Olaf Beyersdorff

Non-classical Aspects in Proof Complexity

Cuvillier Verlag Göttingen
Internationaler wissenschaftlicher Fachverlag

Non-classical Aspects in Proof Complexity

Non-classical Aspects in Proof Complexity

Habilitationsschrift

Olaf Beyersdorff

Leibniz Universität Hannover

2011

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

1. Aufl. - Göttingen: Cuvillier, 2012
Zugl.: Hannover, Univ., Diss., 2011

978-3-95404-036-0

Nonnenstieg 8, 37075 Göttingen
Telefon: 0551-54724-0
Telefax: 0551-54724-21
www.cuvillier.de

1. Auflage, 2012
Gedruckt auf säurefreiem Papier

978-3-95404-036-0

Zusammenfassung

Die vorliegende Habilitationsschrift untersucht zwei neue nichtklassische Aspekte in der Beweiskomplexität: Beweissysteme mit nicht-uniformer Information (Advice) und parametrisierte Beweiskomplexität. Wir erläutern zunächst den Hintergrund und die Motivation für unsere Untersuchungen und beschreiben anschließend unsere erzielten Hauptresultate.

Aussagenlogische Beweiskomplexität

Die Beweiskomplexität ist ein aktives Forschungsgebiet, das in interdisziplinärer Weise logische, kombinatorische und komplexitätstheoretische Methoden zur Untersuchung formaler Beweise einsetzt. Neben der Untersuchung erststufiger Kalküle hat sich der überwiegende Teil bisheriger Forschungsaktivität mit aussagenlogischen Beweisen beschäftigt. Die zentrale Frage der aussagenlogischen Beweiskomplexität kann folgendermaßen formuliert werden: wie lang ist der kürzeste Beweis einer gegebenen aussagenlogischen Tautologie in einem spezifizierten Beweissystem? Wegen Anwendungen im automatischen Theorembeweisen ist daneben auch die effiziente Konstruktion von Beweisen von großer Bedeutung.

Der Begriff eines aussagenlogischen Beweissystems wurde 1979 in einer wegweisenden Arbeit von Cook und Reckhow als eine in polynomieller Zeit berechenbare Funktion formalisiert, deren Wertebereich mit der Menge aller aussagenlogischen Tautologien übereinstimmt [CR79]. Die Betonung wird in dieser Definition auf das effiziente Verifizieren von Beweisen gelegt, wohingegen das Konstruieren von Beweisen wesentlich schwieriger sein kann.

Der Zusammenhang zwischen der Beweislänge und zentralen komplexitätstheoretischen Fragen wurde bereits von Cook und Reckhow hergestellt [CR79]. Ein Beweissystem heißt polynomiell beschränkt, falls das Beweissystem polynomiell lange Beweise aller Tautologien erlaubt. Cook und Reckhow zeigten, dass es genau dann ein polynomiell beschränktes aussagenlogisches Beweissystem gibt, wenn die Klasse NP unter Komplementbildung abgeschlossen ist. Mithin können superpolynomielle untere Schranken für die Beweislänge in konkreten Beweissystemen als Etappe auf dem Weg zum Nachweis von $\mathsf{NP} \neq \mathsf{coNP}$ (und damit auch $\mathsf{P} \neq \mathsf{NP}$) verstanden werden.

Eine erste wichtige Station in diesem Programm war Hakens Nachweis exponentieller unterer Schranken für das Pigeonhole Principle im Resolutionskalkül [Hak85]. In den letzten zwei Jahrzehnten konnte dieses Ergebnis auf eine Reihe weiterer Beweissysteme wie das Nullstellensatzsystem [BIK$^+$96], Cutting Planes [BPR97, Pud97] oder den Polynomial Calculus [CEI96, Raz98] ausgedehnt werden. Für all diese Beweissysteme sind exponentielle untere Schranken für die Beweislänge für konkrete, zumeist kombinatorische Prinzipien beschreibende Tautologienfolgen bekannt.

Im Zuge dieser Fortschritte wurden mehrere generische Ansätze und allgemeine Techniken zum Nachweis unterer Schranken entwickelt: so die auf Krajíček zurückgehende Methode der effizienten Interpolation [Kra97], der erstmals von Ben-Sasson und Wigderson benutzte Zusammenhang zwischen der Größe und Breite von Beweisen [BSW01], sowie der von Krajíček und Razborov verfolgte Einsatz kryptographischer Pseudozufallsgeneratoren in der Beweiskomplexität [ABSRW04, Kra01, Kra04a, Kra07].

Beweissysteme und beschränkte Arithmetik

Eine wichtige Querverbindung führt von aussagenlogischen Beweissystemen in die erststufige Logik zur beschränkten Arithmetik. Diese Theorien, eingeführt von Cook [Coo75] und Buss [Bus86], sind schwache Fragmente der Peano-Arithmetik, die gerade genug Ausdruckskraft besitzen, um effiziente Berechnungsmodelle formalisieren zu können. Die Beziehung zu aussagenlogischen Beweissystemen ergibt sich durch die Übersetzung erststufiger arithmetischer Formeln in Folgen aussagenlogischer Tautologien. Eine erste Übersetzung geht ebenfalls auf Cook zurück [Coo75], weitere stammen von Paris und Wilkie [PW85] sowie von Krajíček und Pudlák [KP90].

Mittels dieser Übersetzungen formalisieren Krajíček und Pudlák [KP90] eine Korrespondenz zwischen aussagenlogischen Beweissystemen und Theorien beschränkter Arithmetik, wobei sich beide Komponenten bezüglich ihrer Ausdruckskraft wechselseitig charakterisieren. Unter Benutzung dieser Korrespondenz zeigte Ajtai seine berühmte untere Schranke für Frege-Systeme beschränkter Tiefe [Ajt94], die zusammen mit nachfolgenden Verbesserungen [BIK$^+$92, BPI93, KPW95] eines der derzeit stärksten Resultate der Beweiskomplexität darstellt. Auch obere Schranken sowie Simulationen zwischen Beweissystemen lassen sich sehr elegant mittels der Korrespondenz zur beschränkten Arithmetik erzielen [KP89, Pud91, KP98, Kra04b].

Neben der Beziehung zu aussagenlogischen Beweissystemen gibt es eine enge Verbindung zwischen beschränkter Arithmetik und Komplexitätstheorie [Kra95, CN10]. Die fruchtbaren Querbeziehungen zwischen Komplexitätstheorie, Beweiskomplexität und beschränkter Arithmetik werden auch in vorliegender Arbeit deutlich, in der Techniken und Konzepte dieser drei Gebiete zum Einsatz kommen.

