{F} \wedge A) \vee (\neg \mathsf{F} \wedge \neg A), A}}{\to (\exists x)[(x \wedge A) \vee (\neg x \wedge \neg A)], A}\ \exists\text{-right}}{\to (\exists x)[(x \wedge A) \vee (\neg x \wedge \neg A)]}\ \text{Cut on } A(\vec{p})$$

Figure 5.1: A sequent calculus proof of the formula (5.1) on page 87, where $A(\vec{p})$ is quantifier-free. We suppress the free variables $\vec{p}$ in the proof. This proof is a KPG_1^*-proof, and it is also a G_0^*-proof; however, it is not a KPG_0^*-proof.

contains quantified formulas.

The restriction that KPG_i and KPG_i^* can only reason about Σ_i^q-formulas does not seem very natural. Moreover, the known correspondences between T_2^i and KPG_i and between S_2^i and KPG_i^* (Theorem 5.29 below) are not optimal; for example, T_2^i can reason about formulas with more than $i-1$ quantifier alternations while T_2^i-proofs of such formulas cannot, by definition, be translated into KPG_i-proofs.

We remedy the situation by modifying the definition of KPG_i and KPG_i^* to obtain the systems which we call G_i and G_i^*.

Definition 5.7. *G is obtained by augmenting PK with the four quantifier-introduction rules, with the additional restriction that the target of every $\forall$-left and $\exists$-right step be quantifier-free. For $i \geq 0$, G_i is G with cuts restricted to $\Sigma_i^q \cup \Pi_i^q$-formulas. G_i^* is the tree-like version of G_i.*

Note that the proof in Figure 5.1 on page 90 is a G_0^*-proof, since the cut formula $A(\vec{p})$ is quantifier-free. Note that G_0^*-proofs are allowed to contain quantified formulas.

We have changed the system G_i which Krajíček and Pudlák defined in [KP90], and the reader may be concerned that the modification may result in confusion. First, we would like to argue that our modification need not cause any confusion. The reason is that, for every $i \geq 1$, KPG_i and G_i (under our Definition 5.7) are p-equivalent w.r.t. $\Sigma_i^q \cup \Pi_i^p$-formulas (Lemma 5.10 below). Thus, every QPC formula has the same proof complexity (up to an application of

a polynomial) in KPG_i and G_i. And similarly for KPG_i^* and G_i.

Moreover, we argue that we gain a lot by modifying KPG_i into G_i. The biggest advantage is that the modification enables us to ask new questions that could not be formulated under the original formulations. More specifically, for every $i \geq 0$, G_i is now a complete proof system for the whole QPC, as opposed to KPG_i, which is a proof system for only $\Sigma_i^q \cup \Pi_i^q$-formulas. This allows us to ask questions on the power of the systems G_i and G_i^* with respect to formulas that have more than $i-1$ quantifier alternations. For example, under our definition, we can discuss the proof lengths of QPC formulas in G_0 and G_0^*, while we cannot ask such a question for KPG_0^* and KPG_0, since quantified formulas are not allowed to appear in the proofs in the latter systems. In Theorem 5.9, we show that every valid QPC sentence has an exponential-size G-proof while the best upper bound for G_0^* is doubly exponential, and now we can ask the question: can we separate G and G_0^* by quantified formulas? Furthermore, the fact that G_i and G_i^* are complete proof systems for QPC allows us to extend the known connections between bounded arithmetic theories and QPC sequent calculus (Theorems 5.29 and 5.30), and it also gives rise to the question on the complexity of the QPC witnessing problems for various formulas and systems (see Chapter 6). In particular, many of our results on the complexity of the QPC witnessing problems cannot be stated under the original definitions of KPG_i and KPG_i^*.

Our modification not only has many advantages but also it is natural. The definition of G_i and G_i^* by restricting the complexity of cut formulas is in the spirit of traditional proof theory, and it is motivated by the way Pitassi defines the bounded-depth propositional PK system by restricting the depth of cut formulas [Pit02]. The restriction that the target of $\forall$-left and $\exists$-right rules of G be quantifier-free ensures that all quantifier-introduction rules increase the quantifier complexity of the auxiliary formula, as opposed to those rules of KPG, which can result in a decrease of the quantifier complexity.

5.2 The Reasoning Power of the QPC Calculus Systems

We have defined the systems G_i and G_i^* syntactically; here we would like to present some intuition on what kind of reasoning can be carried out and what kind of QPC tautologies have or are expected to have short (i.e., polynomial in the size of the endsequent) proofs in each of these systems. These systems are defined by changing two parameters: the power of cut, and whether a proof is required to be tree-like. We describe intuitive meaning of these parameters.

The power of cut formulas is related to the power of lemmas that we are allowed to use. When we prove a theorem T using a lemma L, we first prove that L is true and then we prove that L implies T. In sequent calculus, this is represented as a proof of $\rightarrow T$ from the two sequents $\rightarrow L$ and $L \rightarrow T$:

$$\dfrac{L \rightarrow T \qquad \dfrac{\rightarrow L}{\rightarrow T, L}\,\text{Weakening}}{\rightarrow T}\,\text{Cut on } L$$

Let $i \geq 0$ and let H_i be either G_i or G_i^*. The above description means that H_i has a short proof of theorem T if the two sequents $\rightarrow L$ and $L \rightarrow T$ has short proofs. Although our characterization of the reasoning power of H_i is still incomplete, we can summarize it as follows:

- H_i is a QPC proof system simulating human reasoning that uses as lemmas Σ_i^q-formulas that have short proofs in H_i.

For every circuit C, there is a propositional formula $\phi_C(\vec{p}, \vec{w})$ such that (i) the size of ϕ_C is polynomial in the size of C; (ii) $\vec{p}$ are variables corresponding to the input gates of C; (iii) $\vec{w}$ are variables corresponding to the internal gates of C; and (iv) ϕ_C is a Horn formula. Such a formula is constructed by a standard method showing that the satisfiability for propositional Horn formulas is complete for $\mathbf{P}$. In [KP90, Kra95], it is proven that G_1^* has polynomial-size proofs of

$$(\exists \vec{w})\phi_C(\vec{p}, \vec{w}). \tag{5.2}$$

Note that the above Σ_1^q-formula asserts the existence of the internal values of a circuit, whose discovery is a complete problem for $\mathbf{FP}$. Thus, G_1^* has short proofs of the existence of any

object that can be constructed in polynomial-time. This fact gives a more concrete statement on the reasoning power of G_1^*:

- G_1^* is a QPC proof system that can reason about polynomial-time constructible objects.

Recall that the difference between tree-like proofs and dag proofs is that, in a dag proof, a derived sequent can be used as an upper sequent as many times as need, while in a tree-like proof it has to be derived from scratch every time it is used. This is still a syntactic description and it does not tell us much what kind of natural reasoning can be carried out in a dag proof but not in a tree-like proof. The following is a more intuitive description of the different power of tree-like and dag proofs.

We first formulate this example in first-order arithmetic. Let $A(x)$ be a first-order formula representing a relation on $\mathbb{N}$. Assume that we have the following sequents: $\rightarrow A(0)$, and $(\forall x)[A(i) \rightarrow A(i+1)]$. Suppose that we want to derive $\rightarrow A(k)$ for some $k \in \mathbb{N}$. A natural thing to do in a tree-like proof is to proceed in the following way. First, instantiate $(\forall x)[A(i) \rightarrow A(i+1)]$ so that we have $A(i) \rightarrow A(i+1)$ for each $i \in [0, k-1]$. Next, for each i that is a multiple of 2, derive $A(i) \rightarrow A(i+2)$ by

$$\frac{A(i) \rightarrow A(i+1) \qquad A(i+1) \rightarrow A(i+2)}{A(i) \rightarrow A(i+2)} \text{ Cut on } A(i+1)$$

Subsequently, for each i that is a multiple of 4, derive $A(i) \rightarrow A(i+4)$ from

$$A(i) \rightarrow A(i+2) \text{ and } A(i+2) \rightarrow A(i+4)$$

by a cut on $A(i+2)$. Proceed in this way until we derive $A(0) \rightarrow A(k)$. Then $\rightarrow A(k)$ is derived from this sequent and $\rightarrow A(0)$ by a cut on $A(0)$.

There are two properties of this tree-like proof that we would like to point out: first, all the cut formulas are of the form $A(i)$ for some $i \in \mathbb{N}$, and this proof is a tree of height $O(\log k)$ and contains $O(k)$ sequents. What this means is that, even if we have $A(i) \rightarrow A(i+1)$ for every $i \in \mathbb{N}$, a tree-like sequent calculus system has a short proof of $\rightarrow A(k)$ of the above form for small k only, and it is not clear if there is a proof shorter than this.

Next, we show how to construct a dag proof of $A(k)$. The construction proceeds as follows. Suppose that we have $(\forall x)[A(x) \rightarrow A(x + 2^i)]$ for some i. Initially, we have it for $i = 0$. Instantiate it so that we have two sequents

$$A(b) \rightarrow A(b + 2^i) \quad \text{and} \quad A(b + 2^i) \rightarrow A(b + 2^{i+1})$$

where b is a free variable. Derive $A(b) \rightarrow A(b + 2^{i+1})$, from which we conclude $(\forall x)[A(x) \rightarrow A(x + 2^{i+1})]$. Note that this is a dag proof and not tree-like since the sequent $(\forall x)[A(x) \rightarrow A(x + 2^i)]$ has to be instantiated twice, i.e., it has to be used at least twice. In this dag proof, all cut formulas are of the form $A(b + 2^i)$ If the endsequent is $A(k)$, then the proof has only $O(\log k)$ sequents.

The above example shows the following: in proofs of size $O(n)$, a tree-like system can derive

$$A(0) \wedge (\forall x)[A(x) \rightarrow A(x + 1)] \rightarrow A(n)$$

while a dag system can derive

$$A(0) \wedge (\forall x)[A(x) \rightarrow A(x + 1)] \rightarrow A(2^n).$$

Thus, the difference between tree-like dag proofs amounts to the difference between A-IND and A-LIND (see Definition 2.18, page 31) in bounded arithmetic, interpreting n as the binary length of k such that we want to prove $A(k)$. Let $i \geq 0$. Since G_i and G_i^* can cut Σ_i^q-formulas, it follows that, informally speaking,

- G_i^* simulates Σ_i^b-LIND in short proofs, and
- G_i simulates Σ_i^b-IND in short proofs.

Combining all of the above, the power of G_i and G_i^*, for $i \geq 0$, is summarized as the following, informal statements:

- G_i^* is a QPC proof system that simulates reasoning that uses length-induction (LIND) on Σ_i^q-formulas, and

- G_i is a QPC proof system that simulates reasoning that uses induction (IND) on Σ_i^q-formulas.

The above statements means that G_i^* and G_i simulate S_2^i and T_2^i, respectively, in some precise sense, and this is formally proven in [KP90] for G_i and in [Kra95] for G_i^*. We simply stated their arguments in an informal way.

From the above and the results on the definable search problems in bounded arithmetic (Theorem 2.22), we can derive the following, informal statements on the power of G_i and G_i^* for $i \geq 1$:

- G_i^* can simulate reasoning that involves objects that are computable in polynomial-time using a $\mathbf{\Sigma}_{i-1}^p$-oracle; and
- G_i can simulate reasoning that involves objects that are solutions for a $\mathbf{PLS}$-problem with $\mathbf{\Sigma}_{i-1}^p$-oracle.

We can consider G as a propositional proof system by focusing our attention to G-proofs with propositional endsequents, and similarly for G_i and G_i^* for $i \geq 0$. Then, by definition, G_0^* is tree-like PK and G_0 is PK, and by Krajíček's result in [Kra95] (Theorem 2.14), G_0^* and G_0 are p-equivalent w.r.t. propositional tautologies. Moreover, Krajíček also shows that G_1^* is p-equivalent to any *extended Frege system* w.r.t. propositional tautologies. This makes sense, since extended Frege systems are proof systems that reason about small circuits, which characterize polynomial-time, while we have argued that G_1^* is also a proof system that deals with polynomial-time computable objects.

We know the following hold, where $\leq_p^{prop}$ and $\equiv_p^{prop}$ denotes p-simulation and p-equivalence, respectively, w.r.t. propositional tautologies:

$$G_0^* \equiv_p^{prop} G_0 \leq_p^{prop} G_1^* \leq_p^{prop} G_1 \leq_p^{prop} \ldots \leq_p^{prop} G_i^* \leq_p^{prop} G_i \leq_p^{prop} \ldots \leq_p^{prop} G$$

Currently it is open whether G_1^* and G_1 are p-equivalent; in fact, it is open whether any p-simulation in the above can be made p-equivalence.

5.3 Basic QPC Proof Complexity

In this Section, we prove some basic results in QPC proof complexity. The following lemma is useful in the subsequent discussions.

Lemma 5.8. *Let $\exists x A(x)$ and B be QPC formulas and $A(B)$ be the result of substituting B for all occurrences of x in $A(x)$. The following four sequents have cut-free G_0^*-proofs of size $O(|A(B)|^2)$:*
(T1): $B, A(B) \to A(\mathsf{T})$,
(T2): $A(B) \to A(\mathsf{F}), B$,
(T3): $B, A(\mathsf{T}) \to A(B)$, and
(T4): $A(\mathsf{F}) \to B, A(B)$.

Proof. Simultaneous induction on the structure of $A(x)$. □

First of all, we prove upper bounds on the length of proofs in G and its variants.

Theorem 5.9. *(i) Every valid QPC sequent S has a cut-free G_0^* proof of size $2^{2^{O(|S|)}}$. (ii) ([Coo03]) Every valid sequent S of QPC sentences has a G-proof of size $2^{O(|S|)}$. (iii) Let $i \geq 0$. Every valid sequent consisting of $\Sigma_i^q \cup \Pi_i^q$-sentences has a G_i^*-proof of size $2^{O(|S|)}$.*

Proof. Item (i) is proven by the usual method for of the completeness of sequent calculus systems as follows. We maintain a tree of sequents whose root node is S and whose edges correspond to inference steps. Leaves are *active* if they are not logical axioms; otherwise they are *inactive*. Starting with the tree with single node S which is an active leaf, we repeatedly apply the following procedure: pick an active leaf L and pick an arbitrary, nonatomic formula A of L. Using the inference rule that introduces the outermost connective of A, grow the tree upward by attaching one or more upper sequents to L as new active leaves. L itself becomes inactive. For example, if L is $\Gamma \to \Delta_1, B \wedge C, \Delta$, then the new leaves are

$$\Gamma \to \Delta_1, B, \Delta_2 \text{ and } \Gamma \to \Delta_1, C, \Delta_2.$$

Note that there are appropriate exchange steps between L and the new leaves.

In the above procedure, if the inference step for L is $\exists$-left or $\forall$-right, the new leaf is obtained simply by introducing a new eigenvariable that doesn't occur elsewhere in the current tree. If the inference step is $\exists$-right, then the new leaf is obtained as follows:

$$\dfrac{\dfrac{\Gamma \to \Delta, \phi(\mathsf{T}), \phi(\mathsf{F})}{\Gamma \to \Delta, (\exists x)\phi(x), (\exists x)\phi(x)}\ \exists\text{-right}}{\Gamma \to \Delta, (\exists x)\phi(x)}\ \text{contraction-right}$$

And $\forall$-left is handled similarly.

Let c and q be the numbers of connectives and quantifiers that occur in S, respectively. The above procedure generates at most 2^{c2^q} sequents, each of which is length $O(2^q|S|)$. Thus, the resulting cut-free proof is of size $2^{2^{O(|S|)}}$.

Cook in [Coo03] showed (ii), and here we sketch his proof. The idea is to modify the above procedure so that no free variable occurs in active leaves. Suppose we picked an active leaf L and a sentence A of L. If A starts with a propositional connective, then apply the same procedure as above. Suppose that A is of the form $(\exists x)\phi(x)$ and is in the succedent of L. Since A is a sentence, it is either true or false. If it is true, then there is a truth value $u \in \{\mathsf{T}, \mathsf{F}\}$ such that $\phi(u)$ is a true sentence. Thus, a new active leaf is the sequent obtained by replacing A with $\phi(u)$. If A is false, then we simply derive A from $\phi(\mathsf{T})$. The case for $\forall$-left is handled similarly.

The cases for $\forall$-right and $\exists$-left are modified as follows. Suppose that L is $\Gamma \to \Delta, \forall x\phi(x)$, where A is the sentence $\forall x\phi(x)$. The new active leaves are constructed in the following way. Note that double lines indicate that we are omitting multiple applications of structural rules.

$$\dfrac{\dfrac{\dfrac{\phi(\mathsf{F}) \to b, \phi(b) \qquad \Gamma \to \Delta, \phi(\mathsf{F})}{\Gamma \to \Delta, \phi(b), b}\ \text{cut on } \phi(\mathsf{F}) \qquad \dfrac{b, \phi(\mathsf{T}) \to \phi(b) \qquad \Gamma \to \Delta, \phi(\mathsf{T})}{b, \Gamma \to \Delta, \phi(b)}\ \text{cut on } \phi(\mathsf{T})}{\Gamma \to \Delta, \phi(b)}\ \text{cut on } b}{\Gamma \to \Delta, (\forall x)\phi(x)}\ \forall\text{-right}$$

By Lemma 5.8, we have short proofs of the sequents $\phi(\mathsf{F}) \to b, \phi(b)$ and $b, \phi(\mathsf{T}) \to \phi(b)$. Thus, the new active leaves are $\Gamma \to \Delta, \phi(\mathsf{F})$ and $\Gamma \to \Delta, \phi(\mathsf{T})$.

Cook's procedure generates a tree-like G-proof with $2^{O(|S|)}$ sequents, each of which is of size $O(|S|)$, and hence (ii) holds. Moreover, if S is a $\Sigma_i^q \cup \Pi_i^q$-sentence, then all the cut formulas are either Σ_i^q or Π_i^q, and therefore (iii) follows. □

The following Lemma shows that our systems are natural extensions of Krajíček and Pudlák's systems.

Lemma 5.10. *G and KPG are p-equivalent. Moreover, for every $i \geq 1$, KPG_i and KPG_i^* are p-equivalent to G_i and G_i^*, respectively, w.r.t. $\Sigma_i^q \cup \Pi_i^q$.*

Proof. We prove that KPG_i and G_i are p-equivalent for every $i \geq 0$. The proof is identical for KPG_i^* versus G_i^* and KPG versus G.

KPG_i obviously p-simulates G_i with respect to proving valid $\Sigma_i^q \cup \Pi_i^q$-formulas. For the other direction, it suffices to show that G_i can simulate $\exists$-right and $\forall$-left steps in KPG_i with quantified targets. For the $\exists$-right case, let S be the sequent $\Gamma \to \Delta, \exists x A(x)$ which is derived from $\Gamma \to \Delta, A(B)$ with B quantified. G_i simulates this step by multiple inference steps in the following way using the sequents of the form (T1) and (T2) of Lemma 5.8:

$$\dfrac{\dfrac{\dfrac{(T1)\quad B, A(B) \to A(\mathsf{T})}{B, A(B) \to \exists x A(x)}\,\exists\text{-right} \quad \dfrac{(T2)\quad A(B) \to A(\mathsf{F}), B}{A(B) \to \exists x A(x), B}\,\exists\text{-right}}{\dfrac{A(B) \to \exists x A(x)}{A(B), \Gamma \to \Delta, \exists x A(x)}} \text{Cut} \qquad \dfrac{\Gamma \to \Delta, A(B)}{\Gamma \to \Delta, \exists x A(x), A(B)}}{\Gamma \to \Delta, \exists x A(x)}\,\text{Cut}$$

G_i can cut $A(B)$ since, by the definition of KPG_i, $A(B)$ is $\Sigma_i^q \cup \Pi_i^q$. Note that the targets of $\exists$-right steps in the above proof are constants T and F.

The simulation of $\forall$-left steps in KPG_i by G_i is done in an analogous way using (T3) and (T4) of Lemma 5.8. □

In fact, the proof of Lemma 5.10 shows the following:

Theorem 5.11. *Let AG be G with an additional restriction that the target of $\exists$-right and $\forall$-left be atomic. Then AG p-simulates G w.r.t. QPC. Similarly, for every $i \geq 0$, AG_i and AG_i^* p-simulate G_i and G_i^* w.r.t. $\Sigma_i^q \cup \Pi_i^q$, respectively.*

It is not known whether AG_i and AG_i^* p-simulate G_i and G_i^* w.r.t. Σ_{i+1}^q, respectively. For this reason, we allow non-atomic targets in Definition 5.7.

Let π be a G-proof. Free variables of π that occur in the endsequent are called *parameter variables*. Below we follow Buss [Bus86, Bus98b] in defining the *free variable normal form* of proofs, which is slightly stronger than Takeuti's *regular proofs* [Tak87].

Definition 5.12. *([Bus86, Bus98b]) Let π be a tree-like G-proof. We say that π is in* free variable normal form *if the following conditions are met: (i) no parameter variable is used as an eigenvariable; and (ii) every nonparameter variable is used as an eigenvariable exactly once in π.*

If π is a tree-like proof in free variable normal form, it follows that, for every nonparameter variable b, the sequents containing an occurrence of b form a subtree of π whose root is the upper sequent of the inference in which b is used as the eigenvariable. Any tree-like proof can be converted into free variable normal form by renaming bound variables and replacing nonparameter variables b with the logical constant T if b is never used as an eigenvariable. Throughout this dissertation, we assume that all tree-like QPC proofs are in free variable normal form.

Let A be an arbitrary QPC formula. A *prenexification* of A is any prenex formula A^* equivalent to A that is obtained from A by moving the quantifiers to the front in an appropriate manner. In general, A has multiple prenexifications; however, if A is already prenex or A is quantifier-free, then A has a unique prenexification, which is A itself.

Let A^* be an arbitrary prenexification of A. We write A^* as $Q\vec{x}A'$, where

$$Q\vec{x} =_{syn} Q_1 x_1 \ldots Q_k x_k$$

is a sequence of quantifiers with each $Q_i \in \{\forall, \exists\}$ and A' is quantifier-free. The *dual of* $Q\vec{x}$ is a sequence $Q^d\vec{x}$ of quantifiers such that, for every $i \in [1,k]$, Q_i^d is $\exists$ if Q_i is $\forall$, and Q_i^d is $\forall$ if Q_i is $\exists$.

Theorem 5.13. *Let $i \geq 1$. Define $\hat{G}_i^*$ to be G_i^* with an additional restriction that all cut formulas be prenex Σ_i^q. Then $\hat{G}_i^*$ is p-equivalent to G_i^* w.r.t. QPC formulas.*

Proof. Let $i \geq 1$ be arbitrary. Let $\pi = S_1, \ldots, S_k$ be a G_i^*-proof. Since π is tree-like, every formula in π is an ancestor of either an endformula or a cut formula but not both.

Let A be a cut formula of π. Assume without loss of generality that A is Σ_i^q, since otherwise we can simply insert the $\neg$-steps just before the cut step so that the cut formula is $\neg A$. Fix an arbitrary prenexification A^* of A with $A^* \in \Sigma_i^q$. Denote A^* as $Q\vec{x}A'$, where A' is quantifier-free. Let B be an ancestor of A. Then there is a subformula β of A such that B is the result of substituting propositional formulas for x-variables that occur free in β; we say B corresponds to β. For every ancestor B of A, we fix a prenexification $B^* =_{syn} R\vec{x}B'$ with B' quantifier-free so that it satisfies the following: (i) if B corresponds to a subformula β of A that is within the scope of an even number of negations, then $R\vec{x}$ is a subsequence of $Q\vec{x}$; and (ii) if β is within the scope of an odd number of negations, then $R\vec{x}$ is a subsequence of $Q^d\vec{x}$.

For each i, define sequent S_i' by replacing every ancestor B of A in S_i with B^*. We show that $S_1' \ldots, S_i'$ for each $i \in [1,k]$ can be converted into a G_i^*-proof with only a polynomial blowup in the proof size by induction on i. It suffices to show show to derive S_i' from preceding sequents when S_i is the lower sequent of an inference step whose principal formula is an ancestor B of A such that B is not identical to B^*.

The cases in which S_i is derived by exchange or contraction are trivial. If B is introduced into S_i by weakening, S_i' is obtained by a weakening step introducing B^*. The cases for quantifier rules are also easy because of the way we define B^*.

Suppose that the step deriving S_i is $\neg$-left:

$$\dfrac{[S_j] \qquad \Gamma \to \Delta, B}{[S_i] \quad \neg B, \Gamma \to \Delta}$$

Then $B^* =_{syn} R\vec{x}B'$ and $(\neg B)^* =_{syn} R^d\vec{x}\neg B'$, where $R^d\vec{x}$ is the dual of $R\vec{x}$. S_i' is derived from S_j' as follows:

$$\dfrac{\dfrac{[*] \quad R\vec{x}B, R^d\vec{x}\neg B \rightarrow}{R\vec{x}B, R^d\vec{x}\neg B, \Gamma \rightarrow \Delta} \quad \dfrac{\Gamma \rightarrow \Delta, R\vec{x}B \quad [S_j']}{R^d\vec{x}\neg B, \Gamma \rightarrow \Delta, R\vec{x}B}}{[S_i'] \quad R^d\vec{x}\neg B, \Gamma \rightarrow \Delta} \text{ Cut on } R\vec{x}B$$

Note that the sequent marked $[*]$ has short cut-free proof. The case for $\neg$-right is analogous.

The cases for $\vee$-right, $\vee$-left, $\wedge$-right, and $\wedge$-left are all similar. We present below the $\wedge$-right case. Assume that S_i is derived from S_j an S_k:

$$\dfrac{[S_j] \quad \Gamma \rightarrow \Delta, B \qquad \Gamma \rightarrow \Delta, C \quad [S_k]}{[S_i] \quad \Gamma \rightarrow \Delta, B \wedge C}$$

Then the prenexifications B^*, C^*, and $(B \wedge C)^*$ are of the following form:

$$\begin{aligned} B^* &=_{syn} R\vec{x}B' \\ C^* &=_{syn} P\vec{y}C' \\ (B \wedge C)^* &=_{syn} Q\vec{z}(B' \wedge C') \end{aligned}$$

where both $R\vec{x}$ and $P\vec{y}$ are subsequences of $Q\vec{z}$. Then S_i' is derived first from S_j' and another sequent that we mark '$\diamond$':

$$\dfrac{[\diamond] \quad R\vec{x}B', \Gamma \rightarrow \Delta, Q\vec{z}(B' \wedge C') \qquad \dfrac{\Gamma \rightarrow \Delta, R\vec{x}B' \quad [S_j']}{\Gamma \rightarrow \Delta, Q\vec{z}(B' \wedge C'), R\vec{x}B'}}{\Gamma \rightarrow \Delta, Q\vec{z}(B' \wedge C')} \text{ Cut on } B^*$$

The sequent $\diamond$ follows from S_k' as below:

$$\dfrac{\dfrac{[*] \quad P\vec{y}C', R\vec{x}B' \rightarrow Q\vec{z}(B' \wedge C')}{P\vec{y}C', R\vec{x}B', \Gamma \rightarrow \Delta, Q\vec{z}(B' \wedge C')} \quad \dfrac{\Gamma \rightarrow \Delta, P\vec{y}C' \quad [S_k']}{R\vec{x}B', \Gamma \rightarrow \Delta, Q\vec{z}(B' \wedge C'), P\vec{y}C'}}{[\diamond] \quad R\vec{x}B', \Gamma \rightarrow \Delta, Q\vec{z}(B' \wedge C')} \text{ Cut on } C^*$$

Note that the sequent $(*)$ has a short cut-free proof.