Nichtklassische Modelle für Beweissysteme

Zur Analyse algorithmischer Probleme werden heute neben der Einteilung in klassische Aufwandsklassen wie P und NP vielfach alternative Ressourcen wie Randomisierung oder Quantenberechnungen sowie neue komplexitätstheoretische Konzepte wie parametrisierte Komplexität herangezogen. In der Beweiskomplexität, in der bislang klassische Beweissysteme wie Resolution im Vordergrund standen, wurden diese Betrachtungen im letzten Jahrzehnt von mehreren Forschergruppen mit einer Reihe eindrucksvoller Resultate begonnen, befinden sich aber insgesamt noch in einem frühen Stadium.

In vorliegender Habilitationsschrift leisten wir zu dieser Forschungslinie mit der Untersuchung zweier wichtiger komplexitätstheoretischer Paradigmen, die in jüngster Zeit viel Aufmerksamkeit erfahren haben, mehrere Beiträge.

Beweissysteme mit Advice

Im klassischen Modell von Cook und Reckhow werden Beweise in Polynomialzeit verifiziert. Eine interessante Frage ist, welche Ausdrucksstärke sich für Beweissysteme ergibt, die zur Verifikation stärkere Berechnungsressourcen nutzen. Mit der Untersuchung von aussagenlogischen *Beweissystemen mit Advice* haben Cook und Krajíček kürzlich erste interessante Ergebnisse zu obiger Frage geliefert [CK07]. Die Benutzung von Advice, die auf Karp und Lipton zurückgeht [KL80], erlaubt der Basismaschine den Zugriff auf eine kontrollierte Menge nichtuniformer Information. Im Fall von polynomiellem Advice entspricht dieses Modell polynomiell großen booleschen Schaltkreisen [Pip79]. In [CK07] untersuchen Cook und Krajíček erstmals Beweissysteme mit Advice und zeigen, dass auch diese Systeme in enger Korrespondenz zu arithmetischen Theorien stehen.

In vorliegender Habilitationsschrift führen wir Cook und Krajíčeks Forschungslinie fort und entwickeln Bausteine zu einer allgemeinen Theorie von Beweissystemen mit Advice. In Kapitel 4 widmen wir uns der Frage, für welche Sprachen es polynomiell beschränkte Beweissysteme mit Advice gibt. Wie in Cook und Reckhows klassischem Resultat [CR79] erhalten wir für diese Frage eine vollständige komplexitätstheoretische Charakterisierung. Insbesondere ergibt sich eine überraschende Verbindung zur nichtdeterministischen Instanz-Komplexität von Arvind, Köbler, Mundhenk und Torán [AKMT00], die ähnlich wie die Kolmogoroff-Komplexität die Härte einzelner Instanzen – jedoch relativ zu einer gegebenen Sprache – untersucht.

Möglichkeiten zur Vereinfachung des für den praktischen Einsatz unrealistisch starken Advice-Modells analysieren wir in Abschnitt 4.3. Mit Hilfe einer neuen Zähltechnik von Buhrman und Hitchcock [BH08] erreichen wir folgende Vereinfachung: falls Advice nützlich im Sinne kürzerer Beweise ist, so lässt sich die Advice-Menge reduzieren. Zudem zeigen wir, dass zum Beweis aussagenlogischer Tautologien statt logarithmischem Advice auch ein dünnes NP-Orakel genügt, wodurch die Nichtuniformität des Advice-Modells stark eingeschränkt wird.

Ein überraschendes Resultat von Cook und Krajíček [CK07] ist die Angabe eines optimalen aussagenlogischen Beweissystems, welches nur ein Advice-Bit benötigt. Hinsichtlich dieses Optimalitätsresultats, das wir auf beliebige Sprachen erweitern, ist es interessant, ob sich das optimale Beweissystem zu einem p-optimalen System verbessern lässt, d. h. ob sich die Simulationen auch effizient berechnen lassen. In Abschnitt 4.4 gehen wir dieser Frage nach und geben sowohl positive als auch negative Teilantworten. In Abschnitt 4.5 zeigen wir, dass die Beziehungen zwischen optimalen Beweissystemen und vollständigen Problemen für Promise-Klassen auch im Advice- bzw. Orakelmodell gültig bleiben.

Beweissysteme mit Advice und beschränkte Arithmetik

Das klassische Resultat von Karp und Lipton [KL80] über Kollapskonsequenzen aus der Annahme der Existenz polynomiell großer Schaltkreise für SAT, d. h. SAT $\in \mathsf{P/poly}$, hat eine ganze Reihe von Forschern zu raffinierten Verbesserungen inspiriert [KW98, Cai07]. In [CK07] untersuchen Cook und Krajíček die Frage, welche Kollapskonsequenzen sich aus der zusätzlichen Annahme eines einfachen Beweises für SAT $\in \mathsf{P/poly}$ ergeben. Insbesondere zeigen Cook und Krajíček, dass unter der Annahme der Beweisbarkeit von SAT $\in \mathsf{P/poly}$ in der Theorie PV die Polynomialzeithierarchie auf die boolesche Hierarchie kollabiert, welches im Vergleich zu den oben erwähnten klassischen Resultaten einen deutlich stärkeren Kollaps darstellt. In [CK07] stellen Cook und Krajíček die Frage, ob dieser Kollaps bereits die Annahme der Beweisbarkeit von SAT $\in \mathsf{P/poly}$ in PV charakterisiert. Diese Frage können wir in Kapitel 5 durch Formalisierung einer Technik von Buhrman, Chang und Fortnow [BCF03] positiv beantworten.

In diesem Ergebnis machen wir ebenfalls von Beweissystemen mit Advice und ihren Bezügen zur beschränkten Arithmetik Gebrauch. Zudem geben wir eine natürliche Beschreibung für das optimale Beweissystem in Form eines erweiterten Frege Systems mit Advice.

Parametrisierte Beweiskomplexität

Ein derzeit viel beachtetes Paradigma ist die parametrisierte Komplexität, die eine wesentlich feinere Sicht auf NP-vollständige Probleme erlaubt [FG06, Nie06]. In der parametrisierten Komplexität bestehen Instanzen aus der Eingabe x zusammen mit einem Parameter k. Ein Problem ist *fixed-parameter tractable* wenn es in Zeit $f(k) \cdot n^{O(1)}$ mit einer beliebigen Funktion f gelöst werden kann. In diesem Modell erlauben viele klassisch harte Probleme wie etwa Vertex Cover mit der Größe der Knotenüberdeckung als Parameter effiziente Lösungen für kleine Parameterwerte.

Neben der Klasse FPT aller fixed-parameter lösbaren Probleme gibt es eine Vielzahl parametrisierter Komplexitätsklassen mit Problemen, die vermutlich nicht fpt-lösbar sind. Die prominentesten Klassen bilden die sogenannte Weft-Hierarchie $\mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \mathsf{W}[3] \subseteq \dots$ Ein zentrales Problem aus der Klasse $\mathsf{W}[2]$ ist Weighted CNF Sat, bei dem man für Instanzen (φ, k) mit einer CNF φ nach einer erfüllenden Belegung für φ mit Gewicht k fragt (d. h. k Variablen werden auf 1 gesetzt). Viele kombinatorische Probleme wie die Suche nach einer Clique der Größe k lassen sich in natürlicher Weise in Weighted CNF Sat beschreiben.

Parametrisierte Beweissysteme wurden kürzlich von Dantchev, Martin und Szeider eingeführt [DMS07]. In diesem Modell steht mit einer fpt-Zeitschranke nicht nur zur Beweisverifikation mehr Zeit zur Verfügung – auch die Beweislängen lassen sich abhängig vom gewählten Parameter genauer im Spektrum zwischen polynomiell und exponentiell einordnen. In der Arbeit [DMS07] konzentrieren sich die Autoren auf *parametrisierte Widersprüche*, d. h. aussagenlogische Formeln φ, die nicht durch Belegungen mit Gewicht höchstens k erfüllt werden. Da es sich hierbei um das Komplement von Weighted CNF Sat handelt, ist die Klasse aller parametrisierten Widersprüche vollständig für $\mathsf{coW}[2]$. Für die Sprache aller parametrisierten Widersprüche konstruieren Dantchev et al. [DMS07] eine parametrisierte Variante des Resolutionssystems.

Parametrisierte Komplexität führt zu einem genaueren algorithmischen Verständnis klassisch schwieriger, insbesondere NP-vollständiger Probleme. Diese Feinstruktur des Effizienten in die Beweiskomplexität zu übertragen ist eines der Hauptziele unserer Untersuchungen in Kapitel 7. Aussagenlogische Beweissysteme lassen sich als nichtdeterministische Algorithmen für das Tautologieproblem interpretieren. Mit der Untersuchung parametrisierter Beweissysteme gelangen wir also wie in der Theorie der parametrisierten Komplexität zu einem genaueren Verständnis der Abgrenzung zwischen Effizienz und Nicht-Effizienz für *nichtdeterministische* Algorithmen.

Für die Beweiskomplexität kondensiert dieses genauere algorithmische Verständnis in einer feineren Klassifizierung der Beweislängen. Dies wird auch in unseren Ergebnissen in Kapitel 7 deutlich. Im Gegensatz zur klassischen Resolution erscheint parametrisierte Resolution als relativ starkes Beweissystem, da eine Reihe klassisch harter Formeln fpt-beschränkte Beweise sogar in baumförmiger parametrisierter Resolution besitzt. Dies zeigen wir, indem wir das Konzept eines Kernels aus der parametrisierten Komplexität in die Beweiskomplexität übertragen und für viele klassische Prinzipien, wie die Klasse aller CNFs beschränkter Breite, Kernelisierungen angeben. Konkrete Beispiele für klassisch harte Formeln mit fpt-beschränkten Beweisen sind das Lineare Ordnungsprinzip, Pebbling Tautologien, Färbbarkeitsprinzipien oder Tseitin Tautologien.

Für Härteresultate in baumförmiger parametrisierter Resolution entwickeln wir ein 2-Personen-Spiel, mit dessen Hilfe wir Beweislängen in baumförmiger Resolution im parametrisierten Kontext abschätzen. Dieses Spiel entwickelt das Prover-Delayer-Spiel von Pudlák und Impagliazzo [PI00] weiter. Für Pudlák und Impagliazzos Methode ist es wichtig, dass die Beweisbäume möglichst große balancierte Teilbäume enthalten – nur für diese wird die untere Schranke gezeigt. Bei parametrisierten Widerlegungen sind die kürzesten Pfade aber immer von der Länge k: es gibt also keine genügend großen balancierten Teilbäume. Daher entwerfen wir in Kapitel 6 ein asymmetrisches Prover-Delayer-Spiel, das auch für nichtbalancierte Beweisbäume einsetzbar ist. Zudem zeigen wir, dass unsere neue Technik auch starke untere Schranken in anderen baumförmigen Beweissystemen liefert.

Für allgemeine, d. h. nicht baumförmige, parametrisierte Resolution zeigen wir in Abschnitt 7.8 für das Pigeonhole Principle die erste untere Schranke. Auch hierzu benutzen wir einen spieltheoretischen Ansatz, der auf Pudlák [Pud00] zurückgeht, sich jedoch grundlegend von obigem Spiel unterscheidet.

Im abschließenden Kapitel 8 stellen wir unsere Untersuchungen in einen größeren Forschungszusammenhang. Hierbei geht es darum, erfolgreiche komplexitätstheoretische Konzepte wie etwa Advice, Parametrisierungen, aber auch Randomisierung, Quantenberechnungen oder Platzbedarf verstärkt zur beweistheoretischen Analyse einzusetzen. Wir glauben, dass weitere Forschung in dieser Richtung sowohl die beidseitigen Beziehungen zwischen Komplexitätstheorie und Beweiskomplexität stärken als auch fruchtbare Rückschlüsse für klassische Beweissysteme ermöglichen wird.

Contents

Acknowledgments

This work was supported by many people. First of all I am very grateful to Heribert Vollmer for his constant support and advice during the preparation of this thesis. I am also very indebted to Nicola Galesi who was my host during a very inspiring visit to Sapienza University Rome from October 2009 until April 2010 where part of the research presented in this work was done.

There are many other people with whom I had the privilege to collaborate in the last years. In particular, I thank Johannes Köbler, Jan Krajíček, Massimo Lauria, Arne Meier, Sebastian Müller, Zenon Sadowski, and Michael Thomas for research collaboration and many helpful and stimulating discussions on the topic of this work.

A significant part of the research presented here was supported by DFG grant KO 1053/5 and by grant N. 20517 from the John Templeton Foundation.