By repeating the above procedure for every cut formula and all of its ancestors, we can construct π' in which all cut formulas are prenex Σ_i^q. Clearly the size of π' is polynomial in $|\pi|$. $\square$

Proving an analogous assertion for G_i is more complicated since, in a G_i-proof, a formula can be an ancestor of both an endformula and a cut formula. In fact, we prove in Section 6.3 that, if Theorem 5.13 holds for G_i, then there is a complexity-theoretic consequence (Theorem 6.10). Nonetheless, we expect the following to hold: for every $i \geq 1$, every G_i-proof π of a prenex formula can be converted into another G_i-proof π' of the same formula such that (i) all cut formulas in π' are prenex; and (ii) the size of π' is polynomial in $|\pi|$.

5.4 $\mathbf{TC}^0$ and the Proof Theory for QPC

Much of the existing work on the relationship between QPC sequent calculus and bounded arithmetic [KP90, Kra95] depends on the fact that many of the parsing operations for sequent calculus proofs are polynomial-time and in fact definable in S^1_2; see Section 5.5.

The goal of this section is to show that such parsing operations as well as other proof-theoretic operations for QPC belong to $\mathbf{TC}^0$, a class apparently much smaller than $\mathbf{P}$. First, in Section 5.4.1, we demonstrate that many basic parsing operations are in $\mathbf{TC}^0$. Based on this fact, we prove the $\mathbf{TC}^0$-versions of Gentzen's midsequent theorem and Herbrand's theorem for G_0 and G_0^*. This is an interesting metamathematical statement in its own right, but it is also useful in the study of the witnessing problem for G_0 in Chapter 6 and its relationship to the second-order theory $\mathbf{VNC}^1$ of bounded arithmetic in Chapter 8. Also the result in Section 6.4 on the complexity of the $T\Sigma^q_1$-witnessing problems directly depend on the results of this Section.

Note that the standard proofs of Gentzen's midsequent theorem and Herbrand's theorem for the first-order sequent calculus LK are based on cut elimination [Bus98b], which involves an exponential blow-up of proofs size. However, in the current context, there is no need for us to carry out cut-elimination in $\mathbf{TC}^0$, since the proofs in G_0 and G_0^* already satisfy the necessary property that all cut formulas be quantifier-free.

We believe that many of the results in this Section can be formalized in a suitable theory

corresponding to $\mathbf{TC}^0$, such as $\mathbf{VTC}^0$ of Nguyen and Cook [NC04]. Proving this assertion is out of the scope of this dissertation, and we leave it as a conjecture.

5.4.1 Parsing Operations

Since we are concerned with a low-level complexity class $\mathbf{TC}^0$, we need to be more specific about how QPC syntax is encoded. We fix the alphabet Γ_{QPC} so that G-proofs are represented as strings over it:

$$\Gamma_{QPC} = \{\mathsf{T}, \mathsf{F}, p, x, 0, 1, (,), \wedge, \vee, \neg, \exists, \forall, \rightarrow, comma, \#\},$$

where $comma$ denotes the comma. The symbols 0 and 1 are used to denote the indices of variables. A variable p_i is written as $p^\frown i$ with $i \in \{0,1\}^+$, where $^\frown$ is string concatenation, and similarly for x_i. Formulas and sequents are encoded as strings over Γ_{QPC} without the sharp symbol (#). G-proofs are representable as $S_1\#S_2\#\ldots\#S_m$; that is, a sequence of sequents separated by the sharp (#) symbol such that every S_i is either an initial sequent or derived from at most two preceding sequents by an appropriate inference rule. We call this the *sequence representation of proofs*. Later we will discuss how to represent tree-like proofs as strings over Γ_{QPC}, since parsing tree-like proofs in $\mathbf{TC}^0$ causes subtle problems. We also fix an encoding scheme for truth assignments.

Throughout this subsection, Z will always denote a finite string over Γ_{QPC} and $n = |Z|$. For $1 \leq i \leq n$, α_i is the ith symbol of Z, i.e.,

$$Z =_{syn} \alpha_1\alpha_2\ldots\alpha_n.$$

For $1 \leq i \leq j \leq n$, we write $Z[i,j]$ to denote the substring $\alpha_i\ldots\alpha_j$ of Z. If $j < i$ then $Z[i,j]$ is empty.

Our goal is to show that various relations of the form $R \subseteq \mathbb{N}^k \times (\Gamma_{QPC})^*$ belongs to $\mathbf{TC}^0$. An example of such R is $R(i,j,Z)$ which is true iff $Z[i,j]$ encodes a QPC formula. We use the method of descriptive complexity of Section 2.1.2: in order to show that $R(i,j,Z)$ is in

$\mathbf{TC}^0$, we argue that there is an FOM formula $\phi(i,j)$ that represents R. Note that the number arguments of R correspond to the free variables of ϕ, and (the binary encoding of) the string argument Z is implicitly described by the Bit predicate of FOM.

To simplify the presentation, we will write 'C is $\mathbf{TC}^0$-recognizable' to mean that the relation $R(i,j,Z)$ is in $\mathbf{TC}^0$, where $R(i,j,Z)$ holds iff $Z[i,j]$ is an encoding of C. Here C could be a formula, QPC formula, G-proof and so on.

Buss in [Bus91] shows that $\mathbf{NC}^1$ can parse propositional formulas and also recognize Frege proofs. His proofs actually show that these can be done in $\mathbf{TC}^0$. It is easy to extend this fact to the QPC case:

Lemma 5.14. *Formulas, QPC formulas, and sequents are all* $\mathbf{TC}^0$*-recognizable.*

Proof. (Sketch) In order to show that formulas are $\mathbf{TC}^0$-recognizable, it suffices to ensure that (i) $Z[i,j]$ is correctly parenthesized, and that (ii) there is no substring of $Z[i,j]$ of length 2 that is impossible in a formula. Examples of impossible substrings are px, $x)$, $\neg\exists$, etc. Condition (i) holds iff (i-a) $Z[i,j]$ contains an equal number of occurrences of '(' and ')'; and (i-b) for all $u \in [i, j-1]$, the number of occurrences of '(' in $Z[i,u]$ exceeds that of ')'. It is easy to see that both (i-a) and (i-b) are FOM-expressible.

For QPC formulas, we only need to check that $Z[i,j]$ is a formula and that every occurrence of an x-variable x_k in $Z[i,j]$ is within the scope of either $\exists x_k$ or $\forall x_k$. Checking whether x_k is in the scope of a quantifier can be done by counting the numbers of parentheses, and therefore QPC formulas are $\mathbf{TC}^0$-recognizable.

We omit a proof for sequents, which is easily seen to be FOM-expressible. □

Based on Lemma 5.14, it is easy to see that PK-proofs are $\mathbf{TC}^0$-recognizable, since all we need to do is to verify in $\mathbf{TC}^0$, for each inference step, the simple syntactic relationship between the auxiliary formulas and the principal formula. Recognizing G-proofs is a bit more tricky because of the more complex relationship between the auxiliary formula and the principal formula of a quantifier rule, especially $\exists$-right and $\forall$-right. More specifically, for $\exists$-right and

$\forall$-left, we need to show that it is $\mathbf{TC}^0$ to verify that the auxiliary formula A and the principal formula (QxA') satisfies the following: there is a propositional subformula B of A such that, if we replace every occurrence of x in A' with B, the result is identical to A. Note that verifying this condition requires working on internal structure of formulas.

We introduce the notion of *identifier* of a subformula. Let A be a formula. For each subformula B of A, we define its identifier (ID) as the string over $\{1, 2\}$ that uniquely determines its location within A as a path from the root of A to the root of B, thinking of A as a tree.

Definition 5.15. *Let A be a formula. For every subformula B of A, $ID_A(B)$ is defined inductively as follows. (1) $ID_A(A)$ is the empty string ϵ; (2) if B is $(B_1 \odot B_2)$ with $\odot \in \{\wedge, \vee\}$, then $ID_A(B_1)$ is $ID_A(B)^\frown 1$ and $ID_A(B_2)$ is $ID_A(B)^\frown 2$; and (3) if B is either $(\neg B_1)$ or (QxB_1), then $ID(B_1)$ is $ID_A(B)^\frown 1$.*

For example, if A is $(\exists x_1(B \vee (\neg C)))$, then $ID_A(B) = 11$ and $ID_A(C) = 121$.

Lemma 5.16. *The following is a $\mathbf{TC}^0$-function: given Z, i, j, a, b such that $i \leq a < b \leq j$ and $Z[i, j]$ and $Z[a, b]$ encode formulas A and B, respectively, output $ID_A(B)$.*

Proof. It suffices to show that the bit graph of $ID_A(B)$ is in $\mathbf{TC}^0$. First, the ith symbol of $ID_A(B)$ is nonempty iff there exists l, m satisfying (i) $Z[l, m]$ is a subformula of A; (ii) B is a subformula of $Z[l, m]$; and (iii) the number of '(' in $Z[i, l-1]$ minus the number of ')' in $Z[i, l-1]$ is equal to i.

Suppose that the ith bit of $ID_A(B)$ is nonempty. Then the ith bit is '2' iff $\alpha_{l-1} \in \{\wedge, \vee\}$.

□

Now we can prove the following.

Theorem 5.17. *The following are $\mathbf{TC}^0$-recognizable: G-proofs, and G_i-proofs for every $i \geq 0$.*

Proof. We prove that G_i-proofs are $\mathbf{TC}^0$-recognizable, since the case for G-proofs is completely analogous. Our goal is to show that there is an FOM formula $\phi(i, j)$ which holds if and only if $Z[i, j]$ is a G_i-proof. By Lemma 5.14, we already have an FOM formula that holds

iff $Z[i,j]$ is of the form $S_1\#S_2\#\ldots\#S_m$, where each S_i is a sequent. It remains to show that the following relation is FOM-expressible: for every sequent S_k in the proof, either S_k is an initial sequent or S_k follows from one or two preceding sequent(s). The only nontrivial case is recognizing S_k that follows from S_j with $j < k$ by a quantifier-introduction rule.

We describe how to decide, given (j,k), whether S_k follows from S_j by an application of the $\exists$-right rule. It is easy to write an FOM formula asserting that the two sequents are identical except for the principal and auxiliary formulas and that the principal formula is of the form $(\exists x_i A)$. Let A' stand for the auxiliary formula. We need to verify that there is a propositional subformula B of A' such that A' is identical to the result of substituting B for every occurrence of x in A. This is expressed in FOM as follows: there exists a propositional subformula B of A' such that, for all subformula C of A, there exists a subformula C' of A' with $ID_A(C) = ID_{A'}(C')$ such that the following hold:
- if C is the atomic formula (x_i), then C' is B;
- if C is atomic but not (x_i), then C and C' are identical; and
- if C is not atomic, then C and C' have the same outer connective.

Note that the above conditions are expressible in FOM. The $\forall$-left rule is handled analogously. For $\exists$-left and $\forall$-right, we need to ensure that the eigenvariable condition holds, which is easily done in FOM. □

There are some proof-theoretic operations that seem unlikely to be in $\mathbf{TC}^0$. In particular, there are three such operations: (i) recognition of tree-like proofs; (ii) recognition of ancestors of the given sequent in a proof; and (iii) recognition of ancestors of the given formula in a proof. For (i), we do not know how to verify that each sequent in a proof is used as an upper sequent at most once. The reason we suspect that (ii) and (iii) are not in $\mathbf{TC}^0$ is that the reachability in directed acyclic graphs is complete for nondeterministic logspace, and (ii) and (iii) are basically the reachability problems whose edges are defined by inference steps.

In order to bypass the difficulty of detecting 'tree-likeness' of a proof in sequence representation, we fix a special encoding for tree-like proofs.

Definition 5.18. *Let π be a tree-like proof with S its endsequent. The* bracket representation *Π_S of π is defined inductively as follows. If π consists of a single initial sequent S, then Π_S is $[S]$. If π ends with a step deriving S from S', then Π_S is $[\Pi_{S'} \quad S]$. If π ends with a step deriving S from S' and S'', Π_S is $[\Pi_{S'} \quad \Pi_{S''} \quad S]$.*

The size of Π_S is linear in $|\pi|$. Another advantage of this encoding is that $\mathbf{TC}^0$ can recognize ancestors of a sequent in a bracket representation proof by counting the number of brackets. This is crucial in the $\mathbf{TC}^0$ midsequent theorem that we prove in the next subsection. The following is easily seen to hold:

Theorem 5.19. *For every $i \geq 0$, G_i^*-proofs in bracket representation are $\mathbf{TC}^0$-recognizable.*

The proofs of Theorems 5.17 and 5.19 easily generalize to the $\mathbf{TC}^0$-recognizability of first-order sequent calculus proofs whose underlying language and nonlogical axioms are $\mathbf{TC}^0$-recognizable. We know from our email correspondence with Samuel Buss that this fact has been known to Buss and possibly a few others, but, as far as we know, it has not been explicitly stated in print. $\mathbf{TC}^0$ is widely believed to be the smallest complexity class in which counting is possible. Since parsing operations require counting in general, apparently $\mathbf{TC}^0$ is the smallest class containing those parsing operations.

In his comments on a preliminary version of this dissertation, Jan Krajíček asked whether the complexity of parsing and other proof-theoretic operations could be further reduced to $\mathbf{AC^0}$ by imposing more structure on the representations of proofs. This is quite plausible, but we do not pursue this direction in this dissertation.

5.4.2 Herbrand's Theorem and the $\mathbf{TC}^0$ Midsequent Theorem for G_0^*

KPG_0 and KPG_0^* are quantifier-free propositional proof systems (PK and tree-like PK, respectively), and thus our G_0 and G_0^* are, as far as we know, new quantified proof systems for the whole QPC that have never been studied. Since the cut formulas of G_0 and G_0^* are quantifier-free, they are similar to first-order theories T axiomatized by purely universal formulas, all of

whose theorems have LK derivations with all cuts on quantifier-free formulas by cut elimination [Tak87, Bus98b] (also see Theorem 7.20). Therefore, it is reasonable to attempt to obtain for G_0 and G_0^* the counterparts of the proof-theoretic statements regarding such theories T.

In this section we will prove Gentzen's midsequent theorem for G_0^* and related statements, such as Herbrand's theorem. That these statements hold for the QPC systems is not surprising. Our focus is on the lengths of proofs and the complexity of the operations that transform proofs into certain normal forms, which traditional proof theory has not paid much attention to. Based on the results of the preceding subsection, we will show the complexity of these proof transformations to be $\mathbf{TC}^0$. As we stated in the opening remarks of Section 5.4, we do not carry out cut elimination in $\mathbf{TC}^0$. Our proofs in this section depend crucially on the fact that every cut formula in G_0^* is quantifier-free.

Definition 5.20. *Let π be a sequent calculus proof of a quantified formula A in either predicate calculus or QPC. We say that π is a* midsequent proof *if there is a quantifier-free sequent S in π satisfying the following: (i) no quantified formula appears in the derivation of S in π; and (ii) the endsequent is derived from S using contraction, exchange, $\exists$-right, and $\forall$-right only.*

Gentzen proved, in his 1935 paper which introduced the sequent calculus LK, the midsequent theorem for first-order logic. The definition of LK is found in Chapter 7 (as Definition 7.18).

Theorem 5.21. *([Gen35]) If A is a valid first-order formula in prenex form, then there is a midsequent LK-proof of A.*

Proof. (Sketch) Since A is a valid prenex formula, it has a cut-free LK-proof π [Tak87, Bus98b]. Gentzen demonstrates an algorithm to convert π into a midsequent proof π'. Intuitively, his algorithm pushes quantifier steps in π downward so that no propositional step occurs below a quantifier step. We present a simple example. Suppose that π contains $\exists$-right steps followed by $\vee$-left step:

$$\dfrac{\dfrac{B, \Gamma \to \Delta, A(t)}{B, \Gamma \to \Delta, \exists x A(x)}\,\exists\text{-right} \quad \dfrac{C, \Gamma \to \Delta, A(t')}{C, \Gamma \to \Delta, \exists x A(x)}\,\exists\text{-right}}{B \vee C, \Gamma \to \Delta, \exists x A(x)}\,\vee\text{-left}$$

We modify the proof above so that $\exists$-right steps occur below the $\vee$-left step:

$$\dfrac{\dfrac{\dfrac{\dfrac{B, \Gamma \to \Delta, A(t) \qquad C, \Gamma \to \Delta, A(t')}{B \vee C, \Gamma \to \Delta, A(t), A(t')}\,\vee\text{-left}}{B \vee C, \Gamma \to \Delta, A(t), \exists x A(x)}\,\exists\text{-right}}{B \vee C, \Gamma \to \Delta, \exists x A(x), \exists x A(x)}\,\exists\text{-right}}{B \vee C, \Gamma \to \Delta, \exists x A(x)}\,\text{contraction-right}$$

The double lines in the above proof indicate that we omitted multiple applications of structural rules.

By repeatedly applying the above procedure, all quantifier introduction steps can be pushed downward so that the result is a midsequent proof π': see [Gen35, Tak87] for more details. □

Clearly, the midsequent theorem holds for G, and this fact has been pointed out by Krajíček in [Kra95]. Note that Gentzen's construction of π' from π in the above proof is polynomial-time in $|\pi|$. Because of the sequential manner in which this construction works, it is not clear if this can be done in a smaller complexity class. Using a different idea, we show below that there is a $\mathbf{TC}^0$-function that, given π, outputs a midsequent proof of the same endsequent.

The π-Prototypes and Herbrand Disjunctions

Recall that all the cut formulas in a G_0-proof are quantifier-free. It follows that, if the antecedent of the endsequent is quantifier-free, then the proof contains neither $\forall$-left nor $\exists$-left step.

Definition 5.22. *Suppose that π is a G_0-proof of a quantified QPC formula A in prenex form. If a quantifier-free formula A' is the auxiliary formula of a quantifier-introduction step, then we call it a* π-prototype *of A, and we call this quantifier-introduction step the* critical step of A'. *We define the* Herbrand π-disjunction *to be the sequent*

$$\to A_1, \ldots, A_m$$

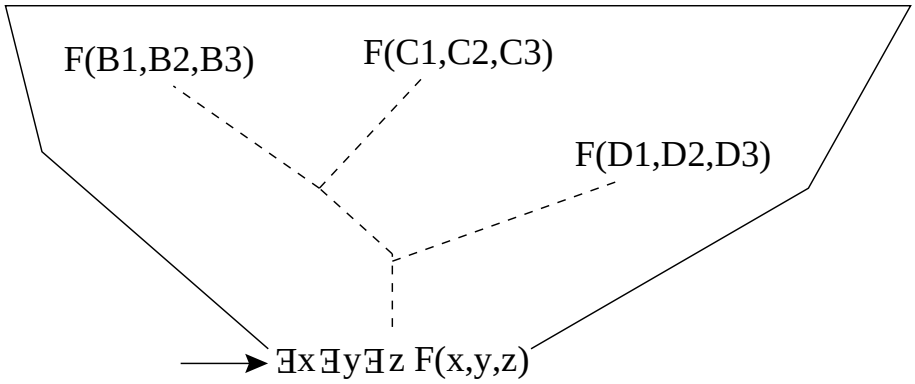

Figure 5.2: A G_0^*-proof of a prenex Σ_1^q-formula $\exists x \exists y \exists z F(x, y, z)$.

where $A_1, \ldots, A_m$ are all the π-prototypes of A.

It is easy to see that π-prototypes are $\mathbf{TC}^0$-recognizable. As an example, consider a G_0^*-proof π of a prenex Σ_1^q-formula A of the form

$$\exists x \exists y \exists z F(x, y, z)$$

that is depicted in Figure 5.2, where $F(x, y, z)$ is quantifier-free. In π, there are three formulas, $F(B1, B2, B3)$, $F(C1, C2, C3)$, and $F(D1, D2, D3)$, that are quantifier-free ancestors to which $\exists$-right is applied; that is, each of these three formulas is a π-prototype of the endformula. The Herbrand π-disjunction is the sequent

$$\rightarrow F(B1, B2, B3), F(C1, C2, C3), F(D1, D2, D3)$$

and we will later prove that this sequent is valid and has a short PK-proof (Lemma 5.25).

Assume that π is a G_0-proof of the sequent $\rightarrow A$ where A is of the form

$$Q_1 x_1 \ldots Q_k x_k F(\vec{p}, x_1, \ldots, x_k) \tag{5.3}$$

with $Q_i \in \{\exists, \forall\}$ for each $i \in [1, k]$ and F quantifier-free. Then, for every π-prototype A' of A, there exists a unique sequence $B_1, \ldots, B_k$ of propositional formulas such that

$$A' =_{syn} F(\vec{p}, B_1, \ldots, B_k).$$

Intuitively, each B_i is either the target of the $\exists$-right step or the eigenvariable of the $\forall$-right step that introduces the bound variable x_i into A'. In Figure 5.2, $B1$, $B2$, and $B3$ are the first, second, and third component of the π-prototype $A(B1, B2, B3)$, respectively.

Definition 5.23. *Let π, A, A', and $B_1, \ldots, B_k$ be as in the preceding paragraph. For each $i \in [1, k]$, we call B_i* the ith component of A'.

Let A' be a π-prototype of A of the form (5.3). For $i \in [1, k]$, the ith component of A' is the propositional formula B that occurs as a subformula of A' such that, for every occurrence of B in A', there exists an occurrence of (x_i) in F satisfying $ID_{A'}(B) = ID_F(x_i)$. Thus, ith components are $\mathbf{TC}^0$-recognizable.

Before proving that some proof-theoretic operations, including the construction of a midsequent proof, can be done in $\mathbf{TC}^0$, we state a useful lemma:

Lemma 5.24. *There is a $\mathbf{TC}^0$-function f satisfying the following: if S_1 and S_2 are two sequents such that S_2 can be derived from S_1 using structural rules only, then $f(S_1, S_2)$ is such a derivation. The derivation can be either in sequence representation or in bracket representation.*

Proof. (Sketch) $\mathbf{TC}^0$ can easily verify that S_2 does follow from S_1 by structural rules only. Also there is a $\mathbf{TC}^0$-function that, given (S_1, S_2), outputs a sequence $T_1, \ldots, T_k$ of sequents such that: (i) $T_1 =_{syn} S_1$, (ii) $T_k =_{syn} S_2$, and (iii) for each $i \in [2, k]$, T_i follows from T_{i-1} by one application of either contraction or weakening as well as multiple exchange steps before and after it. Finally, it is easy to see that there is another $\mathbf{TC}^0$ function that, given $T_1, \ldots, T_k$, inserts necessary exchange steps between T_i's so that the output is a desired proof of S_2 from S_1. Since the class of $\mathbf{TC}^0$-functions is closed under composition, the claim holds. □

The following lemma will be useful in both the midsequent theorem and Section 6.2 when we study the complexity of the Σ_1^q-witnessing problem of G_0.

Lemma 5.25. *Let π be a G_0-proof of a sequent $\rightarrow A$ with A a quantified QPC formula in prenex form. Then the Herbrand π-disjunction is valid, and there is a $\mathbf{TC}^0$-function f such that $f(\pi)$ is a PK-proof of the Herbrand π-disjunction.*

Proof. Assume that π is the sequence $S_1, \ldots, S_k$ of sequents, where S_k is $\rightarrow A$. We construct a polynomial-size PK-proof of the Herbrand π-disjunction as follows. For every $i \in [1, k]$, if sequent S_i is $\Gamma_i \rightarrow \Delta_i$, then define S_i' to be $\Gamma_i \rightarrow \Delta_i'$, where Δ_i' is obtained from Δ_i by removing all quantified formulas and adding all π-prototypes $A_1, \ldots, A_m$. Note that Γ_i contains no quantified formula, and S_k' is the Herbrand π-disjunction.

We argue that every S_i' has a PK-proof of size polynomial in $|\pi|$ by induction on i. If S_i contains no quantified formula, then $S_i =_{syn} S_i'$ and there is nothing to prove. Assume that S_i contains a quantified formula. The only nontrivial case is when S_i is derived from S_j which does not contain a quantified formula, and this happens only in weakening or quantifier introduction. In either case, S_i' follows from S_j' by introducing $A_1, \ldots, A_m$ by weakening.

We argue that the above construction is $\mathbf{TC}^0$. First, there is a $\mathbf{TC}^0$-function that, given π, outputs a sequence $S_1', \ldots, S_k'$. This sequence can be easily converted to a PK-proof by a $\mathbf{TC}^0$, since it only requires removing redundant sequents and inserting necessary structural steps between sequents, as in Lemma 5.24. □

We emphasize that, in Lemma 5.25, the validity of the Herbrand π-disjunction in depends crucially on the fact that all cut formulas in G_0-proofs are quantifier-free.

Below we state and prove a $\mathbf{TC}^0$ midsequent theorem for G_0^*.

Theorem 5.26. *(The $\mathbf{TC}^0$ Midsequent Theorem for G_0^*) Let π be a G_0^*-proof of a prenex formula A. Then there exists a G_0^*-proof π' of A such that π' is a midsequent proof whose size is polynomial in $|\pi|$. Moreover, there is a $\mathbf{TC}^0$-function that, given π, outputs such a midsequent proof π'.*

Proof. Let A be of the form

$$Q_1 x_1 \ldots Q_k x_k F(\vec{p}, x_1, \ldots, x_k)$$

with F quantifier-free and $Q_i \in \{\exists, \forall\}$ for each $i \in [1, k]$. Let $A_1, \ldots, A_m$ be all the π-prototypes. By Lemma 5.25, there is a $\mathbf{TC}^0$-function f such that $f(\pi)$ is a PK-proof of the

Herbrand π-disjunction

$$S_\pi =_{syn} \quad \rightarrow A_1, \ldots, A_m.$$

It suffices to show that there is a $\mathbf{TC}^0$-function g such that $g(\pi)$ is a G_0^*-proof of $\rightarrow A$ from S_π using contraction, exchange, $\forall$-right and $\exists$-right rules only.

Assume that π is in bracket representation and consists of t sequents $S_1, \ldots, S_t$. For every $i \in [1, t]$, we say that S_i is *at level* l if there are l inference steps between S_i and the endsequent S_t. Let l^* be the maximum level in π. Thus, S_t is at level zero. Note that l^* is $\mathbf{TC}^0$-computable. Also, for each $l \in [0, l^*]$, the sequents at level l are $\mathbf{TC}^0$-recognizable.