Chapter 1

Introduction

> Wenn es eine Maschine mit [...] gäbe, hätte das Folgerungen von der größten Tragweite. Es würde offenbar bedeuten, daß man trotz der Unlösbarkeit des Entscheidungsproblems die Denkarbeit der Mathematiker bei ja-oder-nein Fragen vollständig (abgesehen von der Aufstellung der Axiome) durch eine Maschine ersetzen könnte.[1]
>
> Kurt Gödel

Proof complexity is the area of research within complexity theory whose main aim is to understand and classify the complexity of theorem-proving procedures. Proof complexity is a theory which provides a very promising approach based mainly on mathematical logic, on model theory, and on combinatorics to some of the main questions and problems in complexity theory, as for instance the exact relationship between the classes P and NP.

Since its origins in the late sixties, computational complexity considered as its main computational paradigm the classical computational model of the Turing machine, invented by Alan Turing in the thirties [Tur36]. In the last 25 years, complexity theory moved forward to expand the concept of computational model. New models of computation were introduced, also looking at and exploring other scientific disciplines as physics or advanced mathematics, like probability theory. The new computational models studied frequently involve alternative resources such as randomization, quantum computation, or even a limited amount non-computable information. The investigation of the main questions of complexity theory from the point of view of these new computational paradigms and models was very fertile and fruitful in the last twenty years. For instance, quantum algorithms have been proved to be strictly more efficient than classical algorithms on very important problems. The use of randomization, among many other

[1] Kurt Gödel in a letter to John von Neuman in 1956 (reprinted in [Göd93]).

aspects, has provided us with the new theory of probabilistically checkable proofs which represents one of the main attempts to tackle questions like P vs. NP.

Proof complexity is younger than complexity theory and so far its investigation has been essentially based only on the classical computational model of the Turing machine. Only in the very last years some leading research groups have initiated the study of proof complexity from the point of view of other models of computation. This thesis focuses on developing and contributing to these recent lines of research by considering two non-classical aspects from computational complexity for theorem proving: proof systems with advice and parameterized proof systems. In the following section we will give a brief overview of our main results on these two models.

1.1 Motivation, Models, and Main Results

1.1.1 Proof Systems with Advice

Complexity classes with advice were introduced by Karp and Lipton [KL80]. The idea is here to enhance efficient computations with a limited amount of non-uniform information: *the advice*. By using advice we leave the realm of effective computability because the advice can be arbitrarily complex, even non-computable. But we impose limits on the *amount* of advice that we are allowed to use and in this way obtain interesting computational models such as Boolean circuits [Pip79].

Recently, Cook and Krajíček [CK07] introduced *proof systems with advice* which—similarly as in the complexity approach mentioned above—may use a limited amount of non-uniform information for the verification of proofs. Their results show that, like in the classical Cook-Reckhow setting, these proof systems enjoy a close connection to theories of bounded arithmetic. Moreover, Cook and Krajíček obtained the surprising result that with only one bit of advice, an *optimal* proof system can be realized. The existence of such an optimal proof system, i. e., the strongest possible system, is not known in the classical model. Thus proof systems with advice appear to be a strictly more powerful model.

In this thesis we provide a rigorous development of the theory of proof systems with advice and investigate the following fundamental questions for this new model:

Q1: Given a language L, do there exist polynomially bounded proof systems with advice for L?
Q2: For propositional proof systems, does advice help to shorten proofs?
Q3: Do there exist optimal proof systems with advice for L?

For question Q1, one of the major motivations for proof complexity [CR79], we obtain a complete complexity-theoretic characterization. The classical Cook-Reckhow Theorem states that $\mathsf{NP} = \mathsf{coNP}$ if and only if the set of all tautologies

TAUT has a polynomially bounded proof system, i.e., there exists a polynomial p such that every tautology φ has a proof of size $\leq p(|\varphi|)$ in the system. Consequently, showing super-polynomial lower bounds to the proof size in propositional proof systems of increasing strength provides one way to attack the P vs. NP problem. This approach, also known as the Cook-Reckhow program, has led to a very fruitful research on the length of propositional proofs (cf. [Pud98]).

As in the Cook-Reckhow Theorem above, we obtain a series of results leading to a complete characterization for Q1. In particular, we show a tight connection of this problem to the notion of nondeterministic instance complexity. Similarly as Kolmogorov complexity, instance complexity measures the complexity of individual instances of a language [OKSW94]. In its nondeterministic version, Arvind, Köbler, Mundhenk, and Torán [AKMT00] used this complexity measure to show that, under reasonable complexity-theoretic assumptions, there are infinitely many tautologies that are hard to prove in every propositional proof system. In the light of our investigation, this connection between nondeterministic instance complexity and proof complexity is strengthened by results of the following form: *all elements of a given language L have small instance complexity if and only if L has a proof system with advice such that every $x \in L$ has a short proof.*

For question Q2 we concentrate on the most interesting case of propositional proof systems. Unfortunately, proof systems with advice do not constitute a feasible model for the verification of proofs in practice, as the non-uniform advice can be very complex (and even non-recursive). Approaching question Q2, we therefore investigate whether the advice can be simplified or even eliminated without increasing the proof length. Our first result in this direction shows that proving propositional tautologies does not require complicated or even non-recursive advice: every propositional proof system with up to logarithmic advice is simulated by a propositional proof system computable in polynomial time with access to a sparse NP-oracle. Thus in propositional proof complexity, computation with advice can be replaced by a more realistic computational model.

While this result holds unconditionally, our next two results explore consequences of a positive or negative answer to question Q2. Assume first that advice helps to prove tautologies in the sense that proof systems with advice admit nontrivial upper bounds on the lengths of proofs. Then we show that the same upper bound can be achieved in a proof system with a simplified advice model. On the other hand, if the answer is negative in the sense that advice does not help to shorten proofs even for simple tautologies, then we obtain optimal propositional proof systems without advice.

This brings us to our last question Q3. While the existence of optimal proof systems in the classical model is a prominent open problem posed by Krajíček and Pudlák twenty years ago [KP89], question Q3 receives a surprising positive answer: optimal proof systems exist when a small amount of advice is allowed. For propositional proof systems this was already shown by Cook and Krajíček

[CK07]. Using the proof technique from [CK07], we show that for every language L, the class of all proof systems for L using logarithmic advice contains an optimal proof system and investigate whether the optimality result can be strengthened to it's efficient version of p-optimality. In addition, we show that the connection between optimal proof systems and promise classes also holds in the presence of advice.

Propositional proof systems enjoy a close connection to bounded arithmetic (cf. the monographs [Kra95, CN10] or the survey [Bey09]). Cook and Krajíček [CK07] use the correspondence between proof systems with advice and arithmetic theories to obtain a very strong Karp-Lipton collapse result in bounded arithmetic: if SAT has polynomial-size Boolean circuits, then the polynomial hierarchy collapses to the Boolean hierarchy. In Chapter 5 we show that this collapse consequence is in fact optimal with respect to the theory PV, thereby answering a question of Cook and Krajíček [CK07].

1.1.2 Parameterized Proof Systems

Parameterized complexity is widely considered one of the modern paradigms of computational complexity which considerably advances our understanding of intractable problems by offering a refined view on running times of algorithms. In proof complexity, this investigation has started recently with the work of Dantchev, Martin, and Szeider [DMS07]. There the authors introduce a general framework for parameterized proof complexity and consider a parameterized version of Resolution which is the best studied and most important propositional proof system in terms of applications.

In parameterized proof complexity, our main objective is to reach a more refined understanding of theorem proving by adapting concepts and techniques from parameterized complexity to proof complexity. Proof systems can be understood as non-deterministic algorithms for the tautology problem. Therefore, by considering parameterized proof systems we reach a better understanding of the borderline between efficiency and non-efficiency for *non-deterministic* algorithms. In proof complexity this condensates in a more refined classification of proof lengths. This view is supported by previous results from [DMS07, Gao09] and our investigation in this dissertation. For example, the hard case in the classical dichotomy for tree-like Resolution of Riis [Rii01] splits in the parameterized context into two cases: tautologies with fpt-bounded proofs and tautologies for which the shortest parameterized proof has size similar to exhaustive search, as shown in [DMS07].

In Chapter 7 we show that in contrast to classical Resolution, *Parameterized Resolution* appears to be a relatively powerful proof system as a number of classically hard principles admit fpt-bounded proofs even in *tree-like* Parameterized Resolution. We show this by transferring the concept of a kernel from parameterized complexity to proof complexity and constructing kernelizations for many classically hard principles as the class of all CNF's of bounded width. Specific examples of formulas which are hard for classical Resolution, but possess fpt-bounded proofs even in tree-like Parameterized Resolution include the linear ordering principle, pebbling tautologies, coloring principles, and Tseitin tautologies.

For hardness results we introduce a powerful two-player game to model and study the complexity of proofs in tree-like Parameterized Resolution. Our game refines the Prover-Delayer game of Pudlák and Impagliazzo [PI00] and makes it applicable in situations where the proof trees are very unbalanced. This technique also yields improved lower bounds for non-parameterized proof systems as we show in Chapter 6.

Although the Prover-Delayer game is a very general technique, it cannot be used for *dag-like* proofs. In Section 7.8 we obtain the first lower bound for dag-like Parameterized Resolution for the pigeonhole principle. For this lower bound we again use a game-theoretic argument originating in Pudlák's work [Pud00].

1.2 Organization of the Thesis and Published Parts

This thesis is organized as follows. Chapters 2 and 3 contain background information on proof complexity and computational complexity, respectively. These two chapters are largely of preliminary nature. Apart from known definitions and results, Sections 3.4 to 3.6 contain some new results on non-deterministic instance complexity, promise classes, and optimal proof systems which we apply in Chapter 4.

In Chapters 4 and 5 we investigate proof systems with advice, first from the perspective of computational complexity (Chapter 4) and then with respect to their relation to bounded arithmetic (Chapter 5).

In Chapter 6 we introduce a new technique for lower bounds in tree-like proof systems—the asymmetric Prover-Delayer game—and apply it to classical Resolution. Chapter 7 then contains our investigation of parameterized proof complexity and in particular of Parameterized Resolution where we again use the game of Chapter 6.

Chapter 8 concludes with a discussion of our two non-classical aspects that we investigate here and puts this work into a broader context.

Part of the results from this thesis are already published in journals or in conference proceedings. The relevant publications are

- [BKM11] containing Section 3.4 and most of Chapter 4;
- [BS11] containing Sections 3.5 and 3.6;
- [BM10b] containing Section 4.4.2 and all of Chapter 5;

- [BGL10] containing Chapter 6;
- [BGL11] and [BGLR11] containing most of the material in Chapter 7.

Chapter 2

Proof Complexity

> Nach diesen Bemerkungen sei es dem Verfasser noch erlaubt, einige Worte für sich anzuführen. Er hat sich bemüht, so kurz zu schreiben, als es ihm möglich war und es diese Gattung von Arbeiten erfordert. Es wäre zu wünschen, daß man sich dieses Gesetz der Kürze bei allen Büchern über das Altertum, die doch nicht unser ganzes Leben beschäftigen sollen, vorhalten möchte. Die meisten antiquarischen Schriftsteller gleichen durch ihre Weitschweifigkeit den Flüssen, die anschwellen, wenn man ihres Wassers nicht bedarf, und trocken bleiben, wo eben Wasser nötig wäre.[1]
>
> Johann Joachim Winckelmann

One of the starting points of propositional proof complexity is the seminal paper of Cook and Reckhow [CR79] where they formalized propositional proof systems as polynomial-time computable functions which have as their range the set of all propositional tautologies. In that paper, Cook and Reckhow also observed a fundamental connection between lengths of proofs and the separation of complexity classes: they showed that there exists a propositional proof system which has polynomial-size proofs for all tautologies (a *polynomially bounded* proof system) if and only if the class NP is closed under complementation. From this observation the so called *Cook-Reckhow program* was derived which serves as one of the major motivations for propositional proof complexity: to separate NP from coNP (and hence P from NP) it suffices to show super-polynomial lower bounds to the size of proofs in all propositional proof systems.

Although the first super-polynomial lower bound to the lengths of proofs had already been shown by Tseitin in the late 60's for a sub-system of Resolution [Tse68], the first major achievement in this program was made by Haken in 1985

[1] Winckelmann in der Vorrede der *Beschreibung der geschnittenen Steine des seligen Baron Stosch* (Florenz, 1760)

when he showed an exponential lower bound to the proof size in Resolution for a sequence of propositional formulas describing the pigeonhole principle [Hak85]. In the last two decades these lower bounds were extended to a number of further propositional systems such as the Nullstellensatz system [BIK$^+$96], Cutting Planes [BPR97, Pud97], Polynomial Calculus [CEI96, Raz98], or bounded-depth Frege systems [Ajt94, BIK$^+$92, BPI93, KPW95]. For all these proof systems we know exponential lower bounds to the lengths of proofs for concrete sequences of tautologies arising mostly from natural propositional encodings of combinatorial statements.