For each $l \in [0, l^*]$, we define the sequent $S(l)$ to be

$$\rightarrow \Pi(l), \Delta(l)$$

where $\Pi(l)$ is the cedent of all quantified formulas that occur in a sequent at level l, and $\Delta(l)$ is the cedent containing all π-prototypes A' such that the lower sequent of the critical step of A' is at level l' with $l' < l$. Thus, $S(l)$ contains π-prototypes and quantified descendents of π-prototypes only.

Let ρ be the sequence $S(l^*), S(l^* - 1), \ldots, S(1), S(0)$ of sequents. Note that $S(l^*)$ is identical to S_π up to applications of exchanges and that $S(0)$ is identical to the endformula. Moreover, for every $l \in [1, l^*]$, for every formula B in $S(l)$, at least one the following holds: (i) B is present in $S(l-1)$; or B is not present in $S(l-1)$ but it follows from a formula B' in $S(l-1)$ by either $\forall$-right or $\exists$-right. Moreover, if B follows from B' by $\forall$-right, then every occurrence of the eigenvariable b is in B'; in particular, b occurs neither in $S(l)$ nor in $S(l-1)$ outside B'.

It is easy to see that there is a $\mathbf{TC}^0$-function h such that, given $(S(l-1), S(l)$ as an input, outputs a G_0^*-proof of $S(l)$ from $S(l-1)$ by inserting multiple applications of exchange, contraction, $\forall$-right, and $\exists$-right. The $\mathbf{TC}^0$-function g can be constructed using h. □

If π is a G_0-proof instead of G_0^*-proof, our proof above does not work, since the Herbrand π-disjunction S_π may contain two π-prototypes A_1 and A_2 such that an eigenvariable b occurs

in both. If this is the case, there is no G_0-proof of the endformula from S_π because $\forall$-right rule on b can be applied neither to A_1 nor A_2. Thus, it is open whether a G_0-proof of a prenex formula can be converted to a midsequent proof without an exponential blowup of the proof size. If all quantifier-introduction steps in π are $\exists$-right, then this problem never arises, and thus we can prove the following weaker statement for G_0:

Theorem 5.27. *There is a* $\mathbf{TC}^0$*-function* f *such that, if* π *is a* G_0*-proof of a prenex* Σ_1^q*-formula* A*, then* $f(\pi)$ *is a midsequent proof of* A*.*

Proof. Since the endformula is Σ_1^q, all quantifier-introduction steps in π are $\exists$-right. From the Herbrand π-disjunction $\rightarrow A_1, \ldots, A_m$ we can easily derive a sequent containing m copies of A by repeated applications of $\exists$-right rule. □

It is shown by Krajíček that tree-like PK p-simulates dag-like PK ([Kra95], Theorem 2.14 in this dissertation) and therefore G_0^* p-simulates G_0 w.r.t. propositional tautologies. Based on this and the above results, we obtain a stronger p-simulation for G_0^* and G_0.

Theorem 5.28. G_0^* *p-simulates* G_0 *w.r.t. prenex* Σ_1^q*-formulas.*

Proof. Let π be a G_0-proof of a sequent S containing one prenex Σ_1^q-formula. Apply Theorem 5.27 to obtain π' that proves S from the Herbrand π-disjunction S_π. Since the subproof π_1 of π' rooted at S_π is a PK-proof, and tree-like PK p-simulates PK [Kra95], we can convert π_1 into a tree-like PK-proof π_2 with only polynomial size increase. Finally, replace π_1 in π' with π_2; the result is a G_0^*-proof of S. □

Theorem 5.28 is interesting because we later show that a similar p-simulation for G_1 and G_1^* implies $C(\mathbf{PLS}) = \mathbf{FP}$ (see Theorem 6.2), which is not believed to be true. It is open whether Theorem 5.28 can be generalized to a p-simulation of G_0 by G_0^* for proving all prenex formulas. In fact, it is open whether G_0^* can p-simulate G_0 for Σ_2^q-formulas. Since we are not able to prove Theorem 5.26 for G_0, the above argument does not work for a more general case.

5.5 Connections to Bounded Arithmetic

Krajíček and Pudlák in [KP90] define a translation of bounded formulas of first-order bounded arithmetic into families of QPC formulas. We do not give the details of the translation here, and we only state some of its important properties. For simplicity, let $A(a)$ be a bounded formula with one free variable a. Then, for each $n \in \mathbb{N}$, Krajíček and Pudlák define a QPC formula $||A(a)||_n$ satisfying the following: (i) the size of $||A(a)||_n$ is polynomial in n; (ii) the free variable a of $A(a)$ is represented as an n-bit binary string by a vector of p-variables of length n; (iii) each bound variable of $A(a)$ is similarly represented by a vector of x-variables of polynomial length; and (iv) for $i \geq 1$, if $A(a) \in \Sigma_i^b$, then $||A(a)||_n \in \Sigma_i^q$; and (v)

$$||A(a)||_n \text{ is valid} \quad \text{if and only if} \quad \underline{\mathbb{N}} \models (\forall a)[|a| \leq n \supset A(a)].$$

Based on the above translation, Krajíček and Pudlák showed a close connection between T_2^i and G_i [KP90], and subsequently Krajíček demonstrated a similar connection between S_2^i and G_i^* [Kra95].

Theorem 5.29. *([KP90, Kra95]) For $i \geq 1$, if A is a Σ_i^b theorem of T_2^i, then the corresponding family of valid Σ_i^q-formulas $||A||_n$ has polynomial-size G_i-proofs which can be constructed in time polynomial in n. Similarly for S_2^i and G_i^*.*

Proof. (Idea) Let $i \geq 1$. If A is a Σ_i^b-proof of S_2^i, then there is an LK+LIND-proof π of A from the axioms of S_2^i such that all cut formulas are Σ_i^b; see Theorem 7.21. Given n, construct a G_i^*-proof π' of $||A||_n$ by first translating every sequent in π into a QPC sequent, and simulate each inference step in π by multiple steps in G_i^*. All steps except the length induction step is easy. The length induction step is simulated in the way described in Section 5.2. Note that the simulation of $\exists$-right and $\exists$-left requires a cut on the translation of the auxiliary formula. This is ok, since in π every $\exists$-right and $\forall$-left step has a $\Sigma_i^b \cup \Pi_i^b$-formula as its auxiliary formula.

Similarly for Σ_i^b-theorems of T_2^i and G_i. □

The above result is tight with respect to the quantifier complexity of A, since its proof does

not work for bounded theorems of T_2^i or S_2^i that are not Σ_i^q. More specifically, if A is not Σ_i^b, then the LK+LIND-proof of A contains $\exists$-right or $\forall$-left steps whose auxiliary formulas are not $\Sigma_i^b \cup \Pi_i^b$. Since the above proof requires a cut on such a formula, this step cannot be simulated in G_i^*. This is a direct consequence of the properties of the translation method, and we do not see how, using the same translation method, the above translation theorem can be generalized to a translation of *any* bounded theorem of T_2^i and S_2^i. Nonetheless, using the second-order translations described in Section 8.1, we can show that any bounded theorem of T_2^i (or S_2^i) can be translated into the corresponding valid QPC formulas with polysize proofs.

Theorem 5.30. *Theorem 5.29 continues to hold when A is a Σ_j^b theorem of T_2^i (respectively S_2^i) for any $j \geq 1$.*

Proof. S_2^i and T_2^i are RSUV-isomorphic to the second-order theories $\mathbf{V}^i$ and $\mathbf{TV}^i$, respectively (Theorems 7.8 and 7.10). Hence, the claim follows from the QPC translation theorem (Theorem 8.3) for $\mathbf{V}^i$ and $\mathbf{TV}^i$. □

Definition 5.31. *Let H be a QPC proof system. For $i \geq 0$, the* i-reflection principle for H *is essentially the soundness of H with respect to Σ_i^q, and it is represented by the following formula:*

$$i\text{-}RFN(H) \quad \equiv \quad \forall\vec{v}\forall\pi\forall A(\vec{p})[(A(\vec{p}) \in \Sigma_i^q \wedge \pi : G_i^* \vdash A(\vec{p})) \supset A(\vec{v}) \text{ is valid}] \tag{5.4}$$

where $\vec{v}, \pi, A(\vec{p})$ are actually numbers that encode the corresponding truth assignment, proof, and formula, respectively. The validity of $A(\vec{v})$ is represented by a Σ_i^b-formula which asserts the existence of a witness to the outermost existential quantifiers of $A(\vec{v})$. Thus the i-reflection principle is a $\forall\Sigma_i^b$-sentence.

Krajíček and Pudlák defined the above reflection principles, and they obtained results on their provability in bounded arithmetic:

Theorem 5.32. *([KP90, Kra95]) For every $i \geq 1$, T_2^i proves i-$RFN(G_i)$, and S_2^i proves i-$RFN(G_i^*)$.*

Furthermore, Krajíček and Pudlák prove that G_i^* and G_i are the strongest QPC proof systems for which S_2^i and T_2^i can prove the i-refection principle, respectively.

Theorem 5.33. *([KP90, Kra95]) Let i, j be such that $1 \leq j \leq i$, and let H be a QPC proof system. (i) If T_2^i proves j-$RFN(H)$, then G_i p-simulates H w.r.t. Σ_j^q. (ii) If S_2^i proves j-$RFN(H)$, then G_i^* p-simulates H w.r.t. Σ_j^q.*

From Theorems 5.32 and 5.33 together with Theorem 2.20, we obtain the following:

Corollary 5.34. *For every $i \geq 1$, G_i and G_{i+1}^* are p-equivalent w.r.t. Σ_i^q.*

Proof. Let $i \geq 1$. T_2^i proves i-$RFN(G_{i+1}^*)$ since it is a Σ_i^b-theorem of S_2^{i+1}, and S_2^{i+1} is Σ_{i+1}^b-conservative over T_2^i (Theorem 2.20). Thus, G_i p-simulates G_{i+1}^* w.r.t. Σ_i^q by Theorem 5.33.

Similarly, a p-simulation of G_i by G_{i+1}^* w.r.t. Σ_i^q follows from Theorem 5.33 and the fact that S_2^{i+1} proves i-$RFN(G_i)$. □

It is open whether the above p-simulation of G_{i+1}^* by G_i can be strengthened for Σ_{i+1}^q-formulas.

Chapter 6

The Witnessing Problems for QPC

In this Chapter, we introduce a new kind of total search problem, the QPC witnessing problems. The QPC witnessing problems provide a tangible link between the complexity of search problems and the length of proofs in QPC.

Definition 6.1. *Let $i \geq 0$ and let H be either G_i or G_i^*. For $j \geq 1$, define* the Σ_j^q-witnessing problem for H, *written $Witness[H, \Sigma_j^q]$, as follows. The input is $(\pi, \vec{v})$, where π is an H-proof of a Σ_j^q-formula $A(\vec{p})$ of the form*

$$A(\vec{p}) =_{syn} \exists x_1 \ldots \exists x_k F(\vec{p}, x_1, \ldots, x_k) \tag{6.1}$$

with F prenex Π_{j-1}^q, and $\vec{v}$ is a truth assignment to the free variables $\vec{p}$. A solution for the problem is a witness for $A(\vec{v})$, i.e., a sequence $\vec{u}$ of T, F such that $F(\vec{v}, \vec{u})$ holds.

If the input is not of the correct form (e.g., π is not an H-proof), then we define every string to be a solution for this input.

Let us describe how the QPC witnessing problem captures the relationship between the complexity of search problems and the proof lengths in QPC. Consider the following total search problem W, which is similar to the Σ_1^q-witnessing problem except that no proof is provided as part of the input. W is defined as follows. The inputs to W are of the form $(A, \vec{v})$, where A is a prenex Σ_1^q-formula of the form (6.1) with F quantifier-free, and $\vec{v}$ is a truth

assignment to the free variables of A. If $A(\vec{v})$ is a true sentence, then every witness $\vec{u}$ to $A(\vec{v})$'s existential quantifiers is a solution; otherwise, every string is a solution.

It is easy to see that W is in $\mathbf{FP}^{\mathbf{NP}}$ and hard for $\mathbf{NP}$. Moreover, every search problem defined by an $\mathbf{NP}$-relation is reducible to W. Finally, for every QPC proof system H, $Witness[H, \Sigma_1^q]$ reduces to W.

Let $H \in \{G_0^*, G_0, G_1^*, G_1\}$ and compare the complexity of $Witness[H, \Sigma_1^q]$ with that of W. This comparison shows how much the presence of an H-proof of a Σ_1^q-formula A reduces the complexity of witnessing A. The result is the following:

- $Witness[G_0^*, \Sigma_1^q]$ and $Witness[G_0, \Sigma_1^q]$ are complete for $\mathbf{FNC}^1$. (Theorem 6.5)
- $Witness[G_1^*, \Sigma_1^q]$ is complete for $\mathbf{FP}$. (Theorem 6.2)
- $Witness[G_1, \Sigma_1^q]$ is complete for $C(\mathbf{PLS})$. (Theorem 6.2)
- W is in $\mathbf{FP}^{\mathbf{NP}}$ and hard for $\mathbf{NP}$

Thus, when $H \in \{G_0^*, G_0, G_1^*, G_1\}$, the presence of an H-proof in the input dramatically decrease the complexity of witnessing the given formula A, and the amount of the decrease is larger if H is a weaker proof system. This shows that H-proofs of Σ_1^q-formulas contain information on how to construct a witness for the endformula A, and the weaker the system the more constructive the proofs.

Moreover, from the hardness directions of the above itemized statements, we can conclude that every search problem in $\mathbf{FNC}^1$ has a short G_0^*-proof of its totality, and similarly for $\mathbf{FP}$ and G_1^*, and $C(\mathbf{PLS})$ and G_1, respectively. This immediately tell us that G_0 does not have short $\mathbf{NC}^1$-uniform proofs of the formula $\exists \vec{w} \phi_C(\vec{p}, \vec{w})$ in (5.2) of page 92 unless $\mathbf{P} = \mathbf{NC}^1$.

Thanks to Krajíček and Pudlák's work on the close connection between QPC and bounded arithmetic [KP90], it is often possible to obtain complexity bounds on the QPC witnessing problems from results in bounded arithmetic. However, looking for direct proofs of the complexity bounds without going through bounded arithmetic is important for two reasons. First,

QPC proof complexity is an interesting research are of computer science on its own right, and in particular it is relevant to those who are working on the design of efficient algorithms for QPC provers. Proofs using bounded arithmetic are elegant, but they look unnecessarily complicated to those who do not work with it on regular basis. Second, we will show in Chapter 8 that the complexity of the Σ^q_k-witnessing problems for G^*_i and G_i do *not* match the definability results for T^i_2 and S^i_2, respectively, for $k \geq i+2$, and therefore direct proofs are the only methods for obtaining bounds for these witnessing problems.

We take a mixed approach and some results are obtained using bounded arithmetic, and some are obtained by direct methods. In particular, the hardness of various QPC witnessing problems are most easily obtained via the QPC translation theorems of bounded arithmetic (see Chapter 8).

Finally, since our results on the QPC witnessing problems are scattered through this Section and Chapter 8, we will summarize all the results in Chapter 9, where a table of all the results is provided (Table 9.2).

This Chapter is organized as follows. In Section 6.1, we obtain the completeness results for both $Witness[G^*_i, \Sigma^q_i]$ and $Witness[G_i, \Sigma^q_i]$ for every $i \geq 1$. While we prove these result by going through bounded arithmetic, they can be shown by direct proofs. In Section 6.2, we show that both $Witness[G^*_0, \Sigma^q_1]$ and $Witness[G_0, \Sigma^q_1]$ are complete for $\mathbf{FNC}^1$ under many-one $\mathbf{AC^0}$-reduction. Our proof uses the Herbrand π-disjunction, which we introduced in Section 5.4.2. In Section 6.3, we prove the completeness result for $Witness[G^*_i, \Sigma^q_{i+1}]$ for every $i \geq 1$. We prove it by a direct method, since there is no bounded arithmetic result that implies this. Note that the hardness direction is proven using the results in Chapter 8.

6.1 The Complexity of $Witness[G_i, \Sigma_i^q]$ and $Witness[G_i^*, \Sigma_i^q]$ for $i \geq 1$

The complexity of Σ_i^q-witnessing problems for G_i and G_i^* for $i \geq 1$ follow from the existing results in bounded arithmetic.

Theorem 6.2. *Let $i \geq 1$. (i) $Witness[G_i^*, \Sigma_i^q]$ is complete for $\mathbf{FP}^{\mathbf{\Sigma}_{i-1}^p}$; and (ii) $Witness[G_i, \Sigma_i^q]$ is complete for $C(\mathbf{PLS})^{\mathbf{\Sigma}_{i-1}^p}$. The completeness is with respect to polynomial-time many-one reducibility.*

Proof. We first prove that $Witness[G_i^*, \Sigma_i^q]$ is in $\mathbf{FP}^{\Sigma_{i-1}^q}$. Recall the definition of $i\text{-}RFN(G_i^*)$ in (5.4) and note that $Witness[G_i^*, \Sigma_i^q]$ is the search problem whose defining relation is represented by $i\text{-}RFN(G_i^*)$. It follows that $Witness[G_i^*, \Sigma_i^q]$ is a $\hat{\Sigma}_i^b$-definable search problem of S_2^i, and, by Buss's witnessing theorem in [Bus86] (Theorem 2.22 of this dissertation), it is in $\mathbf{FP}^{\Sigma_{i-1}^p}$.

That $Witness[G_i, \Sigma_i^q] \in C(\mathbf{PLS})^{\Sigma_{i-1}^p}$ follows in an analogous manner from the provability of $i\text{-}RFN(G_i)$ in T_2^i (Theorem 5.32) and the $\hat{\Sigma}_i^b$-witnessing theorem for T_2^i (Item (ii) of Theorem 2.22).

The hardness of $Witness[G_i^*, \Sigma_i^q]$ is shown as follows. Let Q be a search problem in $\mathbf{FP}^{\Sigma_{i-1}^p}$. By Theorem 2.22, Q is is $\hat{\Sigma}_i^b$-definable in S_2^i; that is, there exists $\phi(x, y) \in \hat{\Sigma}_i^b$ such that the following two conditions hold: (i) $\phi(x, y)$ represents a relation R' such that, for every $x, y \in \mathbb{N}$, $R'(x, y)$ implies $y \in Q(x)$; and (ii) S_2^i proves $\forall x \exists y \phi(x, y)$. Let $A(a)$ denote $\exists y \phi(a, y)$ with one free variable a. By Theorem 5.29, given $a \in \mathbb{N}$ with $|a| = n$, we can construct in polynomial-time a G_i^*-proof π of the Σ_i^q-formula $||A(a)||_n$. Let A^q denote $||A(a)||_n$. Since A^q is not prenex even when A is prenex, we do the following. Let $A^+ \in \Sigma_i^q$ be a prenex form of A^q such that

$$A^q \to A^+$$

has short, cut-free G_0^*-proofs. By a cut on A^q we construct from π another G_i^*-proof π' with

endformula A^+. Finally, by letting $\vec{v}$ the truth assignment encoding the bits of a, a solution for $Witness[G_i^*, \Sigma_i^q]$ on the instance $(\pi', \vec{v})$ gives a solution for $Q(a)$. Thus, Q is many-one reducible to $Witness[G_i^*, \Sigma_i^q]$.

The hardness of $Witness[G_i, \Sigma_i^q]$ for $C(\mathbf{PLS})^{\Sigma_{i-1}^p}$ is proven in a way completely analogous to the G_i^* case above, using Theorems 2.22 and 5.29. □

It follows from Theorem 6.2 that, if G_1^* p-simulates G_1 w.r.t. Σ_1^q, then $\mathbf{FP} \supseteq C(\mathbf{PLS})$, which is believed to be false. More generally, a p-equivalence between any two quantified sequent calculi discussed in this paper implies a collapse of the corresponding complexity classes.

Cook [Coo02] has a direct proof of $Witness[G_i^*, \Sigma_i^q] \in \mathbf{FP}^{\mathbf{\Sigma}_{i-1}^p}$. Also Alan Skelley in 2002 described to us in person a direct proof of $Witness[G_i, \Sigma_i^q] \in C(\mathbf{PLS})^{\mathbf{\Sigma}_{i-1}^p}$. Skelley's direct proof has not appeared in print, and Buss in [Bus04], independently of Skelley's work, applied similar ideas with respect to bounded-depth Frege proofs to obtain alternative proofs of Theorem 2.22. An advantage of the proofs by Cook and Skelley is that their proofs do not mention bounded arithmetic at all and therefore are more accessible to a general computer science audience.

6.2 $\mathbf{FNC}^1$ and the Σ_1^q-Witnessing Problems for G_0^* and G_0

Since G_0 and G_0^* are new proof systems for QPC, the complexity of their witnessing problems have never been studied. In this Section, we prove that the Σ_1^q-Witnessing Problems for G_0^* and G_0 are both complete for $\mathbf{FNC}^1$ under many-one $\mathbf{AC^0}$-reduction. We begin with the hardness direction.

The Hardness for $\mathbf{FNC}^1$

A search problem Q is said to be *hard for* $\mathbf{FNC}^1$ *under many-one* $\mathbf{AC^0}$*-reductions* iff every search problem in $\mathbf{FNC}^1$ is many-one $\mathbf{AC^0}$-reducible to Q. Q is *complete for* $\mathbf{FNC}^1$ if $Q \in$

FNC^1.

First, we describe how a weaker claim of the hardness of $Witness[G_0^*, \Sigma_1^q]$ for **NC**^{1} under many-one $\mathbf{AC^0}$-reductions can be proven. For every propositional sentence A, the sequent

$$\rightarrow (\exists x)[(x \wedge A) \vee (\neg x \wedge \neg A)]$$

has a G_0^*-proof π consisting of k sequents, where k is some constant that does not depend on A; in fact, we have seen such a proof in Chapter 5 as Figure 5.1 on page 90. It can be shown that the G_0^*-proof π of Figure 5.1 is constructible from A by an $\mathbf{AC^0}$-function. Finally, note that the witness to x is the truth value of A and that the problem of evaluating propositional sentences is hard for **NC**^{1} [BIS90].

The fact that $Witness[G_0^*, \Sigma_1^q]$ is hard for **NC**^{1} does not immediately imply its hardness for **FNC**^{1}; however, we can use essentially the same idea to prove the **FNC**^{1} hardness.

Theorem 6.3. *Both $Witness[G_0^*, \Sigma_1^q]$ and $Witness[G_0, \Sigma_1^q]$ are hard for* **FNC**^{1} *under many-one* $\mathbf{AC^0}$*-reductions.*

Proof. Let f be an arbitrary **NC**^{1}-function. It suffices to show that f is many-one $\mathbf{AC^0}$-reducible to $Witness[G_0^*, \Sigma_1^q]$. Assume without loss of generality that there is a polynomial p such that $f : \{0,1\}^n \rightarrow \{0,1\}^{p(n)}$ for every n. Since the bit graph $R_f(x,i,c)$ is in **NC**^{1}, and since every **NC**^{1}-predicate is computed by a **Dlogtime**-uniform family of polynomial-size propositional formulas [BIS90], there exists a **Dlogtime**-uniform family $\{A_n\}_n$ of polynomial-size propositional formulas such that, for each $n = |x|$, $A_n(x,i)$ is true if the ith bit of $f(x)$ is 1 and $A_n(x,i)$ is false otherwise, where x and i are represented in A_n by sequences $\vec{p}$ and $\vec{q}$ of propositional variables, respectively.

Let $m = p(n)$ and, for each $i \in [1,m]$, let $\vec{i}$ denote the truth assignment to $\vec{q}$ representing i in unary. Define sequent S_n as follows:

$$\rightarrow (\exists y_1) \ldots (\exists y_m)[(y_1 \leftrightarrow A_n(\vec{p}, \vec{1})) \wedge \ldots \wedge (y_m \leftrightarrow A_n(\vec{p}, \vec{m}))]$$

where '$y_i \leftrightarrow A_n(\vec{p}, \vec{i})$' abbreviates $(y_i \wedge A_n(\vec{p}, \vec{i})) \vee (\neg y_i \wedge \neg A_n(\vec{p}, \vec{i}))$. Suppose that π_n is a G_0^*-proof of S_n and that $\vec{v}$ is a truth assignment to $\vec{p}$ encoding $x \in \{0,1\}^n$. It is easy to

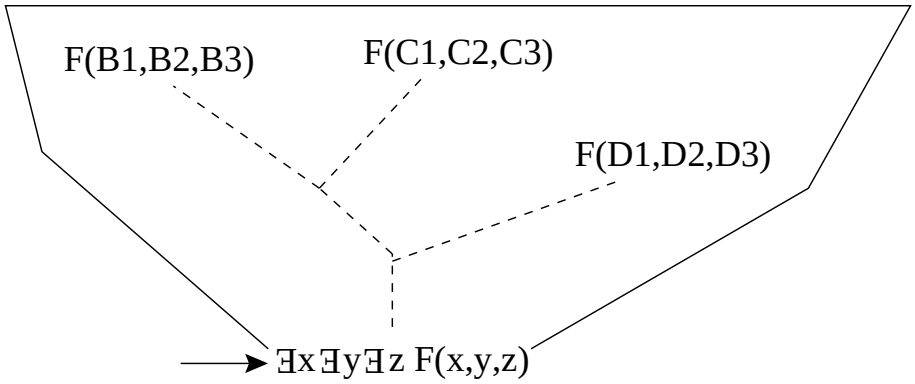

Figure 6.1: A G_0^*-proof of a prenex Σ_1^q-formula $\exists x \exists y \exists z F(x, y, z)$.

see that there is an $\mathbf{AC^0}$ function that computes $f(x)$ given the solution for $Witness[G_0^*, \Sigma_1^q]$ on $(\pi_n, \vec{v})$. Thus, it suffices to show the existence of an $\mathbf{AC^0}$-function g such that $g(x)$ is a G_0^*-proof $\pi_{|x|}$ of $S_{|x|}$ of size polynomial in $|x|$.

Below we give an informal description of the proof π_n. The sequent S_n is derived by m applications of $\exists$-right from

$$\rightarrow (A_n(\vec{p}, \vec{1}) \leftrightarrow A_n(\vec{p}, \vec{1})) \wedge \ldots \wedge (A_n(\vec{p}, \vec{m}) \leftrightarrow A_n(\vec{p}, \vec{m}))$$

which follows by applications of $\wedge$-right from the sequents

$$\rightarrow (A_n(\vec{p}, \vec{i}) \leftrightarrow A_n(\vec{p}, \vec{i})) \tag{6.2}$$

for each $i \in [1, m]$. It is easy to see that every sequent of the form (6.2) has a G_0^*-proof with a constant number of sequents, and this completes the description of π_n. Finally, an $\mathbf{AC^0}$-function can output π_n because each line of π_n has a highly uniform structure and it is easy to determine what the jth sequent of π_n should look like for any j. □

The Membership in FNC^1

Here we prove that there exists an $\mathbf{NC}^1$-function that outputs a solution for $Witness[G_0, \Sigma_1^q]$. We first describe an informal idea why this is the case.