For proving these lower bounds, a number of generic approaches and general techniques have been developed. Most notably, there is the method of feasible interpolation developed by Krajíček [Kra97], the size-width trade-off introduced by Ben-Sasson and Wigderson [BSW01], and the use of pseudorandom generators in proof complexity [ABSRW04, Kra01, Kra04a].

Despite this enormous success many questions still remain open. In particular Frege systems currently form a strong barrier [BBP95], and all current lower bound methods seem to be insufficient for these strong systems. A detailed survey of recent advances in propositional proof complexity is contained in [Seg07].

Let us mention that the separation of complexity classes is not the only motivation for studying lengths of proofs. In particular for strong systems like Frege and its extensions there is a fruitful connection to bounded arithmetic which adds insight to both subjects (cf. [Kra95]). Further, understanding weak systems as Resolution is vital to applications as the design of efficient SAT solvers (see e. g. [PS10] for a more elaborate argument). Last not least, propositional proof complexity has over the years grown into a mature field and many researchers believe that understanding propositional proofs and proving lower bounds—arguably the hardest task in complexity—is a very important and beautiful field of logic which is justified in its own right.

2.1 Proof Systems

We start with a general semantic definition of proof systems:

Definition 2.1.1 *A* proof system *for a language L is a (possibly partial) surjective function $f : \Sigma^* \to L$. For $L = \mathrm{TAUT}$, f is called a* propositional proof system.

In the classical framework of Cook and Reckhow [CR79], proof systems are additionally required to be computable in polynomial time. As we are relaxing this definition in subsequent chapters we have chosen the more general semantic definition above where the computational resources to compute f are not specified.

We review important notions concerning proof systems. A string w with $f(w) = x$ is called an *f-proof* of x. Proof complexity studies lengths of proofs, so we use the following notion: for a function $t : \mathbb{N} \to \mathbb{N}$, a proof system f for L is *t-bounded* if every $x \in L$ has an f-proof of size $\leq t(|x|)$. If t is a polynomial, then f is called *polynomially bounded*. We recall the classical theorem of Cook and Reckhow on polynomially bounded proof systems:

Theorem 2.1.2 (Cook, Reckhow [CR79]) *A language L has a polynomially bounded proof system if and only if $L \in \mathsf{NP}$.*

2.2 Simulations and Optimal Proof Systems

Proof systems are compared according to their strength by simulations as introduced in [CR79] and [KP89]. If f and g are proof systems for L, we say that g *simulates* f (denoted $f \leq g$), if there exists a polynomial p such that for all $x \in L$ and f-proofs w of x there is a g-proof w' of x with $|w'| \leq p(|w|)$. If such a proof w' can even be computed from w in polynomial time, we say that g *p-simulates* f and denote this by $f \leq_p g$. If the systems f and g mutually (p-)simulate each other they are called *(p-)equivalent*, denoted by $f \equiv_{(p)} g$. A proof system for L is *(p-)optimal* if it (p-)simulates all proof systems for L.

Whether or not there exist optimal propositional proof system is open. Posed by Krajíček and Pudlák [KP89], this question has remained unresolved for more than twenty years. Sufficient conditions were established by Krajíček and Pudlák [KP89] by $\mathsf{NE} = \mathsf{coNE}$ for the existence of optimal and $\mathsf{E} = \mathsf{NE}$ for p-optimal propositional proof systems, and these conditions were subsequently weakened by Köbler, Messner, and Torán [KMT03]. Necessary conditions for the existence of optimal proof systems are tightly linked to the following question for promise complexity classes lacking an easy syntactic machine model:

Problem 2.2.1 *Do there exist complete problems for a given promise class C?*

Like the first question of the existence of optimal proof systems also Problem 2.2.1 has a long research record, dating back to the 80's when Kowalczyk [Kow84] and Hartmanis and Hemachandra [HH88] considered this question for $\mathsf{NP} \cap \mathsf{coNP}$ and UP. This research agenda continues to recent days where, due to cryptographic and proof-theoretic applications, disjoint NP-pairs have been intensively studied (cf. [GSS05, GSSZ04, GSZ07, Bey07] and [GSZ06] for a survey). Very recently, Itsykson has shown the surprising result that AvgBPP, the average-case version of BPP, has a complete problem [Its09].

Understanding these questions better through characterizations is an important problem with consequences to seemingly unrelated areas such as descriptive complexity: very recently, Chen and Flum [CF10] have shown that the existence of an optimal propositional proof system is equivalent to the open problem whether $L_{\leq}$ is a P-bounded logic for P. Other recent research concentrated on modified versions of Q1, where a number of surprising positive results have been obtained. Cook and Krajíček [CK07] have shown that optimal propositional proof systems exist under non-uniform information (advice), and even one bit of advice suffices (we will discuss this result in detail in Section 4.4). In another direction, Hirsch and Itsykson [HI10, Hir10] considered randomized proof systems and showed the existence of an optimal system in the class of all automatizable heuristic proof systems (cf. Chapter 8). Still another positive result was very recently obtained by Pitassi and Santhanam [PS10] who show that there exists an optimal quantified propositional proof system under a weak notion of simulation.

2.3 Two Examples of Proof Systems

We give two important examples of propositional proof systems which we will need later on: Resolution and Frege systems.

We start with Resolution. A *literal* is a positive or negated propositional variable and a *clause* is a set of literals. The *width* of a clause is the number of its literals. A clause is interpreted as the disjunctions of its literals and a set of clauses as the conjunction of the clauses. Hence clause sets correspond to formulas in CNF. The *Resolution system* is a refutation system for the set of all unsatisfiable CNF. Resolution uses as its only rule the *Resolution rule*

$$\frac{\{x\} \cup C \qquad \{\neg x\} \cup D}{C \cup D}$$

for clauses C, D and a variable x. The aim in Resolution is to demonstrate unsatisfiability of a clause set by deriving the empty clause. If in a derivation every derived clause is used at most once as a prerequisite of the Resolution rule, then the derivation is called *tree-like*, otherwise it is *dag-like*. The *size* of a Resolution proof is the number of its clauses where multiple instances of the same clause are counted separately. Undoubtedly, Resolution is the most studied and best-understood propositional proof system (cf. [Seg07]).

Our second example are Frege systems. Frege systems derive formulas using axioms and rules. In texts on classical logic these systems are usually referred to as Hilbert-style systems, but in propositional proof complexity it has become customary to call them Frege systems [CR79].