As an example, let $(\pi, \vec{v})$ be an instance of $Witness[G_0, \Sigma_1^q]$, where π is depicted as Figure 6.1, which is a reproduction of Figure 5.2. Assume without loss of generality that π is in free variable normal form; thus, every free variable of π appears in the endformula. Note that π can be converted into free variable normal form by a $\mathbf{TC}^0$-function.

Let A be the endformula $\exists x \exists y \exists z F(x, y, z)$ of π. Recall the definition of π-prototypes and Herbrand π-disjunctions (Definition 5.22 on page 109). A has three π-prototypes $A(B1, B2, B3)$, $A(C1, C2, C3)$, and $A(D1, D2, D3)$, and, by Lemma 5.25, the Herbrand π-disjunction

$$\rightarrow A(B1, B2, B3), A(C1, C2, C3), A(D1, D2, D3)$$

is valid. Hence, under the truth assignment $\vec{v}$, at least one of the three π-prototypes is true. Suppose that $A(C1, C2, C3)$ is true under $\vec{v}$. Then

$$A(eval(C1, \vec{v}), eval(C2, \vec{v}), eval(C3, \vec{v}))$$

is a true sentence, where $eval(\phi, \vec{v})$ denotes the truth value of a formula ϕ under $\vec{v}$. Thus, a solution for the instance $(\pi, \vec{v})$ is obtained by evaluating the components $C1$, $C2$, and $C3$ under $\vec{v}$. (Components of a π-prototype is defined in Definition 5.23.)

In order to show that $Witness[G_0, \Sigma_1^q] \in \mathbf{FNC}^1$, it suffices to show how to compute the ith bit u_i of the witness. Here is an algorithm for it. Given $(\pi, \vec{v})$, find the first π-prototype A' that is true under $\vec{v}$; we will say more about the ordering on π-prototypes below. Then u_i is obtained by evaluating the ith component of A' under $\vec{v}$. Buss has shown in [Bus87] that the Boolean Formula Value Problem of evaluating the input propositional formula under the given truth assignment is in $\mathbf{NC}^1$, and this fact is crucial in our algorithm.

In the above algorithm, π-prototypes are ordered simply by the order of their appearance in π. We need to recognize the *first* π-prototype that is true under $\vec{v}$ since, if there are more than one π-prototype that is true under $\vec{v}$, the bits of a witness $u_1, \ldots, u_k$ must be extracted from the same π-prototype; otherwise the result $\vec{u}$ may not be a witness at all.

Below we argue more formally that $Witness[G_0, \Sigma_1^q] \in \mathbf{FNC}^1$. Let π be the input G_0-proof

of $A(\vec{p})$ of the form

$$A(\vec{p}) =_{syn} \exists x_1 \ldots \exists x_k F(\vec{p}, x_1, \ldots, x_k) \tag{6.3}$$

with F quantifier-free, and let $A_1, \ldots, A_m$ be a sequence of all π-prototypes (Definition 5.22). Note that the sequent $\rightarrow A_1, \ldots, A_m$ is valid by Lemma 5.25. For each $j \in [1, m]$, define a formula E_j which states that A_j is the first in the sequence $A_1, \ldots, A_m$ that is satisfied, i.e.,

$$E_j =_{syn} (\neg A_1 \wedge \neg A_2 \wedge \ldots \wedge \neg A_{j-1}) \wedge A_j.$$

Thus, any truth assignment $\vec{v}$ satisfies E_l for exactly one value $l \in [1, m]$.

Definition 6.4. *Let $A(\vec{p})$ and k be as in (6.3) and $A_1, \ldots, A_m$ and $E_1, \ldots, E_m$ be as in the above paragraph. For all $i \in [1, k]$, define*

$$\phi_i =_{syn} \bigvee_{j=1}^{m} (E_j \wedge B_i^j),$$

where B_i^j is the ith component of A_j. We call ϕ_i the ith π-witness formula.

It is easy to see that a witness $\vec{u}$ to $A(\vec{p})$ is computed by evaluating $\phi_1, \ldots, \phi_k$ under $\vec{v}$. We show that this fact has short PK-proofs in Theorem 6.7.

Theorem 6.5. *Both $Witness[G_0, \Sigma_1^q]$ and $Witness[G_0^*, \Sigma_1^q]$ are in* $\mathbf{FNC}^1$.

Proof. It suffices to show that the following relation is in $\mathbf{NC}^1$:

$$R(\pi, \vec{v}, i) \quad \equiv \quad \text{the } i\text{th } \pi\text{-witness formula } \phi_i \text{ of } \pi \text{ evaluates to } \mathsf{T} \text{ under } \vec{v},$$

where π is either G_0-proof in sequential representation or a G_0^*-proof in bracket representation. It is easy to see that there is a $\mathbf{TC}^0$-function F such that $F(\pi, i)$ is the formula ϕ_i. By Buss's result in [Bus87, Bus93], there is an $\mathbf{NC}^1$ relation B such that $B(\psi, \vec{u})$ holds iff the propositional formula ψ evaluates to T under the truth assignment $\vec{u}$. Then

$$R(\pi, \vec{v}, i) \equiv B(F(\pi, i), \vec{v}).$$

Since $B(F(\pi, i), \vec{v})$ is in $\mathbf{NC}^1$, this completes the proof. □

From Theorems 6.3 and 6.5, we obtain the desired completeness result.

Theorem 6.6. *Both $Witness[G_0^*, \Sigma_1^q]$ and $Witness[G_0, \Sigma_1^q]$ are complete for* $\mathbf{FNC}^1$ *under many-one* $\mathbf{AC^0}$*-reduction.*

The Correctness of the Witness Formulas

We prove that the π-witness formulas compute a solution for the Σ_1^q-witnessing problem for G_0 and that this fact has short PK proofs. This fact may be useful in proving that $\mathbf{VNC}^1$ proves 1-$RFN(G_0)$, which we expect to be true.

Theorem 6.7. *Let π be a G_0-proof of a prenex Σ_1^q-formula A of the form $\exists x_1 \ldots \exists x_k F(\vec{p}, x_1, \ldots, x_k)$, where F is quantifier-free. Let $\phi_1, \ldots, \phi_k$ be the π-witness formulas. Then $F(\vec{p}, \phi_1, \ldots, \phi_k)$ is valid, and there is a* $\mathbf{TC}^0$*-function that, given π, outputs a PK-proof of $F(\vec{p}, \phi_1, \ldots, \phi_k)$.*

Proof. We write F to denote $F(\vec{p}, x_1, \ldots, x_k)$. $F[\vec{x}/\vec{\phi}]$ denotes the result of substituting ϕ_i for x_i, $i \in [1, k]$. Our first goal is to show that $F[\vec{x}/\vec{\phi}]$ has PK-proofs of size polynomial in $|\pi|$. That these PK-proofs can be constructed by a $\mathbf{TC}^0$-function will be clear.

Let $A_1, \ldots, A_m$ be the π-prototypes of A. For each $j \in [1, m]$, define sequents S_j and T_j as

$$S_j =_{syn} \quad \rightarrow F[\vec{x}/\vec{\phi}], A_1, \ldots, A_j$$
$$T_j =_{syn} A_j \rightarrow F[\vec{x}/\vec{\phi}], A_1, \ldots, A_{j-1}$$

S_m is derived by weakening the Herbrand disjunction $\rightarrow A_1, \ldots, A_m$ which, by Lemma 5.25, has a PK-proof of size polynomial in $|\pi|$. For each $j \in [1, m-1]$, S_j is derived from S_{j+1} and T_{j+1} by cut, and finally $F[\vec{x}/\vec{\phi}]$ is derived from S_1 and T_1 by cut. Thus it suffices to prove that, for each $j \in [1, m]$, the sequent T_j has a polysize PK-proof.

Fix $j \in [1, m]$. Let C be a subformula of F, and let $C[\vec{x}/\vec{B^j}]$ be the result of substituting, for every $i \in [1, k]$, B_i^j for x_i in C. Note that $C[\vec{x}/\vec{B^j}]$ is a subformula of A_j. Define two

sequents U_1^C and U_2^C as follows:

$$U_1^C : \quad C[\vec{x}/\vec{B^j}], A_j \to C[\vec{x}/\vec{\phi}], A_1, \ldots, A_{j-1}$$
$$U_2^C : \quad C[\vec{x}/\vec{\phi}], A_j \to C[\vec{x}/\vec{B^j}], A_1, \ldots, A_{j-1}$$

It is clear that T_j follows from U_1^C for $C =_{syn} F$ by contraction. We prove that, for any subformula C of F, both U_1^C and U_2^C have PK-proofs of size polynomial in $|\pi|$.

We proceed by structural induction on C. If C does not contain any occurrence of an x-variable, then $C[\vec{x}/\vec{\phi}]$ and $C[\vec{x}/\vec{B^j}]$ are identical, and therefore both U_1^C and U_2^C follow from initial sequents by weakening.

For the other base case, suppose that C is an atom (x_i) for $i \in [1, k]$. Then $C[\vec{x}/\vec{\phi}]$ is ϕ_i and $C[\vec{x}/\vec{B^j}]$ is B_i^j, and hence we need to show that the following two sequents have polysize PK-proofs:

$$U_1^B : \quad B_i^j, A_j \to \bigvee_{l=1}^{m} (E_l \wedge B_i^l), A_1, \ldots, A_{j-1} \tag{6.4}$$
$$U_2^B : \quad \bigvee_{l=1}^{m} (E_l \wedge B_i^l), A_j, \to B_i^j, A_1, \ldots, A_{j-1} \tag{6.5}$$

The sequent (6.4) is derived by weakening and $\vee$-right from

$$B_i^j, A_j \to (E_j \wedge B_i^j), A_1, \ldots, A_{j-1}$$

which follows from the two sequents $B_i^j \to B_i^j$ and $A_j \to E_j, A_1, \ldots, A_{j-1}$. By the definition of E_j, the latter sequent has short proofs.

The sequent (6.5) is derived by m applications of $\vee$-left from the sequents

$$(E_l \wedge B_i^l), A_j, \to B_i^j, A_1, \ldots, A_{j-1} \tag{6.6}$$

for each $l \in [1, m]$. We claim that all of the sequent of the form (6.6) have short PK proofs. If $l < j$, then the sequent contains A_l in both sides of '$\to$'. If $l = j$, then B_i^j appears in both sides. Finally, if $l > j$, then the antecedent of the sequent contains both A_j and $\neg A_j$. This concludes the case where $B =_{syn} (x_i)$ for some $i \in [1, k]$.

The inductive step is straightforward. If C is $(C_1 \vee C_2)$, $(C_1 \wedge C_2)$, or $(\neg C_1)$, then the sequents U_1^C and U_2^C have short PK-proofs from $U_1^{C_1}$, $U_2^{C_1}$, $U_1^{C_2}$, and $U_2^{C_2}$, all of which have short PK-proofs.

Finally, we claim that the above construction can be carried out by a $\mathbf{TC}^0$-function. First, note that PK-proof of S_m is $\mathbf{TC}^0$-constructible from π by Lemma 5.25. For $j \in [1, m]$, the PK-proofs of T_j that we described above is easily seen to be $\mathbf{TC}^0$-constructible, and therefore the claim holds. □

6.3 The Σ_k^q-Witnessing Problems with Large k

In this Section, we prove the following:

(i) For every $i \geq 1$, $Witness[G_i^*, \Sigma_{i+1}^q]$ is in $\mathbf{FP}^{\mathbf{\Sigma}_i^q}[wit, O(\log n)]$ (Theorem 6.9); and

(ii) For every $i \geq 0$ and $k \geq i+2$, $Witness[G_i, \Sigma_k^q]$ is in $\mathbf{FP}^{\mathbf{\Sigma}_{k-1}^q}[wit, O(\log n)]$ (Theorem 6.11).

For item (i) above, we prove a matching hardness result in Chapter 8 (Theorem 8.6) and therefore $Witness[G_i^*, \Sigma_{i+1}^q]$ is complete for $\mathbf{FP}^{\mathbf{\Sigma}_i^q}[wit, O(\log n)]$. Item (ii) implies an upper bound on $Witness[G_i^*, \Sigma_k^q]$ for every $i \geq 0$ and $k \geq i+2$; this is the best upper bound we have so far.

Note that Theorem 6.2 in Section 6.1 and Theorem 6.9 below (item (i) above) show that the complexity of Σ_j^q-witnessing problems of G_i^* correspond to the complexity of the $\hat{\Sigma}_k^b$-definable search problems of S_2^i for $k \in \{i, i+1\}$; see Theorems 2.22 and 2.23. If the correspondence continued for $k \geq i+2$, we would have $Witness[G_i^*, \Sigma_k^q] \in \mathbf{FP}^{\mathbf{\Sigma}_{k-1}^q}[wit, O(1)]$. However, we will prove that, if this is the case, then $\mathbf{PH}$ collapses (Theorem 8.8). Therefore, the upper bound using $O(\log n)$ witness queries in item (ii) above (Theorem 6.11) cannot be improved to $O(1)$ witness queries, assuming that $\mathbf{PH}$ does not collapse. On this other hand, we only know that $Witness[G_i^*, \Sigma_k^q]$ is hard for $\mathbf{FP}^{\mathbf{\Sigma}_{k-1}^q}[wit, O(1)]$ (Theorem 8.6). Closing the gap between the hardness and the upper bound is an open problem.

We use the following notion of strong-Σ_i^q in the proofs below:

Definition 6.8. *Let $i \geq 1$ be arbitrary. We say that a QPC formula A is* strong-Σ_i^q *if $A \in \Sigma_i^q$ and $A \notin \Pi_i^q$. Similarly, A is* strong-Π_i^q *if $A \in \Pi_i^q$ and $A \notin \Sigma_i^q$.*

6.3.1 The Complexity of $Witness[G_i^*, \Sigma_{i+1}^q]$

First, we prove that $Witness[G_i^*, \Sigma_{i+1}^q]$ is in $\mathbf{FP}^{\mathbf{\Sigma}_i^p}[wit, O(\log n)]$, and in fact it is complete for the class. Recall that, by Theorem 2.7,

$$\mathbf{FP}^{\mathbf{\Sigma}_i^p}[wit, O(\log n)] = \mathbf{FP}_{||}^{\mathbf{\Sigma}_i^p}[wit]$$

for every $i \geq 1$.

Theorem 6.9. *For every $i \geq 1$, $Witness[G_i^*, \Sigma_{i+1}^q]$ is complete for $\mathbf{FP}^{\mathbf{\Sigma}_i^p}[wit, O(\log n)]$ under polynomial-time many-one reduction.*

Proof. The hardness is proven in Theorem 8.6. The membership would easily follow if S_2^i proves $(i+1)$-$RFN(G_i^*)$, but we do not know this. We give a direct proof of the membership, which is actually a first positive step toward deciding whether S_2^i proves $(i+1)$-$RFN(G_i^*)$.

Let $(\pi, \vec{v})$ be an instance of the witnessing problem. By Theorem 5.13, π can be converted in polynomial-time into another G_i^*-proof in which all cut formulas are prenex Σ_i^q. Thus, we assume without loss of generality that all cut formulas in π are prenex Σ_i^q. This and the fact that the endformula of π is prenex implies the following: (i) every formula in π is prenex Σ_{i+1}^q; (ii) every formula that occurs in the antecedent of a sequent in π is Σ_i^q; and (iii) every strong-Π_i^q-formula in π is an ancestor of the endformula, and it occurs in the succedent of a sequent. Note that we do not know whether Theorem 5.13 holds for G_i, i.e., for dag proofs. We will say more about this after this proof.

We also assume without loss of generality that π is in free variable normal form. Since π is tree-like, every formula in π is an ancestor of either a cut formula or the endformula but not both.

We describe an algorithm that solves the witnessing problem in polynomial-time by making polynomially many witness queries to an $\mathbf{\Sigma}_i^p$ oracle in a nonadaptive manner.

Let $\pi[\vec{v}]$ be the result of replacing the parameter variables of π with the truth values specified by $\vec{v}$. Then the free variables of $\pi[\vec{v}]$ are eigenvariables of $\forall$-right and $\exists$-left. Let S be a sequent of $\pi[\vec{v}]$. We represent S in the following way:

$$\Gamma \to \Delta_\forall, \Delta_{\exists\forall}$$

where Γ is the antecedent of S, and the succedent of S is partitioned into two cedents so that $\Delta_\forall$ contains all Π_i^q-formulas, and $\Delta_{\exists\forall}$ contains the rest of the formulas. Note that every formula $\Delta_{\exists\forall}$ is either strong-Σ_i^q or strong-Σ_{i+1}^q. Moreover, every strong-Σ_i^q-formula in $\Delta_{\exists\forall}$ is an ancestor of a cut formula, and every strong-Σ_{i+1}^q-formula is an ancestor of the endformula. Let $\vec{b}$ be all the free variables of S. We define $f(S)$ to be the following sequent:

$$f(S) =_{syn} \quad \Gamma \to \Delta_\forall$$

We define $q(S)$ to be the witness query asking whether the sequent $f(S)$ is falsified by some truth assignment to $\vec{b}$. A positive answer to this query returns a falsifying assignment. Note that, for every S, $f(S)$ is equivalent to a Π_i^q-formula and therefore the query $q(S)$ is a Σ_i^q-witness query.

Clearly, if the succedent of S consists of Π_i^q-formulas only, then $f(S)$ is valid and the answer to $q(S)$ is negative. More specifically, for every intitial sequent S, $f(S)$ is valid. On the other hand, if S is the endsequent, then the query $q(S)$ is trivial since $f(S)$ is the empty sequent, i.e., unsatisfiable. Thus, in every path of $\pi[\vec{v}]$, there must be an inference step with an upper sequent S_1 and the lower sequent S_2 such that $f(S_1)$ is valid but $f(S_2)$ is not. In fact, it is not hard to see that $f(S_1)$ logically implies $f(S_2)$ for every inference rule except for $\exists$-right and cut. More specifically, if $f(S_1)$ does not logically imply $f(S_2)$, then this inference step must be either (i) an $\exists$-right step introducing a strong Σ_i^q-formula, or (ii) a cut on a strong-Σ_i^q-formula and S_1 is the upper-left sequent in

$$\frac{(S_1) \quad \exists x\phi, \Gamma \to \Delta \qquad \Gamma \to \Delta, \exists x\phi \qquad (S_3)}{(S_2) \qquad \Gamma \to \Delta}$$

Note that, in this case, $f(S_3)$ and $f(S_2)$ are logically equivalent. We will use these properties later.

Our algorithm for solving the witness query is simple. Let $S_1, \ldots, S_m$ be the input proof π. First, compose the m witness queries $q(S_1) \ldots, q(S_m)$, and ask all of them at once. After receiving the answers to the queries, find an $\exists$-right step

$$\frac{(T) \quad \Gamma \to \Delta, A(B)}{(T') \quad \Gamma \to \Delta, (\exists x)A(x)}$$

such that $f(T)$ is valid but $f(T')$ has a falsifying assignment. Note that such $f(T)$ and $f(T')$ must be of the form

$$f(T) = \quad \Gamma \to \Delta_\forall, A(B) \qquad \text{and} \qquad f(T') = \quad \Gamma \to \Delta_\forall$$

where $A(B)$ is a strong-Π_i^q-formula that is an ancestor of the endformula. Let $\vec{u}$ be a falsifying assignment given as the answer to $q(T')$. It follows that $A(B)$ evaluates to T under $\vec{u}$. Thus, the solution for $Witness[G_i^*, \Sigma_{i+1}^q]$ is found by evaluating under $\vec{u}$ the targets of $\exists$-right steps that introduce the outermost existential quantifiers of the endformula. Since the targets are propositional subformulas, this algorithm runs in polynomial-time using nonadaptive witness queries to $\mathbf{\Sigma}_i^p$. Thus the witnessing problem is in $\mathbf{FP}_{||}^{\mathbf{\Sigma}_i^p}[wit]$, which is equal to $\mathbf{FP}^{\mathbf{\Sigma}_i^p}[wit, O(\log n)]$ by (iv) of Theorem 2.7.

We still need to prove that the above algorithm is correct, i.e., that there exists an $\exists$-right step with upper sequent T and lower sequent T' such that $f(T)$ is valid and $f(T')$ is invalid. We prove this by traversing $\pi[\vec{v}]$ in the following way. Let P be the 'leftmost' path of $\pi[\vec{v}]$ from the endsequent to an initial sequent; that is, P is a path that contains no sequent that is an upper-right sequent of an inference step. We start traversing $\pi[\vec{v}]$ from the initial sequent of P. First, keep going toward the endsequent until we first encounter a sequent S' with $f(S')$ invalid. If S' is the lower sequent of an $\exists$-right step, we are done. Otherwise, S' is the lower sequent of a cut, and we were in the upper-left sequent of the cut. Note that $f(S') \equiv f(S'')$, where S'' is the upper-right sequent of the cut. Now we start traversing $\pi[\vec{v}]$ upward from S'', while maintaining the invalidity of $f(S)$, where S is the currently visited sequent. When we

encounter a cut, always go to the upper-right sequent. When we encounter binary inference steps (i.e., $\vee$-left or $\wedge$-right), take an arbitrary way up. Since $f(S)$ for every initial sequent S is valid, we must encounter a desired $\exists$-right step. $\square$

As we stated in the proof above, the provability of $(i+1)$-$RFN(G_i^*)$ in S_2^i will follow if we can demonstrate that S_2^i proves the correctness of the above witness-query algorithm.

In the above proof, we define $f(S)$ by removing from the succedent of S the strong-Σ_{i-1}^q-ancestors of the endformulas and the strong-Σ_i^q-ancestors of cut formulas. This definition ensures the following two properties: (i) $f(S)$ is equivalent to a Π_i^q-formula; and (ii) every inference rule except $\exists$-right and cut preserves the validity of $f(S)$ for an upper sequent S. Property (i) ensures that the complexity of witness queries is Σ_i^q, and property (ii) ensures that a witness can be found from $\exists$-steps that do not preserve the validity of $f(S)$. These properties hold because of the fact that every G_i^*-proof can be converted into another proof with cuts on prenex Σ_i^q-formulas only (Theorem 5.13). On the other hand, without the assumption that every cut formula is prenex Σ_i^q, a sequent of π may contain both Σ_i^q-formulas and Π_i^q-formulas in the antecedent. In this case, we do not know how to define $f(S)$ so that properties (i) and (ii) hold and our proof works.

Since we do not know whether Theorem 5.13 holds for G_i, the argument in the above proof of Theorem 6.9 fails to give us any meaningful upper bound on the complexity of $Witness[G_i, \Sigma_{i+1}^q]$. On the other hand, if Theorem 5.13 does hold for G_i as well as for G_i^*, then there is a complexity-theoretic consequence:

Theorem 6.10. *Let $i \geq 1$ and define $\hat{G}_i$ to be G_i with an additional restriction that every cut formula be prenex Σ_i^q. If $\hat{G}_i$ p-simulates G_i w.r.t. prenex Σ_{i+1}^q, then*

$$\mathbf{FP}^{\mathbf{\Sigma}_i^q}[wit, O(\log n)] = \mathbf{FP}^{\mathbf{\Sigma}_i^p}$$

Proof. Let $i \geq 1$ be arbitrary and assume that $\hat{G}_i$ p-simulates G_i w.r.t. Σ_{i+1}^Q. Then the proof of Theorem 6.9 goes through, and thus we have

$$Witness[G_i, \Sigma_{i+1}^q] \in \mathbf{FP}^{\mathbf{\Sigma}_i^q}[wit, O(\log n)].$$

Since $Witness[G_i, \Sigma^q_{i+1}]$ is hard for $\mathbf{FP}^{\mathbf{\Sigma}^p_i}$, it follows that

$$\mathbf{FP}^{\mathbf{\Sigma}^q_i}[wit, O(\log n)] = \mathbf{FP}^{\mathbf{\Sigma}^p_i}.$$

□

Not much is known about whether $\mathbf{FP}^{\mathbf{\Sigma}^q_i}[wit, O(\log n)]$ and $\mathbf{FP}^{\mathbf{\Sigma}^p_i}$ are equal. We only know that

$$\mathbf{FP}^{\mathbf{\Sigma}^q_i}[O(\log n)] = \mathbf{FP}^{\mathbf{\Sigma}^q_i}$$

for some $i \geq 1$ implies $\mathbf{PH} = \mathbf{P}^{\mathbf{\Sigma}^p_{i-1}}$. Krentel [Kre88] shows this for $i = 1$, and it is easy to generalize his proof for any $i \geq 1$. However, the proof of this does not work if witness queries are allowed.

We have not obtained an upper bound on the complexity of $Witness[G_i, \Sigma^q_{i+1}]$, and it seems likely that the upper bound is $\mathbf{FP}^{\mathbf{\Sigma}^p_i}$. There are at least two ways we could prove this: the first way is to show that T^i_2 proves $(i+1)$-$RFN(G_i)$; and the second way is to show the upper bound directly, as we did for Theorem 6.9.

6.3.2 An Upper Bound on the Complexity of $Witness[G_i, \Sigma^q_k]$ for $k \geq i+2$

Theorem 6.11. *For every $i \geq 0$ and $k \geq i+2$, $Witness[G_i, \Sigma^q_k]$ is in $\mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(\log n)]$.*

Proof. Let $i \geq 0$ and $k \geq i+2$ be arbitrary. The claim would follow if T^i_2 (or $\mathbf{TV}^i$ if $i = 0$) proves k-$RFN(G_i)$; however, we know that T^i_2 does *not* prove k-$RFN(G_i)$ unless $\mathbf{PH}$ collapses; see Corollary 8.9. Thus, we prove the claim by a direct proof similar to the proof of Theorem 6.9. Note, however, that this time our proof does not require that all cut formulas be prenex Σ^q_i, and therefore it works for G_i.

Let $(\pi, \vec{v})$ an instance of $Witness[G_i, \Sigma^q_k]$. We proceed as in the proof of Theorem 6.9 and let $\pi[\vec{v}]$ be the result of replacing the parameter variables of π with the truth values specified by $\vec{v}$.

Let S a sequent of $\pi[\vec{v}]$. Represent S as

$$\Gamma \to \Delta_{ns}, \Delta_s$$

where Δ_s is the cedent consisting of all strong-Σ^q_k-formulas in the succedent of S, and Δ_{ns} is the cedent containing the rest of the formulas in the succedent of S. Define $f(S)$ as

$$f(S) =_{syn} \quad \Gamma \to \Delta_{ns}.$$

Note that we have $\Gamma \subseteq \Sigma^q_{i+1} \subseteq \Sigma^q_{k-1}$, and $\Delta_{ns} \subseteq \Pi^q_{k-1}$; therefore, $f(S)$ is equivalent to a Π^q_{k-1}-formula. Let $q(S)$ be the Σ^q_{k-1}-witness query asking whether the sequent $f(S)$ is falsified by some truth assignment.

Suppose that T and T' are an upper sequent and the lower sequent of an inference step in $\pi[\vec{v}]$, respectively. It is easy to check that, if $f(T)$ is valid and $f(T')$ is invalid, then the inference step must be an $\exists$-right step whose auxiliary formula is strong-Π^q_{k-1}. Since $f(S)$ for every initial sequent S is valid while $f(S')$ for the endsequent S' is unsatisfiable, it follows that, in every path in $\pi[\vec{v}]$, there is an $\exists$-right step satisfying the above conditions.

The following algorithm solves $Witness[G^*_i, \Sigma^q_k]$ in polynomial-time using adaptive queries to Σ^q_{k-1}: given $\pi[\vec{v}]$, ask $q(S)$ for every sequent S in $\pi[\vec{v}]$, and find an $\exists$-right step such that $f(T)$ is valid and $f(T')$ is invalid, where T is the upper sequent and T' is the lower sequent. A solution for the witnessing problem is found by evaluating the appropriate quantifier-free subformulas of the auxiliary formula of this step, under the falsifying assignment for $f(T')$. □

6.4 Quantified Propositional Calculi for $\mathbf{TC}^0$

In this section we sketch sequent calculus systems for $\mathbf{TC}^0$. By taking advantage of the fact that many parsing operations are in $\mathbf{TC}^0$, we obtain a witnessing theorem for these $\mathbf{TC}^0$ sequent calculi similar to Theorem 6.5 for G_0.

We describe below the sequent calculus system PTK for propositional threshold logic by Buss and Clote [BC96] with minor modifications. The connectives of PTK are the negation $\neg$

and the unbounded fan-in threshold connectives Th_k for $k \geq 0$. Here $Th_k(A_1, \ldots, A_n)$ holds iff the number of true inputs is at least k. Note that $Th_k(A_1, \ldots, A_n)$ for $k = 1$ and $k = n$ are the $\bigvee_{i=1}^n A_i$ and $\bigwedge_{i=1}^n A_i$, respectively.

The initial sequents of PTK are:

(i) $\rightarrow \mathsf{T}$ and $\mathsf{F} \rightarrow$ and $A \rightarrow A$ for any formula A;

(ii) $Th_k() \rightarrow$ for $k \geq 1$; and

(iii) $\rightarrow Th_0(A_1, \ldots, A_n)$ for $n \geq 1$.

The structural rules of PTK are: weakening, contraction, exchange, and permutation of the arguments of Th in a formula. PTK has cut, $\neg$-left, $\neg$-right, and the following introduction rules for Th_k with $k \geq 1$:

$$Th_k\text{-left:}\frac{Th_k(A_2, \ldots, A_n), \Gamma \rightarrow \Delta \qquad A_1, Th_{k-1}(A_2, \ldots, A_n), \Gamma \rightarrow \Delta}{Th_k(A_1, \ldots, A_n), \Gamma \rightarrow \Delta}$$

$$Th_k\text{-right:}\frac{\Gamma \rightarrow \Delta, A_1, Th_k(A_2, \ldots, A_n) \qquad \Gamma \rightarrow \Delta, Th_{k-1}(A_2, \ldots, A_n)}{\Gamma \rightarrow \Delta, Th_k(A_1, \ldots, A_n)}$$

Let A be a formula of PTK. The *depth* of A is the maximum number of nestings of connectives in A.

Quantified Threshold Calculus (QTC) is obtained by introducing quantifiers in PTK, with the convention that the x-variables are used for bound variables and the p-variables denote free variables. For $i \geq 0$, define $T\Sigma_i^q$ to be the class of Σ_i^q-formulas over the connectives $\neg$ and Th_k for $k \geq 0$.

Definition 6.12. *TG is obtained by augmenting PTK with the quantifier-introduction rules. We require that the target of a $\exists$-right and a $\forall$-left be quantifier-free. TG_0 is TG with cuts only on quantifier-free formulas. For $d \geq 1$, $TG_0(d)$ is TG_0 with a restriction that all quantifier-free formulas in a proof be of depth $\leq d$.*

The $T\Sigma_1^q$-witnessing problem for $TG_0(d)$, written $Witness[TG_0(d), T\Sigma_1^q]$, is defined similarly to $Witness[G_0, \Sigma_1^q]$.

Theorem 6.13. *For every $d \geq 1$, $Witness[TG_0(d), T\Sigma_1^q]$ is solved by some* $\mathbf{TC}^0$*-function.*

Proof. Fix $d \geq 1$ and let $(\pi, \vec{v})$ be an instance of $Witness[TG_0(d), T\Sigma_1^q]$. Assume that the endformula of π is the formula of the form

$$A(\vec{p}) =_{syn} \exists x_1 \ldots \exists x_k F(\vec{p}, x_1, \ldots, x_k).$$

We can define π-prototypes, Herbrand π-disjunction, and the witness formulas for $TG_0(d)$ analogously to those for G_0. First, we claim that there is a $\mathbf{TC}^0$-function F such that $F(\pi, i)$ is the ith π-witness formula. Note that π-prototypes are easily seen to be $\mathbf{TC}^0$-recognizable. The ith component of a π-prototype A' is the propositional subformula B of A' such that, for all occurrence of B, there exists a subformula B_F of F such that $B_F =_{syn} (x_i)$ and $ID_{A'}(B) = ID_F(B_F)$, where ID is appropriately modified to take into account that the fan-in of the connective Th can be arbitrarily large. Let l be the largest fan-in of Th in π. Then ID is defined to be a sequence of $\log l$-bit numbers that specify the location of a subformula in a formula as a tree.

Next, using the methods of [BIS90], we can show that there is $B_d \in \mathbf{TC}^0$ such that $B_d(\psi, \vec{u})$ holds iff ψ is a PTK-formula of depth at most d and ψ evaluates to T under the truth assignment $\vec{u}$.

Finally, consider the $\mathbf{TC}^0$-relation $B_d(F(\pi, i), \vec{v})$. This is the bit graph of a function that solves $Witness[TG(d), T\Sigma_1^q]$. □

Theorem 6.14. *For Every* $\mathbf{TC}^0$*-function* F*, there exists* $d \in \mathbb{N}$ *such that* F *is reducible to* $Witness[TG_0(d), T\Sigma_1^q]$ *under many-one* $\mathbf{AC^0}$*-reduction.*

Proof. This is proven analogously to Theorem 6.3, using the fact that every $\mathbf{TC}^0$ predicate is computed by a **Dlogtime**-uniform family $\{T_n\}_n$ of polynomial-size PTK formulas [BIS90].

□

Chapter 7

Second Order Theories of Bounded Arithmetic

Buss's first-order theories S^i_2 and T^i_2 of bounded arithmetic are closely related to $\mathbf{PH}$, and they have been the objects of intense study since their introduction in Buss's 1985 dissertation published as [Bus86]. Buss's theories are preceded by the first-order theory $\mathbf{I\Delta}_0 + \Omega_1$ of Wilkie and Paris [PW81, WP87] and the equational theories PV of Cook [Coo75] for $\mathbf{P}$ and PRA of Dowd [Dow79], and followed by a number of new theories of bounded arithmetic corresponding to various complexity classes, such as Arai's **AID** [Ara00] for $\mathbf{NC}^1$, and theories for **NC**, $\mathbf{NC}^1$, logspace, and nondeterministic logspace by Clote and Takeuti [CT92], among others. A notable aspect of these developments is a great variety of styles in which these theories are defined: there are equational, first-order, and second-order theories characterizing complexity classes in different ways. As a result of these developments, our knowledge of the connections between complexity classes and logic has greatly increased, but it is nontrivial to compare these theories because of their different flavours.

Based on Zambella's work on second-order bounded arithmetic theories [Zam96], Cook [Coo02] has introduced second-order theories $\mathbf{V}^0$ and $\mathbf{V}^1$ corresponding to $\mathbf{AC^0}$ and $\mathbf{P}$, respectively, and his work led to a framework in which second-order theories characterizing

various complexity classes are developed in a unified way [CK03, NC04, Coo04]. The relatively simple syntax of this framework results in a simplification of both the description of the theories and the proofs of their properties, such as their connections to QPC which will be discussed in Chapter 8. Moreover, the unified syntax of these theories makes it easy to compare them.

The main result of this chapter is the introduction of the second-order theory $\mathbf{VNC}^1$ for $\mathbf{NC}^1$, which is inspired by Arai's **AID**. We obtain $\mathbf{VNC}^1$ by augmenting the base theory $\mathbf{V}^0$ with a scheme Σ_0^B-$TreeRec$ for a tree recursion. In Chapter 8 we will show that $\mathbf{VNC}^1$ is closely related to G_0.

In Section 7.4, we present alternative proofs for Pollett's result (Theorem 2.24) characterizing the $\hat{\Sigma}_k^b$-definable search problems of S_2^i and T_2^i for $i \geq 1$ and $k \geq i+2$. Our proofs are simple and presented in a more general setting. As a result, we obtain a characterization of the Σ_k^B-definable search problems for all $k \geq 2$ of all second-order theories $\mathbf{V}^0$, $\mathbf{VTC}^0$ of [NC04], $\mathbf{VNC}^1$, $\mathbf{V}^1$-Horn of [CK03], and $\mathbf{TV}^\mathbf{0}$, which has been previously unknown. In fact, this characterization applies to every theory T with $\mathbf{V}^0 \subseteq T \subseteq \mathbf{TV}^\mathbf{0}$, and it has an interesting consequence on the provability of reflection principles in bounded arithmetic, which will be discussed in Chapter 8.

We begin with the presentation of Cook's theories $\mathbf{V}^0$, $\mathbf{V}^i$, and $\mathbf{TV}^i$ and their syntax. These theories will be discussed in the coming chapter as well.

7.1 Basic Definitions

7.1.1 Syntax and semantics

Cook's "second order" theories are really two-sorted first order predicate calculus theories, and are based on the elegant syntax of Zambella [Zam96]. The underlying language $\mathcal{L}_A^2$ has variables $x, y, z, \ldots$ for the first sort, called *number variables*, and variables $X, Y, Z, \ldots$ of the second sort, called *string variables*. The number variables are intended to range over $\mathbb{N}$, and

the string variables are intended to range over finite sets of natural numbers (which represent binary strings). The reader may have noticed the similarity between this second-order approach and that of descriptive complexity (Section 2.1.2).

The language $\mathcal{L}_A^2$ extends the language of Peano Arithmetic, and consists of the function and predicate symbols $0, 1, +, \cdot, |\ |; \in, \leq, =_1, =_2$. Here $0, 1, +, \cdot$ are function symbols for numbers, and are intended to have their usual interpretation on $\mathbb{N}$. The function symbol $|X|$ denotes 1 plus the largest element in X, or 0 if X is empty (roughly the length of the corresponding string). $t \in X$ denotes set membership, but we usually use the notation $X(t)$ for $t \in X$, since we think of $X(t)$ as the t-th bit of the string X. Finally $=_1$ and $=_2$ denote equality on numbers and strings, respectively, but we will drop the subscripts, since they will be clear from context.

Number terms are built from the constants 0,1, variables $x, y, z, ...$, and length terms $|X|$ using $+$ and $\cdot$. The only *string terms* are string variables $X, Y, Z, ...$ The atomic formulas are $t = u$, $X = Y$, $t \leq u$, $t \in X$ for any number terms t, u and string variables X, Y. Formulas are built from atomic formulas using $\wedge, \vee, \neg$ and both number and string quantifiers $\exists x, \exists X, \forall x, \forall X$. Bounded number quantifiers are defined as usual, and the bounded string quantifier $\exists X \leq t\ \phi$ stands for $\exists X(|X| \leq t \wedge \phi)$ and $\forall X \leq t\ \phi$ stands for $\forall X(|X| \leq t \supset \phi)$, where X does not occur in the term t.

$\Sigma_0^B = \Pi_0^B$ is the set of all formulas in $\mathcal{L}_A^2$ such that all number quantifiers are bounded, and there are no string quantifiers. (There may be free string variables.) For $i > 0$, Σ_i^B is defined recursively to be the set of all formulas beginning with a block of zero or more bounded existential string quantifiers followed by a Π_{i-1}^B formula, and Π_i^B is defined dually. Note that for $i > 1$ our Σ_i^B and Π_i^B formulas correspond to *strict* versions of the formula classes $\Sigma_i^{1,b}$ and $\Pi_i^{1,b}$ defined in standard treatments because we require that all string quantifiers are in front. One reason for concentrating on this more restricted class of formulas is that the replacement scheme (asserting that a formula beginning with $\forall x \leq t \exists Y \leq s$, where t and s are terms, is equivalent to one beginning $\exists Z \leq s' \forall x \leq t$) does not hold in some of the theories considered here (without surprising complexity-theoretic consequences [CT04]). Another reason is that

this makes it easier to state and prove the connections of these theories to QPC proof systems.

7.1.2 Second order complexity classes

The basic complexity classes in this second-order context are classes of relations $R(\vec{x}, \vec{Y})$, where each x_i in the list $\vec{x}$ ranges over $\mathbb{N}$ and each Y_i in the list $\vec{Y}$ ranges over finite subsets of $\mathbb{N}$. When the complexity class is defined in terms of machines or circuits, we assume that each number input is presented in unary notation, and each finite subset input is presented by the corresponding bit string. Thus **P** is the class of such relations accepted in polynomial time on a Turing machine. Again, this setting is similar to the one used in descriptive complexity theory, and this observation almost immediately leads us to see the following:

Lemma 7.1. *([Coo02, Coo04]) A relation $R(\vec{x}, \vec{X})$ is in $\mathbf{AC^0}$ iff it is represented by some Σ_0^B-formula $\phi(\vec{x}, \vec{X})$.*

Second-order functions are either *number functions* or *string functions*. A number function $f(\vec{x}, \vec{Y})$ takes values in $\mathbb{N}$, and a string function $F(\vec{x}, \vec{Y})$ takes finite subsets of $\mathbb{N}$ as values. A function f or F is *polynomially bounded* (or *p-bounded*) if there is a polynomial $p(\vec{x}, \vec{y})$ such that $f(\vec{x}, \vec{Y}) \leq p(\vec{x}, |\vec{Y}|)$ or $|F(\vec{x}, \vec{Y})| \leq p(\vec{x}, |\vec{Y}|)$.

The following is a paraphrase of the definitions of polynomial-time, $\mathbf{NC}^1$, and $\mathbf{AC^0}$-functions (Definition 2.3) for the second-order context.

Definition 7.2. *The* bit graph B_F *of a string function F is defined by*

$$B_F(i, \vec{x}, \vec{Y}) \leftrightarrow F(\vec{x}, \vec{Y})(i)$$

We say that a number function F is a polynomial-time function *if it is p-bounded and its graph is in $\mathbf{P}$. A string function is called polynomial-time if it is p-bounded and its bit graph is in $\mathbf{P}$. $\mathbf{AC^0}$-functions and $\mathbf{NC}^1$-functions are defined similarly by replacing $\mathbf{P}$ with $\mathbf{NC}^1$ and $\mathbf{AC^0}$, respectively.*

Definition 7.3. *Let T be a theory whose language extends $\mathcal{L}_A^2$. A string function $F(\vec{x}, \vec{X})$ is* Σ_1^B-definable *in T if there is a Σ_1^B-formula ϕ such that*

$$Y = F(\vec{x}, \vec{X}) \leftrightarrow \phi(\vec{x}, \vec{X}, Y) \text{ and}$$
$$T \vdash \forall \vec{x} \forall \vec{X} \exists! Y \phi(\vec{x}, \vec{X}, Y)$$

The Σ_1^B-definability for a number function $f(\vec{x}, \vec{X})$ is defined similarly.

Definition 7.4. *Let Q be a search problem. We say that Q is Σ_i^B-definable in theory T if the following two conditions are met: (i) there is a Σ_i^B-formula $\phi(\vec{X}, Y)$ with all free variables indicated such that*

$$\underline{\mathbb{N}} \models (\forall \vec{X})(\forall Y)[\phi(\vec{X}, Y) \supset Y \in Q(\vec{X})],$$

and (ii) T proves $(\forall \vec{X})(\exists Y)\phi(\vec{X}, Y)$. We call $\phi(\vec{X}, Y)$ a Σ_i^B-defining formula *of Q.*

7.1.3 The theory $\mathbf{V}^0$

The base theory $\mathbf{V}^0$ [CK03, Coo02] (called $\mathbf{\Sigma}_0^p$-$comp$ in [Zam96]) is associated with the complexity class $\mathbf{AC^0}$, and all second order theories considered in this chapter are extensions of $\mathbf{V}^0$. The language of $\mathbf{V}^0$ is $\mathcal{L}_A^2$. The axioms of $\mathbf{V}^0$ consist of the universal closures of the Σ_0^B formulas 2-$BASIC$ together with the Σ_0^B comprehension scheme below. 2-$BASIC$ consists of

B1. $x + 1 \neq 0$

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

B3. $x + 0 = x$

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

B5. $x \cdot 0 = 0$

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

B7. $x \leq x + y$

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

B9. $0 + 1 = 1$

B10. $0 \leq x$

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

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

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

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

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

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

SE. $X = Y \leftrightarrow [|X| = |Y| \wedge \forall i < |X|(X(i) \leftrightarrow Y(i))]$

The Σ_0^B comprehension scheme is

$$\Sigma_0^B\text{-COMP:} \qquad \exists X \leq y \forall z < y(X(z) \leftrightarrow \phi(z, \vec{x}, \vec{Y})) \tag{7.1}$$

where $\phi(z, \vec{x}, \vec{Y})$ is any Σ_0^B formula not containing X.

Although $\mathbf{V}^0$ does not have an explicit induction scheme, axioms L1 and L2 tell us that if X is nonempty then it has a largest element, and thus we can show that $\mathbf{V}^0$ proves a minimization scheme, and the induction formula

$$[X(0) \wedge \forall y < z(X(y) \supset X(y+1))] \supset X(z) \tag{7.2}$$

(See [CK03] or [Coo02] for details.) From this and Σ_0^B-COMP we have

Theorem 7.5. *([Coo02])* $\mathbf{V}^0$ *proves the scheme*

$$\Sigma_0^B\textit{-IND:} \qquad [\phi(0) \wedge \forall x(\phi(x) \supset \phi(x+1))] \supset \forall z \phi(z)$$

where $\phi(x)$ *is any* Σ_0^B*-formula (possibly containing free variables other than* x*).*

Theorem 7.6. *([Coo02, Coo04]) A function (string or number) is* Σ_1^B*-definable in* $\mathbf{V}^0$ *iff it is an* $\mathbf{AC^0}$*-function.*

We use as a pairing function the term

$$\langle x, y \rangle =_{def} (x+y)(x+y+1) + 2y \tag{7.3}$$

Then $\mathbf{V}^0$ proves that the map $(x, y) \mapsto \langle x, y \rangle$ is a one-one map from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$. We use this idea to define a binary array X using the definition $X(x, y) = X(\langle x, y \rangle)$. By iterating the pairing function we can define a multidimensional array $X(\vec{x})$. It is easy to see that $\mathbf{V}^0$ proves the analog of Σ_0^B-COMP (7.1) for multidimensional arrays.

If we think of Z as a two-dimensional array, then we can represent row x in this array by $Z^{[x]}$ [Zam96], where $G(x, Z) = Z^{[x]}$ is the $\mathbf{FAC^0}$ string function with bit-defining axiom

$$Z^{[x]}(i) \leftrightarrow i < |Z| \wedge Z(x, i) \tag{7.4}$$

We can add this string function $Z^{[x]}$ together with its defining equation (7.4) to form a conservative extension of $\mathbf{V}^0$.

7.1.4 $\mathbf{V}^i$ and $\mathbf{TV}^i$

$\mathbf{V}^0$ generalizes to the theory $\mathbf{V}^i$ for $i \geq 1$ in the following way:

Definition 7.7. *([Coo02, Coo04]) For $i \geq 1$, $\mathbf{V}^i$ is defined from $\mathbf{V}^0$ by replacing the comprehension scheme Σ^B_0-COMP by Σ^B_i-COMP. $\mathbf{V}^i$ and $\mathbf{V}^0$ are over the same language $\mathcal{L}^2_A$.*

For each $i \geq 1$, $\mathbf{V}^i$ is essentially equivalent to Buss's S^i_2 in the following sense: semantically, there is a bijection between (isomorphism types of) models of $\mathbf{V}^i$ and models of S^i_2; and syntactically, there are translations between Σ^b_i-theorems of S^i_2 and Σ^B_i-theorems $\mathbf{V}^i$ such that both $\mathbf{V}^i$ and S^i_2 prove the correctness of the translations. This relationship is known as an *RSUV isomorphism* [Kra90, Raz93, Tak93, Kra95, Coo04].

Theorem 7.8. *([Coo04]) For each $i \geq 1$, $\mathbf{V}^i$ is RSUV-isomorphic to S^i_2.*

Cook shows in [Coo02] that $\mathbf{V}^1$ is equivalent to a theory axiomatized by 2-$BASIC$, Σ^B_0-COMP, and Σ^B_1-IND, where Σ^B_1-IND is the following scheme for $\phi \in \Sigma^B_i$:

$$[\phi(0) \wedge \forall y < z(\phi(y) \supset \phi(y+1))] \supset \phi(z)$$

For $i \geq 2$, $\mathbf{V}^i$ is axiomatized similarly using Σ^B_i-IND instead of Σ^B_i-COMP. If we think of a string X as a binary representation of a number, then, via an appropriate syntactic translation between Σ^B_i and Σ^b_i, the scheme Σ^B_i-IND is essentially Σ^b_i-LIND in first-order bounded arithmetic. This way it is intuitively clear that $\mathbf{V}^i$ and S^i_2 are RSUV isomorphic.

A second-order scheme on Σ^B_i-formulas that corresponds to Σ^b_i-IND is obtained as follows. Seeing strings X as binary numbers, the string successor function $S(X)$ that computes (the binary representation of) $X + 1$ is an $\mathbf{AC^0}$-function, and therefore its bit graph is represented by some Σ^B_0-formula. For $i \geq 0$, Cook in [Coo04] defines Σ^B_i-String-IND scheme as

$$\phi(\mathbf{0}) \wedge (\forall X)[\phi(X) \supset \phi(S(X))] \supset \phi(Y)$$

where $\phi \in \Sigma_i^B$. It is intuitively clear that Σ_i^B-String-IND is RSUV isomorphic to Σ_i^b-IND.

Definition 7.9. *([Coo04]) For $i \geq 0$, $\mathbf{TV}^i$ is defined by augmenting $\mathbf{V}^0$ with the Σ_i^B-String-IND scheme.*

Theorem 7.10. *([Coo04]) For $i \geq 1$, $\mathbf{TV}^i$ is RSUV-isomorphic to T_2^i.*

It is an interesting fact that $\mathbf{TV}^0$ is essentially a second-order version of Cook's **QPV** [Coo04] and it captures the complexity class $\mathbf{P}$.

7.2 The theory $\mathbf{VNC}^1$

We define the system $\mathbf{VNC}^1$ by adding a tree recursion axiom scheme Σ_0^B-$TreeRec$ to $\mathbf{V}^0$. This scheme is intended to take the place of the predicates $A^{\ell,B,\overline{D},I}$ and their defining axioms in Arai's theory **AID** [Ara00], which captures reasoning in **Alogtime** (uniform $\mathbf{NC}^1$). Our scheme is a simplified second order version of Arai's Σ_0^b-RD ([Ara00] Definition 7.1), using the idea of the heap data structure.

The Σ_0^B-$TreeRec$ scheme is

$$\begin{aligned} &\exists Z \leq 2a \forall i < a[(Z(i+a) \leftrightarrow \psi(i)) \wedge \\ &0 < i \supset (Z(i) \leftrightarrow \phi(i)[Z(2i), Z(2i+1)])] \end{aligned} \tag{7.5}$$

where $\phi(i)[p,q]$ and $\psi(i)$ are Σ_0^B formulas (which do not contain Z but may contain other parameters) and ϕ contains atoms p, q to be replaced in the axiom by $Z(2i), Z(2i+1)$.

The idea is that the vector Z assigns truth values to the nodes of a binary tree, where the nodes are indexed by the variable $i, 0 < i \leq 2a-1$; see Figure 7.1. The leaves of the tree are indexed by any i such that $a \leq i \leq 2a-1$ and leaf number i is assigned value $\psi(i)$. The internal nodes of the tree are indexed by any i such that $1 \leq i \leq a-1$, and the value $Z(i)$ of node i is determined by the values $Z(2i), Z(2i+1)$ of its two children by the formula ϕ. The root of the tree is indexed by $i = 1$, so $Z(1)$ is the output of the recursion.

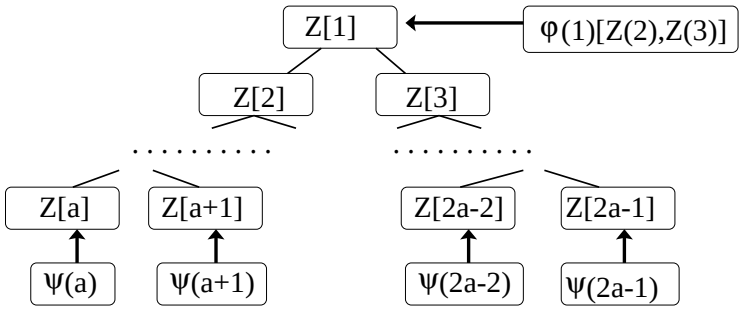

Figure 7.1: Σ_0^B-$TreeRec$. The formula $\psi(i)$ defines the boolean values of the leaf nodes, and $\phi(i)[Z(2i), Z(2i+1)]$ defines the value of node i based on the values of its children $2i$ and $2i+1$.

For Σ_0^B formulas $\phi(i, \vec{x}, \vec{X})[p, q]$ and $\psi(i, \vec{x}, \vec{X})$ in the Σ_0^B-$TreeRec$ scheme (7.5) we define the Σ_0^B formula $B^{\phi,\psi}(a, \vec{x}, \vec{X}, Z)$ to be the part of (7.5) which comes after $\exists Z \leq 2a$. That is,

$$\begin{aligned} B^{\phi,\psi}(a, \vec{x}, \vec{X}, Z) \ \equiv \ \forall i < a[(Z(i+a) \leftrightarrow \psi(i)) \wedge \\ 0 < i \supset (Z(i) \leftrightarrow \phi(i)[Z(2i), Z(2i+1)])] \end{aligned} \tag{7.6}$$

Lemma 7.11. *For all Σ_0^B formulas ϕ, ψ*

$$\mathbf{VNC}^1 \vdash \exists Z! \leq 2a B^{\phi,\psi}(a, \vec{x}, \vec{X}, Z) \tag{7.7}$$

Proof. Existence of Z follows from (7.5). Uniqueness can be proved in $\mathbf{V}^0$ using Σ_0^B-IND. □

Defining $\mathbf{NC}^1$ relations and functions in $\mathbf{VNC}^1$

We start by showing how to define $\mathbf{NC}^1$ relations in $\mathbf{VNC}^1$. Every formula $B^{\phi,\psi}$ (7.6) defines a relation $R^{\phi,\psi}$ (computed by the recursion scheme (7.5)) with defining axiom

$$R^{\phi,\psi}(a, i, \vec{x}, \vec{X}) \leftrightarrow \exists Z \leq 2a(B^{\phi,\psi}(a, \vec{x}, \vec{X}, Z) \wedge Z(i)) \tag{7.8}$$

Lemma 7.12. *The relation $R^{\phi,\psi}$ is in $\mathbf{NC}^1$, for each pair $\phi[p, q], \psi$ of Σ_0^B formulas.*

Proof. By Lemma 7.1, each Σ_0^B-formula represents an $\mathbf{AC^0}$ relation, which is therefore in $\mathbf{Alogtime}$. To prove the lemma, it suffices to show there exists an indexed alternating Turing

machine M with inputs $(a, i, \vec{x}, \vec{X})$ (where number inputs are presented in unary notation) which computes $R^{\phi,\psi}$ in time $O(\log n)$, where n is the length of the input.