A *Frege rule* is a $(k+1)$-tuple $(\varphi_0, \varphi_1 \dots, \varphi_k)$ of propositional formulas such that

$$\{\varphi_1, \varphi_2, \dots, \varphi_k\} \models \varphi_0 \ .$$

The standard notation for rules is

$$\frac{\varphi_1 \quad \varphi_2 \quad \dots \quad \varphi_k}{\varphi_0} \ .$$

A Frege rule with $k = 0$ is called a *Frege axiom.*

A formula ψ_0 can be derived from formulas $\psi_1, \dots, \psi_k$ by a Frege rule $(\varphi_0, \varphi_1 \dots, \varphi_k)$ if there exists a substitution σ such that

$$\sigma(\varphi_i) = \psi_i \quad \text{for } i = 0, \dots, k \ .$$

Let $\mathcal{F}$ be a finite set of Frege rules. An *$\mathcal{F}$-proof* of a formula φ from a set of propositional formulas Φ is a sequence $\varphi_1, \dots, \varphi_l = \varphi$ of propositional formulas such that for all $i = 1, \dots, l$ one of the following holds:

1. $\varphi_i \in \Phi$ or

2. there exist numbers $1 \le i_1 \le \dots \le i_k < i$ such that φ_i can be derived from $\varphi_{i_1}, \dots, \varphi_{i_k}$ by a Frege rule from $\mathcal{F}$.

$\mathcal{F}$ is called *implicationally complete* if for all formulas φ and all sets of formulas Φ, $\Phi \models \varphi$ if and only if there exists an $\mathcal{F}$-proof of φ from Φ. If $\mathcal{F}$ is implicationally complete we call $\mathcal{F}$ a *Frege system.*

Without proof we note that the following set of axioms taken from [Bus98]

$$\begin{array}{l}
p_1 \to (p_2 \to p_1) \\
(p_1 \to p_2) \to (p_1 \to (p_2 \to p_3)) \to (p_1 \to p_3) \\
p_1 \to p_1 \lor p_2 \\
p_2 \to p_1 \lor p_2 \\
(p_1 \to p_3) \to (p_2 \to p_3) \to (p_1 \lor p_2 \to p_3) \\
(p_1 \to p_2) \to (p_1 \to \neg p_2) \to \neg p_1 \\
\neg\neg p_1 \to p_1 \\
p_1 \land p_2 \to p_1 \\
p_1 \land p_2 \to p_2 \\
p_1 \to p_2 \to p_1 \land p_2
\end{array}$$

together with the modus ponens rule

$$\frac{p \qquad p \to q}{q}$$

is an example of a Frege system.

This definition leaves much freedom to design individual Frege systems, but if we are only interested in the lengths of proofs there is only one Frege system F as already noted by Cook and Reckhow [CR79].

Theorem 2.3.1 (Cook, Reckhow [CR79]) *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be Frege systems. Then $\mathcal{F}_1 \equiv_p \mathcal{F}_2$.*

Now we describe an extension of Frege systems as introduced in [CR79]. Let $\mathcal{F}$ be a Frege system. An *extended Frege proof* of a formula φ from a set of formulas Φ is a sequence $(\varphi_1, \dots, \varphi_l = \varphi)$ of propositional formulas such that for each $i = 1, \dots, l$ one of the following holds:

1. $\varphi_i \in \Phi$ or
2. φ_i has been derived by an $\mathcal{F}$-rule or
3. $\varphi_i = q \leftrightarrow \psi$ where ψ is an arbitrary propositional formula and q is a new propositional variable that does not occur in φ, ψ and φ_j for $1 \leq j < i$.

The introduction of the extension rule 3 allows the abbreviation of possibly complex formulas by variables. Hence using this rule for formulas which appear very often in an $\mathcal{F}$-proof can substantially reduce the proof size.

As in Theorem 2.3.1 it follows that all extended Frege systems are polynomially equivalent. Therefore we only speak of the *extended Frege system* and denote it by EF. It is clear that EF simulates Frege systems, but whether EF is indeed a strictly stronger system is an open problem.

Chapter 3

Notions from Computational Complexity

> Men sometimes speak as if the study of the classics would at length make way for more modern and practical studies; but the adventurous student will always study classics, in whatever language they may be written, and however ancient they may be. For what are the classics but the noblest recorded thoughts of man? They are not only oracles which are not decayed, and there are such answers to the most modern inquiry in them as Delphi and Dodona never gave. We might as well omit to study Nature because she is old.
>
> Henry David Thoreau, *Walden*

Throughout this work we will use standard notions from computational complexity as they can be found in the textbooks [BDG88], [Pap94], or [AB09]. In the following we will only review those concepts and complexity classes which we will use in this dissertation, but which might not fall into the "canon" of well known complexity classes like P and NP. In Sections 3.1 to 3.3 we will just fix notation and recall definitions and known results, but starting from Section 3.4 we will also prove new results which we use in subsequent chapters.

3.1 Notation

Throughout the following we fix the alphabet $\Sigma = \{0, 1\}$. Σ^n denotes the set of strings of length n and $(\Sigma^n)^k$ the set of k-tuples of Σ^n. Let $\pi_i : \bigcup_{n\in\mathbb{N}}(\Sigma^*)^n \to \Sigma^*$ be the projection to the i^{th} string of some finite tuple and let $\pi_i^* : \Sigma^* \to \{0, 1\}$ be the projection to the i^{th} bit of a string. As usual we enumerate the bits of a string starting with 0 and thus for example $\pi_0^*(a_0a_1a_2) = a_0$.

Let $\langle\cdot\rangle$ be a polynomial-time computable function, mapping tuples of strings to strings. Its inverse will be denoted by *enc*.

A set $A \subseteq \Sigma^*$ is *sparse* if there exists a polynomial p such that for each $n \in \mathbb{N}$, $|A \cap \Sigma^n| \leq p(n)$. A sparse set A is called *tally* if $A \subseteq \{1^n \mid n \in \mathbb{N}\}$. The class of all sparse and tally sets are denoted by Sparse and Tally, respectively.

3.2 The Boolean Hierarchy

The *Boolean hierarchy* BH is the closure of NP under union, intersection, and complementation. The levels of BH are denoted BH_k, where BH_2 is also known as $\mathsf{D^p}$. The Boolean hierarchy coincides with $\mathsf{P}^{\mathsf{NP}[O(1)]}$ consisting of all languages which can be solved in polynomial time with constantly many queries to an NP oracle. For each level BH_k it is known that k non-adaptive queries to an NP-oracle suffice, i.e., $\mathsf{BH}_k \subseteq \mathsf{P}_{tt}^{\mathsf{NP}[k]}$ (cf. [Bei91]). If we allow $O(\log n)$ adaptive queries we get the presumably larger class $\mathsf{P}^{\mathsf{NP}[\log]}$.