The machine M starts by guessing the binary notation for the input i, and verifying its guess in time $O(\log n)$ using its indexed access to the input tape. It then guesses that $Z(i)$ is true, and verifies its guess by recursively guessing and verifying $Z(j)$ for various values of j. In general, M verifies its guess for $Z(j)$ as follows: First it guesses whether $j < a$ or $j \geq a$. If the guess is $j \geq a$, then it verifies the guess, and verifies $Z(j) \leftrightarrow \psi(j - a, \vec{x}, \vec{X})$, all in time $O(\log n)$. If the guess is $j < a$ it branches universally, verifying the guess on one branch and guessing $Z(2j), Z(2j + 1)$ on the other branch. After the second branch it next does a three-way universal branch: (i) verify $Z(j) \leftrightarrow \phi(j, \vec{x}, \vec{X})[Z(2j), Z(2j + 1)]$, (ii) verify $Z(2j)$ recursively, and (iii) verify $Z(2j + 1)$ recursively.

Note that the depth of the recursion is proportional to the depth of the tree recursion defined by (7.5), which is $O(\log a) = O(\log n)$. □

We now expand the language $\mathcal{L}_A^2$ to $\mathcal{L}_{TreeRec}$ by putting in a predicate symbol $R^{\phi,\psi}$ for each relation $R^{\phi,\psi}$ defined in (7.8). Then $\Sigma_0^B(\mathcal{L}_{TreeRec})$ denotes the class of formulas in this language with no string quantifiers, and all number quantifiers bounded.

Lemma 7.13. *The class of $\Sigma_0^B(\mathcal{L}_{TreeRec})$ formulas represents precisely the* $\mathbf{NC}^1$ *relations.*

Proof. Every such formula represents an $\mathbf{NC}^1$ relation, by the previous lemma, and the easy fact that the $\mathbf{NC}^1$ relations are closed under bounded number quantification and the Boolean operations.

Conversely, we appeal to Theorem 3.1 of [Ara00], which states that every $\mathbf{NC}^1$ relation R is $\Sigma_0^b(\mathcal{L}_{\mathbf{AID}})$-definable in **AID**. By Theorem 7.17, the formula that defines R in **AID** corresponds to a $\Sigma_0^B(\mathcal{L}_{TreeRec})$ formula. □

We denote by $\mathbf{VNC}^1(\mathcal{L}_{TreeRec})$ the theory whose language is $\mathcal{L}_{TreeRec}$ and whose axioms are those of $\mathbf{VNC}^1$ together with the defining axioms (7.8). Clearly $\mathbf{VNC}^1(\mathcal{L}_{TreeRec})$ is a

conservative extension of $\mathbf{VNC}^1$. By $\Sigma_0^B(\mathcal{L}_{TreeRec})$-COMP we mean the scheme (7.1), where ϕ is any $\Sigma_0^B(\mathcal{L}_{TreeRec})$ formula.

Lemma 7.14. $\mathbf{VNC}^1(\mathcal{L}_{TreeRec})$ *proves the* $\Sigma_0^B(\mathcal{L}_{TreeRec})$*-COMP scheme.*

Proof. First note that $\mathbf{VNC}^1$ Δ_1^B-defines each relation $R^{\phi,\psi}$, since the Σ_1^B formula representing $R^{\phi,\psi}$ in (7.8) is provably equivalent to a Π_1^B formula. That is, from (7.7) it follows that $\mathbf{VNC}^1$ proves

$$\exists Z \leq 2a(B^{\phi,\psi}(a,\vec{x},\vec{X},Z) \wedge Z(i)) \;\leftrightarrow\; \forall Z \leq 2a((B^{\phi,\psi}(a,\vec{x},\vec{X},Z) \supset Z(i))$$

The lemma would follow easily from this and the Σ_0^B-COMP axioms if $\mathbf{VNC}^1$ proves the Σ_0^B-replacement scheme, but results in [CT04] suggest that this is unlikely. Instead we show that $\mathbf{VNC}^1(\mathcal{L}_{TreeRec})$ proves $\Sigma_0^B(\mathcal{L}_{TreeRec})$-COMP, by structural induction on formulas ϕ in $\Sigma_0^B(\mathcal{L}_{TreeRec})$. The induction step, when ϕ is built from simpler formulas from the Boolean operations or bounded number quantification, is straightforward. For example, if $\phi(z)$ is $\exists x \leq t\psi(x,z)$, then using the pairing function (7.3) we have by the induction hypothesis

$$\mathbf{VNC}^1(\mathcal{L}_{TreeRec}) \vdash \exists X \forall x \leq t \forall z < y(X(x,z) \leftrightarrow \psi(x,z))$$

Now by Σ_0^B-COMP,

$$\mathbf{V}^0 \vdash \exists X' \leq y \forall z < y(X'(z) \leftrightarrow \exists x \leq t\, X(x,z))$$

Thus $\mathbf{VNC}^1(\mathcal{L}_{TreeRec})$ proves comprehension for ϕ.

The base case of the induction is straightforward except for the case of one of the new relation symbols $R^{\phi,\psi}$. Here it suffices to show $\mathbf{VNC}^1(\mathcal{L}_{TreeRec})$ proves (7.1) where $\phi(z,\vec{x},\vec{Y})$ is replaced by $R^{\phi,\psi}(a,i,\vec{x},\vec{X})$ when z is one of the number variables $a, i, \vec{x}$. The case in which z is i follows from (7.7). Now consider the case in which z is in $\vec{x}$. (The case in which z is a is similar.) To simplify notation, we assume x is $\vec{x}$. By (7.8) it suffices to show

$$\mathbf{VNC}^1 \vdash \exists W \forall x < y[W(x) \leftrightarrow \exists Z \leq 2a(B^{\phi,\psi}(a,x,\vec{X},Z) \wedge Z(i))] \tag{7.9}$$

The RHS of this defines W in terms of trees Z_x for $x = 0, 1, ..., y-1$. In order to show that the existence of W follows from the Σ^B_0-$TreeRec$ scheme (7.5) we collect all of these trees into one large tree U which has them attached to y leaves of the top part of U.

To describe in $\mathbf{VNC}^1$ these tree embeddings we use the fact that the first order theory $\mathbf{I\Delta_0}$, and hence $\mathbf{VNC}^1$, defines functions such as $|x|$ (the length of x in binary) and $2^{|x|}$ and proves their basic properties (see for example [Bus98a, Coo02]).

Tree Z_x is represented in U by the subtree of U rooted at node $root(x) = 2^{|y|} + x$. Note that these y root nodes are consecutive nodes at level $|y|$ in the tree U (where the root of U is at level 0). In general, node i of tree Z_x is at level $level(i) = |i| - 1$ in Z_x and hence at level $level(i) + |y|$ in U. In fact, node i in tree Z_x is represented by node

$$node(i, x) = root(x) \cdot 2^{level(i)} + (i - 2^{level(i)})$$

in U. Note that the leaves of Z_x are at level $|a-1|$ in Z_x, except some may be at level $|a-1|-1$. The leaves of interest in U are the deeper leaves of the embedded trees Z_x, and these have level $|a - 1| + |y|$.

The function $node(i, x)$ is injective for pairs i, x such that $1 \leq i$ and $0 \leq x < y$, and its inverses $icomp(j)$ and $xcomp(j)$ are definable in $I\Delta_0$, and $I\Delta_0$ proves for i, x satisfying these conditions, that if $j = node(i, x)$, then $i = icomp(j)$ and $x = xcomp(j)$.

The formulas $\phi(i, x, \vec{X})[p, q], \psi(i, x, \vec{X})$ used to define the tree Z_x determine

$$\phi'(j, y, a, \vec{X})[p, q], \psi'(j, y, a, \vec{X})$$

to define the tree U, where

$$\begin{aligned} \phi'(j, y, a, \vec{X})[p, q] \;\equiv\; & icomp(j) < a \wedge \phi(icomp(j),\ xcomp(j), \vec{X})[p, q] \;\vee \\ & icomp(j) \geq a \wedge \psi(icomp(j) \dot{-} a,\ xcomp(j), \vec{X}) \end{aligned}$$

$$\psi'(j, y, a, \vec{X}) \equiv \psi(icomp(j + a') \dot{-} a,\ xcomp(j + a'), \vec{X})$$

where a' is defined below. By (7.5) we have

$$\mathbf{VNC}^1 \vdash \exists U \leq 2a'\ B^{\phi', \psi'}(a', y, a, \vec{X}, U) \tag{7.10}$$

where $a' = 2^{|a \dot{-} 1| + |y|}$. The reason for this value of a' is that all leaves of the tree U are at level $|a - 1| + |y|$, as noted above.

Finally $\mathbf{VNC}^1$ proves the existence of W in (7.9) using Σ^B_0-COMP and the definition

$$W(x) \leftrightarrow U(node(i, x))$$

where U is obtained from (7.10). In order to prove that W defined in this way satisfies (7.9), $\mathbf{VNC}^1$ proves each tree Z_x is embedded as claimed in U; that is

$$0 < i < 2a \wedge x < y \wedge B^{\phi,\psi}(a, x, \vec{X}, Z) \wedge B^{\phi',\psi'}(a', y, a, \vec{X}, U) \supset$$
$$(Z(i) \leftrightarrow U(node(i, x))$$

This can be done using Σ^B_0-IND on $(2a \dot{-} i)$. □

Recall from Definition 7.2 that a string function $F(\vec{x}, \vec{X})$ is an $\mathbf{NC}^1$-function iff it is p-bounded and its bit graph is in $\mathbf{NC}^1$. It is easy to check that a number function $f(\vec{x}, \vec{X})$ is an $\mathbf{NC}^1$-function iff it satisfies $f(\vec{x}, \vec{X}) = |F(\vec{x}, \vec{X})|$ for some string $\mathbf{NC}^1$-function F.

Theorem 7.15. *A function (string or number) is Σ^B_1-definable in $\mathbf{VNC}^1$ if and only if it is an $\mathbf{NC}^1$-function.*

Proof. We first prove the *if* direction. Let $F(\vec{x}, \vec{X})$ be a string $\mathbf{NC}^1$-function, and let $R_F(i, \vec{x}, \vec{X})$ be its bit graph. By Lemma 7.13, there is a $\Sigma^B_0(\mathcal{L}_{TreeRec})$-formula $\psi(i, \vec{x}, \vec{X})$ that represents R in $\mathbb{N}$. Then there is a term t such that

$$Y = F(\vec{x}, \vec{X}) \quad \text{if and only if} \quad \underline{\mathbb{N}} \models (\forall i \leq t(\vec{x}, \vec{X}))[Y(i) \leftrightarrow \psi(i, \vec{x}, \vec{X})]$$

It is easy to see that $\mathbf{VNC}^1$ proves $(\exists! Y)(\forall i \leq t(\vec{x}, \vec{X}))[Y(i) \leftrightarrow \psi(i, \vec{x}, \vec{X})]$ using $\Sigma^B_0(\mathcal{L}_{TreeRec})$-COMP (Lemma 7.14). Thus, F is Σ^B_1-COMP (Lemma 7.14). Thus, F is Σ^B_1-definable in $\mathbf{VNC}^1$.

The *only-if* direction is stated as Theorem 8.5, which is proven via the QPC translation theorem for $\mathbf{VNC}^1$ (Theorem 8.3). We also presented a direct proof in [CM04] of a slightly stronger assertion. □

We state two more theorems concerning $\mathbf{VNC}^1$. The proofs of these theorems are found in [CM04].

Theorem 7.16. *([Ngu04, CM04])* $\mathbf{VNC}^1$ *is finitely axiomatizable.*

Theorem 7.17. *([CM04])* $\mathbf{VNC}^1$ *is RSUV isomorphic to* $\mathbf{AID} + \Sigma_0^b$*-CA.*

7.3 Sequent Calculus LK^2

First, we present Gentzen's first-order sequent calculus LK:

Definition 7.18. *([Gen35, Bus98b, Coo02])* LK *is a sequent calculus system for first-order logic. Its inference rules include those of* PK *plus the following quantifier inference rules:*

$$\exists\text{-left} : \frac{A(b), \Gamma \to \Delta}{\exists x A(x), \Gamma \to \Delta} \quad \exists\text{-right} : \frac{\Gamma \to \Delta, A(t)}{\Gamma \to \Delta, \exists x A(x)}$$

$$\forall\text{-left} : \frac{A(t), \Gamma \to \Delta}{\forall x A(x), \Gamma \to \Delta} \quad \forall\text{-right} : \frac{\Gamma \to \Delta, A(b)}{\Gamma \to \Delta, \forall x A(x)}$$

where t *is any term and* b *is an eigenvariable that does not occur in the lower sequent.*

As usual, the initial sequents of LK *are logical axioms of the form (a)* $\to \mathsf{T}$*, (b)* $\mathsf{F} \to$*, or (c)* $A \to A$ *for all formula* A*.*

LK is extended to LK^2 for second-order logic as follows:

Definition 7.19. *([Coo02])* LK^2 *is obtained by augmenting* LK *with the following second-order quantifier introduction rules:*

$$\textit{String } \exists\textit{-left} : \frac{A(\beta), \Gamma \to \Delta}{\exists x A(x), \Gamma \to \Delta} \quad \textit{String } \exists\textit{-right} : \frac{\Gamma \to \Delta, A(\alpha)}{\Gamma \to \Delta, \exists x A(x)}$$

$$\textit{String } \forall\textit{-left} : \frac{A(\alpha), \Gamma \to \Delta}{\forall x A(x), \Gamma \to \Delta} \quad \textit{String } \forall\textit{-right} : \frac{\Gamma \to \Delta, A(\beta)}{\Gamma \to \Delta, \forall x A(x)}$$

where α *and* β *denote string free variables, and, in String* $\exists$*-left and String* $\forall$*-right,* β *is an eigenvariable that does not occur in the lower sequent.*

The following can be proven using a standard method in proof theory [Tak87]. This is called an *anchored completeness* in [Coo02] and a *free-cut free elimination* in [Bus98b].

Theorem 7.20. *Let T be a second-order theory of bounded arithmetic. If ϕ is a theorem of T, then there is a tree-like LK^2-proof π of ϕ satisfying the following: (i) every initial sequent of π is a logical axiom, an equality axiom, or an axiom of T; and (ii) every cut formulas of π occurs in an initial sequent.*

Let $i \geq 1$. Theorem 7.20 means that theorems of $\mathbf{V}^i$ and $\mathbf{TV}^i$ have LK^2-proofs in which complex formulas such as Σ^B_i-IND or Σ^B_i-String-IND appear as initial sequents, and this is not very nice. This can be avoided by using the following additional inference rules:

$$\Sigma^B_i\text{-IND rule:}\frac{\Gamma, A(b) \to A(b+1), \Delta}{\Gamma, A(0) \to A(t), \Delta} \quad \Sigma^B_i\text{String–IND rule:}\frac{\Gamma, A(\beta) \to A(S(\beta)), \Delta}{\Gamma, A(\vec{0}) \to A(X), \Delta}$$

where b and β are free number variable and string variable, respectively, that do not occur in the lower sequent, X is any string variable, and t is any number term. $LK^2 + \Sigma^B_i$-IND and $LK^2 + \Sigma^B_i$-String-IND are defined to be LK^2 plus the corresponding induction rule.

The following is a variant of the anchored completeness for LK plus induction rules. Again it can be proven by a standard method in proof theory: see [Tak87, Bus98b, Coo02].

Theorem 7.21. *Let $i \geq 1$. If ϕ is a bounded theorem of $\mathbf{V}^i$, then there is a tree-like $LK^2 + \Sigma^B_i$-IND proof π of ϕ satisfying the following: (i) every initial sequent of π is a logical axiom, an equality axiom, or an axiom of 2-$BASIC$ or Σ^B_0-COMP; and (ii) every cut formula of π either occurs in an initial sequent or is a Σ^B_i-formula $A(0)$ or $A(t)$ in Σ^B_i-IND step.*

Similarly for $\mathbf{TV}^i$ and $LK + \Sigma^B_i$-String-IND.

7.4 The Σ^B_k-Witnessing Theorems for Large k

In this Section, we present alternative proofs for Pollett's result (Theorem 2.24) characterizing the $\hat{\Sigma}^b_k$-definable search problems of S^i_2 and T^i_2 for $i \geq 1$ and $k \geq i + 2$. Our proofs are

simple and presented in a more general setting. As a result, we obtain a characterization of the Σ^B_k-definable search problems for all $k \geq 2$ of all second-order theories $\mathbf{V}^0$, $\mathbf{VTC}^0$ of [NC04], $\mathbf{VNC}^1$, $\mathbf{V}^1$-Horn of [CK03], and $\mathbf{TV}^\mathbf{0}$, which has been previously unknown. In fact, this characterization applies to every theory T with $\mathbf{V}^0 \subseteq T \subseteq \mathbf{TV}^\mathbf{0}$, and it has an interesting consequence on the provability of reflection principles in bounded arithmetic, which will be discussed in Chapter 8.

The following is the main result of this Chapter:

Theorem 7.22.
(i) ([Pol99]) Let $i \geq 1$ and T denote either S^i_2 or T^i_2. For every $k \geq i+2$, a search problem Q is in $\mathbf{FP}^{\Sigma^p_{k-1}}[wit, O(1)]$ if and only if Q is $\hat{\Sigma}^b_k$-definable in T.
(ii) Let T be a second-order theory such that $\mathbf{V}^0 \subseteq T \subseteq \mathbf{TV}^0$. For every $k \geq 2$, a search problem Q is in $\mathbf{FP}^{\Sigma^p_{k-1}}[wit, O(1)]$ if and only if Q is Σ^B_k-definable in T.

Proof. The 'if' direction of (i) is proven in Theorem 7.23 below. The 'only if' direction is immediate from Lemma 7.25 since S^1_2 is RSUV isomorphic to $\mathbf{V}^1$ (Theorem 7.8), which contains $\mathbf{V}^0$.

The 'if' direction of (ii) follows from the (ii) of Theorem 7.23 with $i = 0$, showing that Σ^B_k-definable search problems of $\mathbf{TV}^0$ are in $\mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(1)]$. The 'only if' direction is immediate from Lemma 7.25. □

The following is the 'if' directions of Theorem 7.22.

Theorem 7.23. *For every $i \geq 1$ and $k \geq i+2$, every $\hat{\Sigma}^b_k$-definable search problem of T^i_2 is in $\mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(1)]$. (ii) For every $i \geq 0$ and $k \geq i+2$, every Σ^B_k-definable search problem of $\mathbf{TV}^i$ is in $\mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(1)]$.*

Proof. We prove (i) below. The assertion (ii) can be proven in a completely analogous way. For $i \geq 1$, (ii) also follows from the RSUV-isomorphism between T^i_2 and $\mathbf{TV}^i$ (Theorem 7.10).

Let $i \geq 0$ and $k \geq i+2$ be arbitrary. Our proof is based on the same idea as the proof that $Witness[G^*_i, \Sigma^q_k]$ is in $\mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(\log n)]$.

We consider a bounded axiomatization of T_2^i, which consists of quantifier-free axioms of BASIC plus the following bounded formulation of Σ_i^b:

$$A(0) \wedge (\forall x \leq a)[A(x) \supset A(Sx)] \supset A(a),$$

where $A \in \Sigma_i^b$ and a is a free variable. Note that this Σ_i^b-IND axiom is a Σ_{i+1}^b-formula, and therefore T_2^i is axiomatized by Σ_{i+1}^b-formulas.

Assume that a search problem Q is $\hat{\Sigma}_k^b$-definable in T_2^i. We show that Q is solved by a polynomial-time machine that makes $O(1)$ witness queries to a $\mathbf{\Sigma}_{k-1}^p$-oracle. By the definition of Σ_k^b-definability (Definition 2.21), there is a Σ_k^b-defining formula

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

where $B \in \Pi_{k-1}^b$ with all free variables indicated and bounds on $\vec{z}$ supressed, such that

$$\mathbb{N} \models (\forall \vec{x})(\forall y)[(\exists \vec{z}) B(\vec{x}, y, \vec{z}) \supset y \in Q(\vec{x})] \qquad \text{and} \qquad T_2^i \vdash (\exists y)(\exists \vec{z}) B(\vec{x}, y, \vec{z}).$$

Let π be an LK-proof of the sequent $\rightarrow (\exists y)(\exists \vec{z}) B(\vec{x}, y, \vec{z})$ from the axioms of T_2^i and the equality axioms. We assume that π is tree-like and in free variable normal form. Moreover, by the free-cut elimination of LK [Bus98b], we assume without loss of generality that every cut formula in π is either an axiom of T_2^i or an equality axiom; that is, every cut formula is Σ_{i+1}^b. Note that the endformula is Σ_k^b with $k > i + 1$. LK is defined in Definition 7.18 in the next Section, and Theorem 7.20 is the second-order analogue of the free-cut elimination of LK. The free-cut elimination is also known as the anchored completeness of LK [Bus98b, Coo02].

Let an instance $\vec{a}$ of Q be given, where $\vec{a}$ is a tuple of natural numbers. Our goal is to find any $b \in \mathbb{N}$ such that

$$\mathbb{N} \models (\exists \vec{z}) B(\vec{a}, b, \vec{z}). \tag{7.11}$$

Let $\pi[\vec{a}]$ be the result of substituting the values $\vec{a}$ for the parameter variables $\vec{x}$ in π. Note that all the free variables of $\pi[\vec{a}]$ are eigenvariables for $\exists$-right and $\forall$-left steps.

From here, we proceed analogously to the proof of Theorem 6.11. For each sequent S of $\pi[\vec{a}]$, define $f(S)$ to be the sequent obtained from S by removing every strong-Σ_k^b-formula.

Every strong-Σ^b_k-formula must be in the succedent of S. Define $q(S)$ to be the witness query asking whether there is a value assignment to the free variables of $f(S)$ that falsifies $f(S)$. Note that $q(S)$ is a $\mathbf{\Sigma}^b_{k-1}$-witness query.

The rest of the proof is identical to that of Theorem 6.9. Our algorithm asks $q(S)$ for every sequent in $\pi[\vec{a}]$, and finds an $\exists$-right step with upper sequent T and lower sequent T' such that $f(T)$ is valid and $f(T')$ is invalid. Note that the number of witness queries is $O(1)$ since it only depends on π, which is not part of the input to the algorithm. The witness to the invalidity of $f(T')$ is a value assignment τ that falsifies T', and it follows that the auxiliary formula A of this $\exists$-right step is a valid sentence under this assignment τ. A solution $b \in \mathbb{N}$ satisfying (7.11) is obtained by evaluating under τ the term t in A that is the target of the $\exists$-right that introduces $\exists y$ into A. Finally, such $\exists$-right step exists because $f(S)$ is valid for every initial sequent S, while $f(S')$ is unsatisfiable for the endsequent S'. □

The above proof is meaningful only when $k \geq i+2$. If $k \leq i+1$, the proof still works but it only shows that $\hat{\Sigma}^b_k$-definable search problems are in $\mathbf{FP}^{\mathbf{\Sigma}^p_{i+1}}[wit, O(1)]$, which is a trivial statement. Note that the complexity of the witness queries remains $\mathbf{\Sigma}^p_{i+1}$ independently of k in this case. This is because Σ^b_i-IND is Σ^b_{i+1}, and therefore sequents $f(S)$ in the above proof still contains Σ^b_{i+1}-formulas even when the endformula is Σ^b_k.

The next Lemma is useful in proving Theorem 7.25, which implies the 'only if' directions of Theorem 7.22.

Lemma 7.24. *For every $k \geq 1$,*

$$\mathbf{FP}^{\mathbf{\Sigma}^p_k}_{||}[wit, O(1)] = \mathbf{FAC}^{\mathbf{0}^{\mathbf{\Sigma}^p_k}}_{||}[wit, O(1)]$$

Proof. This is proven analogously to Jenner and Torán's proof that $\mathbf{FP}^{\mathbf{NP}}_{||}$ is equal to $\mathbf{FAC}^{\mathbf{0NP}}$ in [JT95] and Buss and Hay's proof that $\mathbf{P}^{\mathbf{NP}}_{||}$ is the class of relations representable by $\Sigma^b_0(\Sigma^b_1)$-formulas [BH88].

We show $\mathbf{FP}^{\mathbf{\Sigma}^p_k}_{||}[wit, O(1)] \subseteq \mathbf{FAC}^{\mathbf{0}^{\mathbf{\Sigma}^p_k}}_{||}[wit, O(1)]$, since the $\supseteq$ is trivial. Let $k \geq 1$ be arbitrary. Let M be a polynomial-time Turing machine that, on x, asks c witness queries to

$A \in \mathbf{\Sigma}_k^p$ nonadaptively before halting with an output $y \in Q(x)$. Define M' to be the following machine. On input (x, i) with $i \in [0, c]$, M' first simulates M until M receives the answers to its witness queries. Upon receiving the answers, M' rejects the input if the number of the positive answers is not exactly i. Otherwise M' continues the simulation of M and produces some y with $y \in Q(x)$.

We describe an $\mathbf{AC^0}$ algorithm that solves $Q(x)$. Given x, it first computes witness queries $q(i)$ for each $i \in [0, c]$ of the following form: is there a computation of M' on (x, i)? All of these queries are answered positively, accompanied by a description of a corresponding computation of M'. Note that there is a unique value i such that the query $q(i)$ returns a computation that results in an output $y \in Q(x)$. $\mathbf{AC^0}$ can recognize such a computation and extract y from it. Finally, note that the queries $q(0), \ldots, q(c)$ are also $\mathbf{AC^0}$-computable given x. □

The following theorem implies the 'only if' directions of Theorem 7.22.

Theorem 7.25. *For every $k \geq 2$, every search problem $Q \in \mathbf{FP}^{\mathbf{\Sigma}_{k-1}^p}[wit, O(1)]$ is Σ_k^B-definable in $\mathbf{V}^0$.*

Proof. Fix $k \geq 2$. By Lemma 7.24 and Theorem 2.7, it suffices to show that every search problem Q in $\mathbf{FAC^0}^{\mathbf{\Sigma}_{k-1}^p}[wit, O(1)]$ is Σ_k^B-definable in $\mathbf{V}^0$. Let $\overline{\mathbf{V}}^0$ be the conservative extension of $\mathbf{V}^0$ obtained by adding, for every $\mathbf{AC^0}$-function, a function symbol for it and its defining axiom. We prove that Q is Σ_k^B-definable in $\overline{\mathbf{V}}^0$.

By definition, there exists an oracle $A \in \mathbf{\Sigma}_{k-1}^p$ and $\mathbf{AC^0}$-functions $H, G_1, \ldots, G_c$ such that the following hold: (i) for each $i \in [1, c]$, $G_i(X)$ computes a witness query to A; and (ii) if $Z_1, \ldots, Z_c$ are answers to the witness queries $G_1(X), \ldots, G_c(X)$, then

$$F(X) = H(X, Z_1, \ldots, Z_c).$$

Note that $\overline{\mathbf{V}}^0$ has a symbol for each of $H, G_1, \ldots, G_c$ and contains a defining axiom for it.

In order to simplify our presentation, we do not write out bounds of the bounded quantifiers in the formulas below. By a second-order version of Theorem 2.17, the oracle A is represented

by some Σ^B_{k-1}-formula $\exists W \phi(X, W)$ with $\phi(X, W) \in \Pi^B_{k-2}$. For each $i \in [1, c]$, define a formula $Correct_i(X, B, W)$ as follows:

$$Correct_i(X, B, W) =_{syn} [(|B| = 0 \wedge \forall A \neg \phi(G_i(X), A)) \vee (|B| = 1 \wedge \phi(G_i(X), W))].$$

$Correct_i(X, B, W)$ asserts that, for an input X, B and W collectively encode an answer to the witness query $G_i(X)$, where $|B|$ encodes the yes-no answer to $G_i(X)$, and W is a witness whenever the answer is a yes. It is not hard to show that

$$\overline{\mathbf{V}}^0 \vdash (\exists B)(\exists W) Correct_i(X, B, W)$$

for every $i \in [1, c]$; in fact, the above formula is logically valid.

Recall that $\langle B, W \rangle$ is an encoding of two strings B, W in one; see (7.3) on page 143. An answer Z_i to a witness query $G_i(X)$ can be encoded by $\langle B, W \rangle$ such that $Correct_i(X, B, W)$ holds. Then $\overline{\mathbf{V}}^0$ proves the following:

$$\begin{aligned}
&(\exists Y)(\exists Z_1 \ldots \exists Z_c)(\exists B_1, W_1 \ldots \exists B_c, W_c) \\
&\quad Z_1 = \langle B_1, W_1 \rangle \wedge \ldots \wedge Z_c = \langle B_c, W_c \rangle \\
&\quad \wedge Correct_1(X, B_1, W_1) \wedge \ldots \wedge Correct_c(X, B_c, W_c) \\
&\quad \wedge Y = H(X, Z_1, \ldots, Z_c)
\end{aligned}$$

The above formula is not Σ^B_k since it is not in strict form, but it is clear that $\overline{\mathbf{V}}^0$ proves a strict formula equivalent to it. This completes the proof that Q is Σ^B_k-definable in $\overline{\mathbf{V}}^0$. □

Chapter 8

Bounded Arithmetic and the Witnessing Problems for QPC

Cook in [Coo02] presented a translation of second-order bounded arithmetic formulas into polynomial-size QPC formulas. Using this translation, we prove that any bounded theorem of $\mathbf{V}^i$ and $\mathbf{TV}^i$ translates into families of valid QPC formulas with polynomial-size G_i^*- and G_i-proofs, respectively. We also prove a translation of bounded theorems of $\mathbf{VNC}^1$ into polynomial-size G_0^*-proofs. These translation theorems generalize similar results for S_2^i and T_2^i by Krajíček and Pudlák (Theorem 5.29), since our translation applies to *any* bounded theorem, as opposed to Σ_i^B-theorems. This generality stems from the simple syntax of Cook's second-order framework.

Based on these translation theorems, we address the question whether j-$RFN(G_i)$ is provable in T_2^i for $j \geq i+2$. We will show that the answer is negative, unless $\mathbf{PH}$ collapses.

8.1 Propositional Translations

In this section we give a complete definition of Cook's translation of bounded formulas over $\mathcal{L}_A^2$ to quantified propositional formulas [Coo02]. This translation is similar to the Krajíček-

Pudlák translation of the first-order language of bounded arithmetic (see [KP90] and [Kra95] sec 9.2), but Cook's second-order setting allows a much simpler translation.

First, we show how to translate Σ_0^B-formulas into quantifier-free QPC formulas. Let

$$\phi(x_1, \ldots, x_k, X_1, \ldots, X_l)$$

be a Σ_0^B-formula with all free variables displayed. The translation takes $k + l$ natural numbers $r_1, \ldots, r_k, n_1, \ldots, n_l$ as parameters and produces a Σ_0^q-formula $||\phi(\vec{x}, \vec{X})||[\vec{r}, \vec{n}]$ whose size is polynomial in $max\{\vec{r}, \vec{n}\}$. For $i \in [1, k]$, r_i is the value for the number free variable x_i. For each $i \in [1, l]$, we associate with string variable X_j the propositional variables $p_0^{X_j}, p_1^{X_j}, \ldots, p_{n_j}^{X_j}$, where $p_i^{X_j}$ is intended to mean $X_j(i)$. The translation has the property that, for every set $\vec{r}, \vec{n}$ of parameters, $\|\phi(\vec{x}, \vec{X})\|[\vec{r}, \vec{n}]$ is valid iff

$$\underline{\mathbb{N}} \models (\forall \vec{X})(|X_1| = n_1 \wedge \ldots \wedge X_l = n_l \supset \phi(\vec{r}, \vec{X})).$$

More generally, there is a one-one correspondence between truth assignments satisfying $\|\phi(\vec{x}, \vec{X})\|[\vec{r}, \vec{n}]$ and tuples of strings $\vec{X}$, with $|X_i| = n_i$, satisfying $\phi(\vec{r}, \vec{X})$.

We use the notation $val(t)$ for the numerical value to which t evaluates under the parameters $\vec{r}, \vec{n}$. Our usage of $val(t)$ is such that t contains no bound variables, thus its value only depends on the parameters $\vec{r}, \vec{n}$.

Let us write ϕ to denote $\phi(\vec{x}, \vec{X})$. The first step in defining $||\phi||[\vec{r}, \vec{n}]$ is to replace every atomic formula of the form $X = Y$ by its Σ_0^B definition, given by the RHS of the extensionality axiom SE. After this is done, we define $||\phi||[\vec{r}, \vec{n}]$ by structural induction on ϕ.

The base case is when ϕ is atomic. Four cases arise. First, if ϕ is one of the constants T or F then $||\phi||[] = \phi(\vec{X})$; note that there is no parameter in this case. Second, if ϕ is $t(\vec{x}, \vec{X}) = u(\vec{x}, \vec{X})$, then $||\phi||[\vec{r}, \vec{n}] = \mathsf{T}$ if $val(t(\vec{x}, \vec{X})) = val(u(\vec{x}, \vec{X}))$, and $||\phi||[\vec{r}, \vec{n}] = \mathsf{F}$ otherwise. The third case is when $\phi(\vec{X})$ is $t(|\vec{X}|) \leq u(|\vec{X}|)$, and this is handled similarly to the second case. The forth case is when ϕ is $X_j(t(\vec{x}, \vec{X}))$. Then we set $j = val(t(\vec{x}, \vec{X}))$ and

define

$$||\phi||[\vec{r},\vec{n}] = \begin{cases} p_j^{X_i} & \text{if } j < n_i - 1 \\ \mathsf{T} & \text{if } j = n_i - 1 \\ \mathsf{F} & \text{if } j > n_i - 1 \end{cases}$$

For the induction step, $\phi(\vec{x},\vec{X})$ is built from smaller formulas using a propositional connective $\wedge, \vee, \neg$, or a bounded quantifier. For $\wedge, \vee, \neg$ we make the obvious definition; for example

$$\|\psi(\vec{x},\vec{X}) \wedge \eta(\vec{x},\vec{X})\|[\vec{r},\vec{n}] = (\|\psi(\vec{x},\vec{X})\|[\vec{r},\vec{n}]) \wedge (\|\eta(\vec{x},\vec{X})\|[\vec{r},\vec{n}])$$

For the case of bounded number quantifiers, we define

$$\begin{aligned} \|(\exists y)[y \leq t \wedge \psi(y,\vec{x},\vec{X})]\|[\vec{r},\vec{n}] &= \bigvee_{r_b=0}^{val(t)} \|\psi(b,\vec{x},\vec{X})\|[r_b,\vec{r},\vec{n}] \\ \|(\forall y)[y \leq t \supset \psi(y,\vec{x},\vec{X})\|[\vec{r},\vec{n}] &= \bigwedge_{r_b=0}^{val(t)} \|\psi(b,\vec{x},\vec{X})\|[r_b,\vec{r},\vec{n}] \end{aligned}$$

Note that the bounding term t contains variables from $\vec{x},\vec{X}$ only, thus $val(t)$ depends only on $\vec{r},\vec{n}$. This completes the QPC translation of Σ_0^B-formulas.

For the translation of Σ_i^B-formulas with $i \geq 1$, it suffices to describe how to handle bounded string quantifiers, which is done as follows:

$$\begin{aligned} \|(\exists Y)[|Y| \leq t\psi(\vec{x},Y,\vec{X})\|[\vec{r},\vec{n}] &= \exists p_0^Y ... \exists p_{val(t)}^Y \bigvee_{n_Y=0}^{val(t)} \|\psi(\vec{x},Y,\vec{X})\|[\vec{r},n_Y,\vec{n}] \\ \|(\forall Y)[|Y| \leq t\psi(\vec{x},Y,\vec{X})\|[\vec{r},\vec{n}] &= \forall p_0^Y ... \forall p_{val(t)}^Y \bigwedge_{n_Y=0}^{val(t)} \|\psi(\vec{x},Y,\vec{X})\|[\vec{r},n_Y,\vec{n}] \end{aligned}$$

Each variable p_i^Y above is a metavariable for an appropriate x-variable; this way, we meet our free-bound variable convention.

This completes the definition of the QPC translation of bounded formulas over $\mathcal{L}_A^2$. Notice that Σ_i^B formulas translate to families of Σ_i^q formulas. Moreover, unlike the QPC translation of first-order bounded arithmetic formulas, Σ_0^B-formulas are translated into quantifier-free QPC

formulas (i.e., propositional formulas). However, for $i \geq 1$, the Σ_i^q-formulas obtained by translating Σ_i^B-formulas are in general not prenex.

This translation allows us to state a number of results, which can be inferred from the literature [KP90, Kra95, Coo02, Coo04], connecting a theory T over $\mathcal{L}_A^2$ with a corresponding QPC proof system. For example, a Σ_0^B-theorem of $\mathbf{V}^0$ translates to a tautology family with polynomial size bounded-depth Frege proofs. Second-order analogs of the G_i simulation theorems for S_2^i and T_2^i are presented as Theorem 8.2 below.

Note that, for $\phi \in \Sigma_i^B$ with $i \geq 1$, the QPC formula $||\phi||[\vec{r}, \vec{n}]$ is not prenex in general. We define $\langle\langle\phi\rangle\rangle[\vec{r}, \vec{n}]$ to be the prenexification of $||\phi||[\vec{r}, \vec{n}]$ obtained simply by moving all the quantifiers to the left while maintaining their order.

We define the semi-implicit bounded formulas as follows:

Definition 8.1. *Let $i \geq 1$. A* semi-implicit Σ_i^B*-formula is the formula obtained from a Σ_i^B-formula by pulling all of its string quantifiers to the front.*

For example, a Σ_2^B-formula

$$(\exists Y \leq t(\vec{x}, \vec{X}))(\forall Z \leq s(\vec{x}, \vec{X}, Y))\psi(\vec{x}, \vec{X}, Y, Z),$$

where $\psi(\vec{x}, \vec{X}, Y, Z) \in \Sigma_0^B$, gives rise to the following semi-implicit Σ_2^B-formula:

$$(\exists Y)(\forall Z)[|Y| \leq t(\vec{x}, \vec{X}) \wedge (|Z| \leq s(\vec{x}, \vec{X}, Y) \supset \psi(\vec{x}, \vec{X}, Y, Y, Z))] \tag{8.1}$$

Note that every bounded formula gives rise to a unique semi-implicitly bounded formula.

Let $\phi \in \Sigma_i^B$ for some i and ϕ' be the corresponding semi-implicit Σ_i^B-formula. It is not hard to define a direct translation of ϕ' into prenex Σ_i^q-formulas $\langle\langle\phi\rangle\rangle[\vec{r}, \vec{n}]$. Since such a translation is similar to the above, we only give an example. Let ϕ' be the semi-implicit Σ_2^B-formula in (8.1). Define the term s' to be $s(t(\vec{x}, \vec{X}), \vec{x}, \vec{X})$. Then $\langle\langle\phi\rangle\rangle[\vec{r}, \vec{n}]$ is as follows:

$$\exists p_0^Y \ldots \exists p_{val(t)}^Y \forall p_0^Z \ldots \forall p_{val(s')}^Z \bigvee_{n_Y=0}^{val(t)} \bigwedge_{n_Z=0}^{val(s)} \Big(||\psi(\vec{x}, \vec{X}, Y, Z)||[\vec{r}, \vec{n}, n_Y, n_Z] \Big)$$

Note that $val(t)$ and $val(s')$ depend only on $\vec{r}, \vec{n}$, while $val(s)$ depends on an additional parameter n_Y.

8.2 The QPC Translation Theorems for Second-Order Theories

The following result gives a second-order setting to Krajíček and Pudlák's result [KP90] showing G_i simulates T_2^i, and to Krajíček's result [Kra95] showing G_i^* simulates S_2^i. Our modified definitions of G_i and G_i^* allow us to state the result for arbitrary bounded theorems of $\mathbf{TV}^i$ and $\mathbf{V}^i$, as opposed to just Σ_i^B theorems. This is a straightforward generalization of Cook's result that the Σ_1^B-theorems of $\mathbf{V}^1$ translate into polynomial-size G_1^*-proofs [Coo02].

Theorem 8.2. *For every $i \geq 1$, if $\phi(\vec{x}, \vec{X})$ is a bounded theorem of $\mathbf{V}^i$, then the family $\|\phi(\vec{x}, \vec{X})\|[\vec{r}, \vec{n}]$ has G_i^* proofs which can be computed in time polynomial in $\vec{r}, \vec{n}$. The same is true for $\mathbf{TV}^i$ and G_i.*

Moreover, the above assertion holds even if we use $\langle\langle\phi(\vec{x}, \vec{X})\rangle\rangle[\vec{r}, \vec{n}]$ in place of $\|\phi(\vec{x}, \vec{X})\|[\vec{r}, \vec{n}]$.

Proof. (Sketch) We first sketch a proof for the translation of a bounded theorem ϕ of $\mathbf{V}^i$ into polynomial-size G_i^*-proofs of $||\phi||[\vec{r}, \vec{n}]$. The proof is essentially identical to Cook's proof in [Coo02] of the QPC translation of Σ_1^B-theorems of $\mathbf{V}^1$; we reproduce his argument here for completeness.

Let $i \geq 1$ and $\phi(\vec{x}, \vec{X})$ be a bounded theorem of $\mathbf{V}^i$. Then, there is a tree-like $LK^2 + \Sigma_i^B$-IND proof π of $\phi(\vec{x}, \vec{X})$ satisfying the conditions of Theorem 7.21; in particular, all formulas in π are bounded. Assume that π contains m sequents $S_1, \ldots, S_m$, where S_m is the endsequent $\rightarrow \phi(\vec{x}, \vec{X})$. Let $S'_m(\vec{r}, \vec{n})$ be the QPC sequent obtained by transforming each formula $\psi(\vec{x}, \vec{X})$ to $\|\psi(\vec{x}, \vec{X})\|[\vec{r}, \vec{n}]$. We prove the following assertion by induction on m: there exists a polynomial p such that, for every set of parameters $(\vec{r}, \vec{n})$, the QPC sequent $S'_m(\vec{r}, \vec{n})$ has G_i^*-proofs of size at most $p(max\{\vec{r}, \vec{n}\})$.

The base case is when S_m is an initial sequent, that is, S_m is a logical axiom, an equality axiom, an axiom in 2-$BASIC$, or Σ_0^B-COMP. It is not hard to see that $S'_m(\vec{r}, \vec{n})$ consists of Σ_1^q-formulas only and it has short G_1^*-proofs. For the inductive step, we have a number of cases, one for each inference rule that derives S_m from preceding sequents. The cases for

structural, propositional, and cut rules are easy: just apply the same rule in G_i^*. Note that every cut formula in π is Σ_i^B and G_i^* can cut its QPC translation. The only nontrivial cases are for the quantifier rules and the Σ_i^B-IND rule. Below we show how to handle String $\exists$-right and String $\forall$-right; the other quantifier rules are handled analogously.

Suppose that S_m is derived from S_i by String $\exists$-right rule in the following way:

$$\frac{[S_i] \quad \Gamma \to \Delta, |X| \leq t \wedge A}{[S_m] \quad \Gamma \to \Delta, (\exists X)[|X| \leq t \wedge A]}$$

For simplicity, we ignore all free variables of S_m other than X. Then $S'_m(n_X)$ is of the form

$$||\Gamma||[n_X] \to ||\Delta||[n_X], \exists p_0^X \ldots \exists p_{val(t)}^X \bigvee_{n_X=0}^{val(t)} ||A||[n_X].$$

Then, again ignoring the free variables other than X, $S'_i(n_X)$ is of the form

$$||\Gamma|| \to ||\Delta||, || \, |X| \leq t \wedge A||[n_X]$$

If $n_X > val(t)$, then the auxiliary formula $|| \, |X| \leq t \wedge A||[n_X]$ is simply F and therefore $S'_m(n_X)$ follows from $S'_i(n_X)$ by weakening. If $n_X \leq val(t)$, then $S'_m(n_X)$ is derived from $S'_i(n_X)$ by weakening-right, $\vee$-right, and then $\exists$-right.

Next, suppose that S_m is derived from S_i by String $\forall$-right:

$$\frac{[S_i] \quad \Gamma \to \Delta, |X| \leq t \supset A}{[S_m] \quad \Gamma \to \Delta, (\forall X)[|X| \leq t \supset A]}$$

Then, for each $n_X \leq val(t)$, we have a proof of $S'_i(n_X)$, which is of the form

$$||\Gamma|| \to ||\Delta||, || \, |X| \leq t \supset A||[n_X].$$

By applying $\wedge$-right $val(t) + 1$ times, we derive

$$||\Gamma|| \to ||\Delta||, \bigwedge_{n_X=0}^{val(t)} \Big(|| \, |X| \leq t \supset A||[n_X] \Big),$$

from which $S'_m(n_X)$ follows by $val(t) + 1$ applications of $\forall$-right.

If S_i is derived by a Σ_i^B-IND step, then this step is simulated in a straightforward way by a polynomial-size sequence of cuts in the G_i^* proof, with Σ_i^q cut formulas. This concludes the proof of the first statement for $\mathbf{V}^i$.

For $\mathbf{TV}^i$, the translation is identical except for a Σ_i^B-String-IND step, for which a straightforward translation by a sequence of cuts would result in exponentially many cuts, so instead we use a doubling chain of implications whose intuitive meaning is $\|\psi(X)\| \to \|\psi(X+2^i)\|$, where $+$ is binary addition. This G_i proof is not tree-like, and uses the fact that for $i \geq 1$ a substitution rule for Σ_i^q formulas can be added to G_i with only a polynomial increase in power. This is carried out by Krajíček and Pudlák in their proof of Theorem 5.29 in [KP90].

If ϕ is a bounded theorem of $\mathbf{V}^i$, then the corresponding semi-implicitly bounded formula ϕ' is also a theorem of $\mathbf{V}^i$. The polynomial-size proofs of $\langle\langle\phi\rangle\rangle[\vec{r},\vec{n}]$ are obtained by translating a $LK^2+\Sigma_i^B$-IND proof of ϕ' into G_i^*-proofs in a way similar to the above, using the translation of semi-implicitly bounded formulas into prenex QPC formulas. Similarly for $\mathbf{TV}^i$ and G_i. □

The next result shows that $\mathbf{VNC}^1$ proofs of bounded formulas translate into polynomial size families of G_0^* proofs. This is analogous to Arai's [Ara00] theorem showing that **AID** proofs of Σ_0^b formulas translate into polynomial size families of Frege proofs. Our result is more general, because it applies to all bounded theorems and not just those in Σ_0^B, and simpler, because of our second-order setting.

Theorem 8.3. *If $\phi(\vec{x},\vec{X})$ is a bounded theorem of $\mathbf{VNC}^1$, then the family $\|\phi(\vec{r},\vec{X})\|[\vec{n}]$ has G_0^* proofs of size polynomial in $\vec{r},\vec{n}$, and can be computed by an $\mathbf{NC}^1$ function of $\vec{r},\vec{n}$.*

The above statement holds even if we use $\langle\langle\phi(\vec{x},\vec{X})\rangle\rangle[\vec{r},\vec{n}]$ instead of $\|\phi(\vec{x},\vec{X})\|[\vec{r},\vec{n}]$

Proof. Apply Theorem 7.20 and let π be a tree-like LK^2 proof of a bounded formula $\phi(\vec{x},\vec{X})$ from the equality axioms and the axioms of $\mathbf{VNC}^1$. All cut formulas of π are substitution instances of $\mathbf{VNC}^1$ axioms, and hence are Σ_1^B, and therefore all formulas in π are bounded.

Let parameters $(\vec{r},\vec{n})$ be given. By applying the method of the proof of Theorem 8.2, we first construct a G_1^*-proof π' such that (i) its endsequent is $||\phi(\vec{x},\vec{X})||[\vec{r},\vec{n}]$, and (ii) QPC translations of nonlogical axioms of $\mathbf{VNC}^1$ may appear in π' as initial sequents. Every cut formula of π' is a QPC translation of a cut formula of π, and therefore every cut formula of π' that is not quantifier-free is a QPC translation of either Σ_0^B-COMP or Σ_0^B-$TreeRec$. It remains

to transform π' to a G_0^* proof with no nonlogical axioms.

All axioms of $\mathbf{VNC}^1$ are Σ_0^B except Σ_0^B-COMP and Σ_0^B-$TreeRec$. Each of the Σ_0^B axioms either translates to T, or translates to a valid Σ_0^q-formula with a trivial G_0^* proof. Each of the Σ_1^B axioms of $\mathbf{VNC}^1$ has the form $(\exists Y \leq t)\psi(\vec{x}, \vec{X}, Y)$, where ψ is Σ_0^B. Further, given $(\vec{r}, \vec{n})$, it is easy to find quantifier-free formulas $C_0, ..., C_m$ witnessing the existential quantifiers $\exists x_0^Y ... \exists x_m^Y$ in its translation

$$D \equiv \|\exists Y \leq t\psi(\vec{x}, \vec{X}, Y)\|[\vec{r}, \vec{n}] \tag{8.2}$$

where $m = val(t)$. In fact, if these existential quantifiers are removed from D and each variable x_i^Y is replaced by C_i, the result is a valid Σ_0^q formula D' with a G_0^* proof of size polynomial in $max\{\vec{r}, \vec{n}\}$.

Now consider an uppermost instance of the cut rule in π', with cut formula D from (8.2). We change this instance to an instance in which the cut formula is D' instead of D, but the conclusion is the same. The upper-right sequent of the original instance has D in the succedent: just replace D by D' after deriving D' with a G_0^* proof. The upper-left sequent has D in the antecedent: modify the derivation of this sequent by replacing every eigenvariable p_i^Y in an exists-left rule by C_i throughout the derivation, and remove all $\exists$-left rules used to derive D. Thus we convert

$$\dfrac{\begin{array}{cc}\vdots & \vdots \\ D, \Gamma \to \Delta & \Gamma \to \Delta, D\end{array}}{\Gamma \to \Delta}$$

to

$$\dfrac{\begin{array}{cc} & \vdots \\ \vdots & \to D' \\ D', \Gamma \to \Delta & \Gamma \to \Delta, D'\end{array}}{\Gamma \to \Delta}$$

The result is a G_0^* derivation of the same sequent.

Continue replacing each Σ_1^q cut formula in the proof π' by a Σ_0^q cut formula, in the same way. The result is a G_0^*-proof of $||\phi(\vec{x}, Xvec)||[\vec{r}, \vec{n}]$. □

Since a G_0 proof of a quantifier-free formula is a PK-proof, we obtain the following:

Corollary 8.4. *If $\phi(\vec{x}, \vec{X})$ is a Σ_0^B-theorem of* $\mathbf{VNC}^1$*, then the family $\|\phi(\vec{x}, \vec{X})[\vec{r}, \vec{n}]$ has PK-proofs of size polynomial in $\vec{r}, \vec{n}$, and these can be computed in* $\mathbf{FNC}^1$.

8.3 Implications for QPC Witnessing and Reflection Principles

In this subsection we present some consequences of the QPC translation thereoms for the complexity of the QPC witnessing theorems. First, we state and prove the *only-if* direction of Theorem 7.15.

Theorem 8.5. *If a function (string or number) is Σ_1^B-definable in* $\mathbf{VNC}^1$*, then it is in* $\mathbf{FNC}^1$.

Proof. Let F br a string function Σ_1^B-definable in $\mathbf{VNC}^1$. Then, there is a Σ_1^B-formula $\phi(\vec{x}, \vec{X}, Y)$ representing the graph of F such that $\mathbf{VNC}^1$ proves

$$\mathbf{VNC}^1 \vdash (\exists Y)\phi(\vec{x}, \vec{X}, Y).$$

By Theorem 8.3, there exists an $\mathbf{NC}^1$-function P such that, given the number arguments $\vec{r}$ and string arguments $\vec{X}$ of F, $P(\vec{r}, \vec{X})$ is a pair $(\pi, \vec{v})$, where π is a G_0^*-proof of $\langle\langle(\exists Y)\phi(\vec{x}, \vec{X}, Y)\rangle\rangle[\vec{r}, \vec{n}]$, and $\vec{v}$ is a truth assignment encoding the input strings $\vec{X}$. By running the $\mathbf{NC}^1$ algorithm for $Witness[G_0^*, \Sigma_1^q]$ on $P(\vec{r}, \vec{n}, \vec{X})$, a witness to Y is computed in $\mathbf{NC}^1$. □

Theorem 8.6. *The following hardness results hold with respect to polynomial-time many-one reductions:*

(i) For every $k \geq 2$, $Witness[G_0^*, \Sigma_k^q]$ is hard for $\mathbf{FP}^{\mathbf{\Sigma}_{k-1}^p}[wit, O(1)]$.

(ii) For every $i \geq 1$, $Witness[G_i^*, \Sigma_{i+1}^q]$ is hard for $\mathbf{FP}_{||}^{\mathbf{\Sigma}_i^p}[wit]$.

(iii) For every $i \geq 1$, $Witness[G_i, \Sigma_{i+1}^q]$ is hard for $\mathbf{FP}^{\mathbf{\Sigma}_i^p}$.

(iv) Let $i \geq 1$ be arbitrary and let H_i denote either G_i or G_i^*. Then, for every $k \geq i + 2$, $Witness[H_i, \Sigma_k^q]$ is hard for $\mathbf{FP}^{\mathbf{\Sigma}_{k-1}^p}[wit, O(1)]$.

Proof. For (i), every $Q \in \mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(1)]$ is Σ^B_k-definable in $\mathbf{VNC}^1$ by Corollary 7.22, and by Theorem 8.3 the Σ^B_k-formula defining Q gives rise to a family of polynomial-size G^*_0-proofs π_n. Given an instance of Q, a solution can be found by solving $Witness[G^*_0, \Sigma^q_k]$ on $(\pi_n, \vec{v})$, where n and $\vec{v}$ depend on the instance of Q.

The proofs for the other cases are completely analogous, using Theorems 2.23, 2.24 and 8.2. □

Note that (ii) of the above theorem is the hardness direction of Theorem 6.9. We do not know whether (iii) could be strengthened to a completeness. We show below that (i) and (iv) are unlikely to be strengthened to be the completeness for $\mathbf{FP}^{\mathbf{\Sigma}^p_{j-1}}[wit, O(1)]$. This result is based on the following fact:

Theorem 8.7. *Let $i \geq 1$ and k, k' be such that $1 \leq k < k'$. If $\mathbf{FP}^{\Sigma^p_i}[wit, k] = \mathbf{FP}^{\Sigma^p_i}[wit, k']$, then $\mathbf{PH}$ collapses to $\mathbf{P}^{\Sigma^p_{i+1}}[O(\log n)]$.*

Proof. Assume that

$$\mathbf{FP}^{\Sigma^p_i}[wit, k] = \mathbf{FP}^{\Sigma^p_i}[wit, k']$$

for some $i \geq 1$ and $1 \leq k < k'$. By (iii) of Theorem 2.7, it follows that

$$\mathbf{P}^{\Sigma^p_i}[k] = \mathbf{P}^{\Sigma^p_i}[k'],$$

which implies the collapse of $\mathbf{PH}$ to $\mathbf{P}^{\Sigma^p_{i+1}}[O(\log n)]$ by Kadin's result [Kad88]. □

Theorem 8.8. *Let $i \geq 0$ and $k \geq i + 2$. If $Witness[G_i, \Sigma^q_k] \in \mathbf{FP}^{\Sigma^p_{k-1}}[wit, O(1)]$, then $\mathbf{PH}$ collapses to $\mathbf{P}^{\mathbf{\Sigma}^p_k}[O(\log n)]$. Similarly for $Witness[G^*_i, \Sigma^q_k]$.*

Proof. Let $i \geq 0$ and $k \geq i + 2$. If $Witness[G_i, \Sigma^q_k] \in \mathbf{FP}^{\Sigma^p_k}[wit, O(1)]$, then there is some $c \in \mathbb{N}$ such that $Witness[G_i, \Sigma^q_k] \in \mathbf{FP}^{\Sigma^p_{k-1}}[wit, c]$, and by (iv) of Theorem 8.6, $\mathbf{FP}^{\Sigma^p_{k-1}}[wit, O(1)]$ collapses to $\mathbf{FP}^{\Sigma^p_{k-1}}[wit, c]$. A collapse of $\mathbf{PH}$ follows by Theorem 8.7.

□

Corollary 8.9. *(i) Let $i \geq 1$ and $k \geq i+2$. If k-$RFN(G_i)$ is provable in T^i_2, then* $\mathbf{PH}$ *collapses to $\mathbf{P}^{\mathbf{\Sigma}^p_k}[O(\log n)]$. Similarly for k-$RFN(G^*_i)$ and S^i_2. (ii) Let $k \geq 2$. If* $\mathbf{TV}^0$ *proves k-$RFN(G-0^*)$, then* $\mathbf{PH}$ *collapses to $\mathbf{P}^{\mathbf{\Sigma}^p_k}[O(\log n)]$.*

Proof. Let $i \geq 1$ and $k \geq i+2$. Suppose that T^i_2 proves k-$RFN(G_i)$, which means that $Witness[G_i, \Sigma^q_k]$ is Σ^b_k-defined in T^i_2. By Theorem 2.24, $Witness[G_i, \Sigma^q_k] \in \mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(1)]$ follows, and $\mathbf{PH}$ collapses because of Theorem 8.8. Similarly for S^i_2 and G^*_i, and $\mathbf{TV}^0$ and G^*_0. □

Chapter 9

Concluding Remarks for Part II

Throughout Part II we presented various results on the complexity of QPC witnessing problems. Since we consider these problems for various parameters, we summarize all the known results on the complexity of the QPC witnessing theorem that we have discussed in this dissertation. Also see Table 9.2 on page 173.

- For both G_0 and G_0^*, the Σ_1^q-witnessing problems are complete for $\mathbf{FNC}^1$ under many-one $\mathbf{AC^0}$-reduction (Theorem 6.3).
- For both G_0 and G_0^*, for every $k \geq 2$, the Σ_k^q-witnessing problems are in $\mathbf{FP}^{\mathbf{\Sigma}_{k-1}^p}[wit, O(\log n)]$ (Theorem 6.11).
- For both G_0 and G_0^*, for every $k \geq 2$, the Σ_k^q-witnessing problems are hard for $\mathbf{FP}^{\mathbf{\Sigma}_{k-1}^p}[wit, O(1)]$ under many-one $\mathbf{AC^0}$-reduction (Theorem 8.6). If these problems are complete for these classes, then $\mathbf{PH}$ collapses (Theorem 8.8).

Next, we list the results regarding G_i^* and G_i for an arbitrary $i \geq 1$.

- The Σ_i^q-witnessing problem for G_i^* is complete for $\mathbf{FP}^{\mathbf{\Sigma}_{i-1}^p}$ (Theorem 6.2).
- The Σ_i^q-witnessing problem for G_i is complete for $C(\mathbf{PLS})^{\mathbf{\Sigma}_{i-1}^p}$ (Theorem 6.2).

Note that the above two results match the results on the complexity of $\hat{\Sigma}^b_i$-definable search problems of S^i_2 and T^i_2 (Theorem 2.22).

- The Σ^q_{i+1}-witnessing problem for G^*_i is complete for $\mathbf{FP}^{\mathbf{\Sigma}^p_i}[wit, O(\log n)]$ (Theorem 6.9).

The above result also matches the complexity of the $\hat{\Sigma}^b_{i+1}$-definable search problems of S^i_2 (Theorem 2.23).

- The Σ^q_{i+1}-witnessing problem for G_i is hard for $\mathbf{FP}^{\mathbf{\Sigma}^p_i}$ (Theorem 8.6).

We conjecture that the above problem is indeed complete for $\mathbf{FP}^{\mathbf{\Sigma}^p_i}$, matching the complexity of the $\hat{\Sigma}^b_{i+1}$-definable search problems of T^i_2.

- For every $k \geq i+2$, the Σ^q_k-witnessing problem for both G^*_i and G_i are in $\mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(\log n)]$. (Theorem 6.11)

- For every $k \geq i+2$, the Σ^q_k-witnessing problems for both G^*_i and G_i are hard for $\mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(1)]$. (Theorem 8.6)

- For every $i \geq 0$ and $k \geq i+2$, if the Σ^q_k-witnessing problem for either G_i or G^*_i is in $\mathbf{FP}^{\mathbf{\Sigma}^p_{k-1}}[wit, O(1)]$, then $\mathbf{PH}$ collapses to $\mathbf{P}^{\mathbf{\Sigma}^p_k}[O(\log n)]$ (Theorem 8.8).

The last item above shows that the complexity of $Witness[G_i, \Sigma^q_k]$ does *not* match the complexity of $\hat{\Sigma}^b_k$-definable search problems (Theorem 2.24), and similarly for G^*_i and S^i_2.

The following is open: for $i \geq 2$ and $k < i$, find a complexity class for which the Σ^q_k-witnessing problem for G_i (or G^*_i) is complete. Note that there is no analogous result for the $\hat{\Sigma}^b_k$-definable search problems of T^i_2 (or S^i_2).

Let H be a QPC proof system and let $j \geq 1$. The Σ^q_j-witnessing problem for H is $\hat{\Sigma}^b_j$-definable in theory T if T proves $j\text{-}RFN(H)$. Below we summarize the open problems and known facts on the provability in bounded arithmetic of the reflection principles of various QPC sequent calculus systems. Also see Table 9.1.

- We conjecture that $\mathbf{VNC}^1$ proves both 1-$RFN(G_0)$ and 1-$RFN(G_0^*)$.
- If $\mathbf{TV}^0$ proves j-$RFN(G_0^*)$ for any $j \geq 2$, then $\mathbf{PH}$ collapses (Corollary 8.9). Note that $\mathbf{VNC}^1 \subseteq \mathbf{TV}^0$.

The following hold for any $i \geq 1$.

- S_2^i proves i-$RFN(G_i^*)$. T_2^i proves i-$RFN(G_i)$. ([KP90, Kra95], Theorem 5.32)
- We do not know whether S_2^i proves $(i+1)$-$RFN(G_i^*)$. It is also unknown whether T_2^i proves $(i+1)$-$RFN(G_i)$.
- If S_2^i proves j-$RFN(G_i^*)$ for any $j \geq i+2$, then $\mathbf{PH}$ collapses. Similarly for T_2^i and G_i. (Corollary 8.9)

One interesting open problem is to decide whether T_2^i proves $(i+1)$-$RFN(G_i)$. The positive answer would follow via the Σ_{i+1}^b-conservativity of S_2^{i+1} over T_2^i (Theorem 2.20) if S_2^{i+1} proves $(i+1)$-$RFN(G_i)$. However, we do not know whether S_2^{i+1} indeed proves $(i+1)$-$RFN(G_i)$. In fact, we do not know how to prove a weaker assertion that G_{i+1}^* p-simulates G_i w.r.t. Σ_{i+1}^q. (This assertion is a consequence of the provability of $(i+1)$-$RFN(G_i)$ in S_2^{i+1} via Theorem 5.33.) A p-simulation of G_i by G_{i+1}^* w.r.t. Σ_i^q is known; see Corollary 5.34.

5-RFN	No$^{(*)}$	No$^{(*)}$	No$^{(*)}$	No$^{(*)}$	No$^{(*)}$	No$^{(*)}$	No$^{(*)}$
4-RFN	No$^{(*)}$	No$^{(*)}$	No$^{(*)}$	No$^{(*)}$	No$^{(*)}$	??	??
3-RFN	No$^{(*)}$	No$^{(*)}$	No$^{(*)}$	??	??	Yes	Yes
2-RFN	No$^{(*)}$	??	??	Yes	Yes	Yes	Yes
1-RFN	??	Yes	Yes	Yes	Yes	Yes	Yes
	$\langle \mathbf{VNC}^1, G_0 \rangle$	$\langle S_2^1, G_1^* \rangle$	$\langle T_2^1, G_1 \rangle$	$\langle S_2^2, G_2^* \rangle$	$\langle T_2^2, G_2 \rangle$	$\langle S_2^3, G_3^* \rangle$	$\langle T_2^3, G_3 \rangle$

Table 9.1: The provability of the reflection principles in bounded arithmetic. A cell at column $\langle T, H \rangle$ and row j indicates whether j-$RFN(H)$ is provable in T. Note that all the negative answers in this table are conditional; that is, the answer is negative unless $\mathbf{PH}$ collapses. We do not have answers for the cells containing '??'; we conjecture that the correct answers for these cells are 'Yes'.

Σ_5^q	$(\spadesuit_5)$	$(\spadesuit_5)$	$(\spadesuit_5)$	$(\spadesuit_5)$	$(\spadesuit_5)$	$(\spadesuit_5)$	$(\spadesuit_5)$
Σ_4^q	$(\spadesuit_4)$	$(\spadesuit_4)$	$(\spadesuit_4)$	$(\spadesuit_4)$	$(\spadesuit_4)$	$\mathbf{FP}^{\mathbf{\Sigma}_3^p}[wit, O(\log n)]$	$(\diamondsuit)$ $\mathbf{FP}^{\mathbf{\Sigma}_3^q}$
Σ_3^q	$(\spadesuit_3)$	$(\spadesuit_3)$	$(\spadesuit_3)$	$\mathbf{FP}^{\mathbf{\Sigma}_2^p}[wit, O(\log n)]$	$(\diamondsuit)$ $\mathbf{FP}^{\mathbf{\Sigma}_2^p}$	$\mathbf{FP}^{\mathbf{\Sigma}_2^q}$	$C(\mathbf{PLS})^{\mathbf{\Sigma}_2^p}$
Σ_2^q	$(\spadesuit_2)$	$\mathbf{FP}^{\mathbf{NP}}[wit, O(\log n)]$	$(\diamondsuit)$ $\mathbf{FP}^{\mathbf{NP}}$	$\mathbf{FP}^{\mathbf{NP}}$	$C(\mathbf{PLS})^{\mathbf{NP}}$	??	??
Σ_1^q	$\mathbf{FNC}^1$	$\mathbf{FP}$	$C(\mathbf{PLS})$	??	??	??	??
	G_0^*, G_0	G_1^*	G_1	G_2^*	G_2	G_3^*	G_3

Table 9.2: The complexity of $Witness[H, \Sigma_j^q]$ for $H \in \{G_i^*, G_i : 0 \leq i \leq 3\}$ and $j \in \{1, 2, 3, 4\}$. If a cell is marked with '$\diamondsuit$', then only the hardness is proven in this dissertation, and we conjecture that there is a matching upper bound. For every cell marked with '$\spadesuit_j$', the following holds: (i) the corresponding witnessing problem W is in $\mathbf{FP}^{\mathbf{\Sigma}_{j-1}^p}[wit, O(\log n)]$ and it is hard for $\mathbf{FP}^{\mathbf{\Sigma}_{j-1}^p}[wit, O(1)]$; and if W is in the latter class, then $\mathbf{PH}$ collapses.

Bibliography

[Aar04] S. Aaronson. Lower bounds for local search by quantum arguments. In *Proceedings of ACM STOC 2004*, 2004.

[All99] E. Allender. The permanent requires large uniform threshold circuits. *Chicago Journal of Theoretical Computer Science*, 1999.

[Ara00] T. Arai. A bounded arithmetic theory AID for Frege systems. *Annals of Pure and Applied Logic*, 103:155–199, 2000.

[BB03] A. Beckmann and S. R. Buss. Separation results for the size of constant-depth propositional proofs. *submitted*, 2003.

[BC96] S. Buss and P. Clote. Cutting planes, connectivity, and threshold logic. *Archive for Mathematical Logic*, 35:33–62, 1996.

[BCE+98] P. Beame, S. A. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi. The relative complexity of NP search problems. *Journal of Computer and System Sciences*, 57:3–19, 1998.

[BGS75] T. P. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4:431–442, 1975.

[BH88] S. R. Buss and L. Hay. On Truth-Table Reducibility to SAT and the Difference Hierarchy over NP. In *Proceedings, Structure in Complexity Theory, Third Annual Conference*, pages 224–233. IEEE computer Society Press, 1988.

[BI87] Manuel Blum and Russell Impagliazzo. Generic oracles and oracle classes. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (FOCS'97)*, pages 118–126, 1987.

[BIS90] D. A. M. Barrington, N. Immerman, and H. Straubing. On Uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[BK94] S. R. Buss and J. Krajicek. An application of Boolean complexity to separation problems in bounded arithmetic. *Proceedings of the London Mathematical Society*, 69:1–27, 1994.

[BKT93] S. R. Buss, J. Krajicek, and G. Takeuti. Provably total functions in bounded arithmetic theories R^i_3, U^i_2 and V^i_2. In P. Clote and J. Krajicek, editors, *Arithmetic, Proof Theory, and Computational Complexity*, pages 116–161. Oxford University Press, 1993.

[BOM04] J. Buresh-Oppenheim and T. Morioka. Relativized NP search problems and propositional proof systems. In *Proceedings of the 19th IEEE Conference on Computational Complexity (CCC'04)*, pages 54–67, 2004.

[BP96] P. Beame and T. Pitassi. An exponential separation between the parity principle and the pigeonhole principle. *Annals of Pure and Applied Logic*, 80:197–222, 1996.

[BP98] P. Beame and T. Pitassi. Propositional proof complexity: Past, present, and future. *Bulletin of the European Association for Theoretical Computer Science*, 65:66–89, 1998. The Computational Complexity Column (ed. E. Allender).

[BST03] D. Le Berre, L. Simon, and A. Tacchella. Challenges in the QBF arena: the SAT'03 evaluation of QBF solvers. In *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT2003)*, volume 2919 of *Lecture Notes in Computer Science*, pages 468–485. Springer-Verlag, 2003.

[Bus86] S. R. Buss. *Bounded Arithmetic*. Bibliopolis, 1986.

[Bus87] S. Buss. The Boolean formula value problem is in ALOGTIME. *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC'87)*, pages 123–131, 1987.

[Bus90] S. R. Buss. Axiomatizations and Conservation Results for Fragments of Bounded Arithmetic. *Contemporary Mathematics*, 106:57–84, 1990.

[Bus91] S. Buss. Propositional consistency proofs. *Annals of Pure and Applied Logic*, 52:3–29, 1991.

[Bus93] S. Buss. Algorithms for Boolean formula evaluation and for tree-contraction. In P. Clote and J. Krajicek, editors, *Proof Theory, Complexity, and Arithmetic*, pages 95–115. Oxford University Press, 1993.

[Bus95] S. R. Buss. Relating the Bounded Arithmetic and Polynomial Time Hierarchies. *Annals of Pure and Applied Logic*, 75:67–77, 1995.

[Bus98a] S. R. Buss. First-order proof theory of arithmetic. In S. R. Buss, editor, *Handbook of proof theory*, pages 79–147. Elsevier Science, 1998.

[Bus98b] S. R. Buss. An introduction to proof theory. In S. R. Buss, editor, *Handbook of proof theory*, pages 1–78. Elsevier Science, 1998.

[Bus98c] S. R. Buss. Lower bounds on Nullstellensatz proofs via designs. In P. W. Beame and S. R. Buss, editors, *Proof Complexity and Feasible Arithmetics*, DIMACS, pages 59–71. American Math. Soc, 1998.

[Bus03] S. R. Buss. Polynomial-size Frege and resolution proofs of st-connectivity and Hex principles. *submitted manuscript*, pages 1–31, 2003.

[Bus04] S. R. Buss. Bounded arithmetic and constant depth Frege proofs. To appear in Quaderni, 2004.

[CEI96] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 174–183, 1996.

[CIY97] S. A. Cook, R. Impagliazzo, and T. Yamakami. A tight relationship between generic oracles and type-2 complexity. *Information and Computation*, 137(2):159–170, 1997.

[CK98] M. Chiari and J. Krajicek. Witnessing functions in bounded arithmetic and search problems. *Journal of Symbolic Logic*, 63:1095–1115, 1998.

[CK03] S. Cook and A. Kolokolova. A second-order system for polytime reasoning based on Grädel's theorem. *Annals of Pure and Applied Logic*, 124:193–231, 2003.

[CM04] S. A. Cook and T. Morioka. Quantified propositional calculus and a second-order theory for $\mathbf{NC^1}$. Submitted to Archive for Mathematical Logic, 2004.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 83–97, 1971.

[Coo75] S. A. Cook. Feasibly constructive proofs and the propositional calculus. In *Prceedings of the 7th Annual ACM Symposium on the Theory of Computing*, pages 83–97, 1975.

[Coo02] S. A. Cook. Proof complexity and bounded arithmetic, 2002. Course notes for CSC2429, available at `http://www.cs.toronto.edu/~sacook/`.

[Coo03] S. A. Cook. Relativized propositional calculus. An unpublished working paper, 2003.

[Coo04] S. Cook. Theories for complexity classes and their propositional translations. *submitted*, pages 1–36, 2004.

[CR77] S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1977.

[CT92] P. Clote and G. Takeuti. Bounded arithmetic for NC, AlogTIME, L, and NL. *Annals of Pure and Applied Logic*, 56:73–117, 1992.

[CT04] S. Cook and N. Thapen. The strength of replacement in weak arithmetic. In *Proceedings of the Nineteenth Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, 2004.

[Dow79] M. J. Dowd. *Propositional representations of arithmetic proofs*. PhD thesis, University of Toronto, 1979.

[FS88] L. Fortnow and M. Sipser. Are there interactive proofs for coNP languages? *Information Processing Letters*, 28:249–251, 1988.

[Gen35] G. Gentzen. Untersuchungen über das logische schliessen. *Mathematiche Zeitschrift*, 35:176–210,405–431, 1935. English translation in: M. E. Szabo, The collected papers of Gerhard Gentzen, North-Holland, 1969.

[HCC+92] Hartmanis, Chang, Chari, Ranjan, and Rohatgi. Relativization: A revisionistic retrospective. *BEATCS: Bulletin of the European Association for Theoretical Computer Science*, 47, 1992.

[Imm99] N. Immerman. *Descriptive Complexity*. Springer, 1999.

[Joh90] D. S. Johnson. A catalog of complexity classes. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, pages 67–161. Elsevier Science Publishers, 1990.

[JPY88] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[JT95] B. Jenner and J. Toaŕan. Computing functions with parallel queries to NP. *Theoretical Computer Science*, 141:175–193, 1995.

[JT97] B. Jenner and J. Toaŕan. The complexity of obtaining solutions for problems in NP and NL. In L. A. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, pages 155–178. Springer, 1997.

[Kad88] J. Kadin. The polynomial time hierarchy collapses if the Boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, 1988.

[KP90] J. Krajicek and P. Pudlak. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschrift f. Mathematikal Logik u. Grundlagen d. Mathematik*, 36:29–46, 1990.

[KPT91] J. Krajicek, P. Pudlak, and G. Takeuti. Bounded arithmetic and the polynomial hierarchy. *Annals of Pure and Applied Logic*, 52:143–153, 1991.

[KPW95] J. Krajicek, P. Pudlak, and A. Woods. Exponential lower bounds to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures and Algorithms*, 7(1):15–39, 1995.

[Kra90] J. Krajicek. Exponentiation and second order bounded arithmetic. *Annals of Pure and Applied Logic*, 48(3):261–276, 1990.

[Kra93] J. Krajicek. Fragments of Bounded Arithmetic and Bounded Query Classes. *Transactions of the American Mathematical Society*, 338(2):587–598, 1993.

[Kra95] J. Krajicek. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 1995.

[Kra01] J. Krajicek. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1–3):123–140, 2001.

[Kra04] J. Krajicek. Implicit proofs. *Journal of Symbolic Logic*, 69(2):387–397, 2004.

[Kre88] M. W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36:490–509, 1988.

[LTT89] D. C. Llewellyn, C. Tovey, and M. Trick. Local optimization on graphs. *Discrete Applied Mathematics*, pages 157–178, 1989.

[Mor01] T. Morioka. Classification of Search Problems and Their Definability in Bounded Arithmetic. Master's thesis, University of Toronto, 2001. Also available as ECCC technical report TR01-82 at http://www.eccc.uni-trier.de.

[MP91] N. Megiddo and C. H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81:317–324, 1991.

[MPW00] A. Maciel, T. Pitassi, and A. Woods. A new proof of the weak pigeonhole. In *Proceedings of the Thirty Second Annual ACM Symposium on Theory of Computing (STOC'00)*, pages 368–377, 2000.

[NC04] P. Nguyen and S. Cook. VTC^0: a second-order theory for TC^0. In *Proceedings of the Nineteenth Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, 2004.

[Ngu04] P. Nguyen. Proving that $\mathbf{VNC}^1$ is finitely axiomatizable. unpublished note, 2004.

[Pap94a] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pap94b] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48:498–532, 1994.

[Pap01] C. H. Papadimitriou. Algorithms, Games, and the Internet. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 749–753, 2001.

[PBI93] T. Pitassi, P. Beame, and R. Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3(2):97–140, 1993.

[Pit02] T. Pitassi. Using hardness to prove Frege lower bounds, 2002. A seminar at the Fields Institute for Research in Mathematical Sciences, Toronto, Canada.

[Pol97] C. Pollett. A propositional proof systems for R^i_2. In P. Beame and S. Buss, editors, *Proof Complexity and Feasible Arithmetics*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 253–278. The American Mathematical Society, 1997.

[Pol99] C. Pollett. Structure and Definability in General Bounded Arithmetic Theories. *Annals of Pure and Applied Logic*, 100:189–245, 1999.

[PS88] I. Parberry and G. Schnitger. Parallel computation with threshold functions. *Journal of Computer and System Sciences*, 36:278–302, 1988.

[PW81] J. B. Paris and A. J. Wilkie. Models of arithmetic and the rudimentary sets. *Bull. Soc. Math. Belg.*, 33(1):157–169, 1981.

[PW85] J. Paris and A. Wilkie. Counting problems in bounded arithmetic. In *Methods in Mathematical Logic: Proceedings of teh 6th Latin American Symposium on Mathematical Logic 1983*, volume 1130 of *Lecture Notes in Mathematics*, pages 317–340, Berlin, 1985. Springer-Verlag.

[Raz93] A. A. Razborov. An equivalence between second order bounded domain bounded arithmetic and first order bounded arithmetic. In P. Clote and J. Krajicek, editors, *Arithmetic, Proof Theory and Computational Complexity*, pages 247–277. Oxford University Press, 1993.

[Raz98] A. Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, 1998.

[Rii93] S. Riis. Making infinite structures finite in models of second order bounded arithmetic. In P. Clote and J. Krajicek, editors, *Arithmetic, Proof Theory, and Computational Complexity*, pages 289–319. Oxford University Press, 1993.

[Rii01] S. Riis. A complexity gap for tree resolution. *Computational Complexity*, 10:179–209, 2001.

[Sel94] A. L. Selman. A taxonomy of complexity classes of functions. *Journal of Computer and System Sciences*, 48:357–381, 1994.

[SvS04] R. Savani and B. von Stengel. Exponentially many steps for a Nash equilibrium in a bimatrix game. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004)*, 2004.

[Tak87] G. Takeuti. *Proof Theory*. Elsevier Science Publisers, second edition, 1987.

[Tak93] G. Takeuti. RSUV isomorphisms. In P. Clote and J. Krajicek, editors, *Arithmetic, Proof Theory, and Computational Complexity*, pages 364–386. Oxford University Press, 1993.

[Tow90] M. Townsend. Complexity for type-2 relations. *Notre Dame Journal of Formal Logic*, 31(2):241–262, 1990.

[Urq95] A. Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 1(4), 1995.

[Wag90] K. W. Wagner. Bounded Query Classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

[WP87] A. J. Wilkie and J. B. Paris. On the scheme of induction for bounded arithmetic formulas. *Annals of Pure and Applied Logic*, 35:261–302, 1987.

[Wra77] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1977.

[Yan97] M. Yannakakis. Computational complexity. In E. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 19–55. John Wiley and Sons Ltd., 1997.

[Zam96] D. Zambella. Notes on polynomially bounded arithmetic. *Journal of Symbolic Logic*, 61:942–966, 1996.

Index