Complete problems BL_k for BH_k are inductively given by $\mathrm{BL}_1 = \mathrm{SAT}$ and

$$\begin{aligned}
\mathrm{BL}_{2k} &= \{\langle x_1,\ldots,x_{2k}\rangle \mid \langle x_1,\ldots,x_{2k-1}\rangle \in \mathrm{BL}_{2k-1} \text{ and } x_{2k} \in \overline{\mathrm{SAT}}\} \\
\mathrm{BL}_{2k+1} &= \{\langle x_1,\ldots,x_{2k+1}\rangle \mid \langle x_1,\ldots,x_{2k}\rangle \in \mathrm{BL}_{2k} \text{ or } x_{2k+1} \in \mathrm{SAT}\} \ .
\end{aligned}$$

Observe that $\langle x_1,\ldots,x_k\rangle \in \mathrm{BL}_k$ if and only if there exists an $i \leq k$, such that x_i is satisfiable and the largest such i is odd.

3.3 Complexity Classes with Advice

Complexity classes with advice were introduced by Karp and Lipton [KL80]. In this computational model, efficient computations are augmented with a limited amount of non-uniform information: *the advice*. Using advice we can solve very complex, even non-computable problems. To obtain interesting computational models we impose limits on the *amount* of advice that we are allowed to use.

For each function $h : \mathbb{N} \to \Sigma^*$ and each language L we let

$$L/h = \{\, x \mid \langle x, h(|x|)\rangle \in L \,\} \ .$$

If C is a complexity class and F is a class of functions, then

$$\mathsf{C}/F = \{\, L/h \mid L \in \mathsf{C}, h \in F \,\} \ .$$

Usually the family of functions F is defined by some bound on the length of the values in terms of the argument. Thus, for example, $\mathsf{NP}/O(1)$ denotes the class of languages recognized by NP machines with advice functions h where $|h(n)|$ is bounded by a constant (cf. [BDG88]).

Apart from constant advice, specific choices of interesting advice bounds are *logarithmic advice*, defined as

$$\mathsf{C}/\mathsf{log} = \{\, L/h \mid L \in \mathsf{C}, |h(n)| \leq c \log n \text{ for some constant } c \,\}$$

and *polynomial advice*

$$\mathsf{C}/\mathsf{poly} = \{\, L/h \mid L \in \mathsf{C}, |h(n)| \leq p(n) \text{ for some polynomial } p \,\} \ .$$

It is clear that complexity classes with advice always contain non-recursive problems and hence every non-uniform class contains very complicated languages. It is, however, a very interesting and prominent open problem in computational complexity to determine which portion of classical complexity classes are contained in non-uniform classes, in particular, whether deterministic non-uniform classes contain nondeterministic classes like NP or coNP. One of the first and most prominent results in this direction is the theorem of Karp and Lipton:

Theorem 3.3.1 (Karp, Lipton [KL80]) *If* $\mathsf{NP} \subseteq \mathsf{P}/\mathsf{poly}$*, then the polynomial hierarchy* PH *collapses to its second level* Σ_2^{p}*.*

This theorem has inspired a number of researchers to search for better collapse consequences. We will come back to this issue and explain the results in detail in Chapter 5.

Another important result of this kind, which we will use at several places in this work, is the following theorem:

Theorem 3.3.2 (Buhrman, Chang, Fortnow [BCF03]) *For every constant* $k \geq 1$*,* $\mathsf{coNP} \subseteq \mathsf{NP}/k$ *if and only if* $\mathsf{PH} \subseteq \mathsf{BH}_{2^k}$*.*

3.4 Nondeterministic Instance Complexity

While Kolmogorov complexity studies the hardness of individual strings, the notion of instance complexity was introduced by Orponen, Ko, Schöning, and Watanabe [OKSW94] to measure the hardness of individual instances of a given language. The deterministic instance complexity of [OKSW94] was later generalized to the nondeterministic setting by Arvind, Köbler, Mundhenk, and Torán [AKMT00].

As required for Kolmogorov complexity and instance complexity, we fix a universal Turing machine $U(M, x)$ which executes nondeterministic programs M on inputs x. In the sequel, we refrain from always mentioning U explicitly. Thus we simply write statements like "M is a t-time bounded Turing machine" with the precise meaning that U always spends at most $t(n)$ steps to simulate M on inputs of length n. Likewise, to "simulate a machine M on input x" always means executing $U(M, x)$.

A nondeterministic Turing machine M is *consistent* with a language L (or L-consistent), if $L(M) \subseteq L$. We can now give the definition of nondeterministic instance complexity from [AKMT00].

Definition 3.4.1 (Arvind et al. [AKMT00]) *For a set L and a time bound t, the t-time-bounded* nondeterministic instance complexity *of x with respect to L is defined as*

$$nic^t(x : L) = \min\{\, |M| \, : M \textit{ is an } L\textit{-consistent } t\textit{-time-bounded nondeterministic machine, and } M \textit{ decides correctly on } x \,\} \ .$$

Similarly as in the deterministic case in [OKSW94], we collect all languages with prescribed upper bounds on the running time and nondeterministic instance complexity in a complexity class.

Definition 3.4.2 *Let F_1 and F_2 be two classes of functions. We define*

$$\mathsf{NIC}[F_1, F_2] = \{L : \textit{there exist } s \in F_1 \textit{ and } t \in F_2 \textit{ such that for all } x \in \Sigma^* \; nic^t(x : L) \leq s(|x|)\} \ .$$

A particularly interesting choice for the classes F_1 and F_2 is to allow polynomial running time, but only logarithmic descriptions for the machines. This leads to the class $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$ which plays a central role in Section 4.2. Similarly as in the deterministic case (cf. [OKSW94]), the next proposition locates this class between the nonuniform classes NP/log and $\mathsf{NP}/\mathsf{poly}$.

Proposition 3.4.3 $\mathsf{NP}/\mathsf{log} \subseteq \mathsf{NIC}[\mathsf{log}, \mathsf{poly}] \subseteq \mathsf{NP}/\mathsf{poly}$.

Proof. For the first inclusion, let $L \in \mathsf{NP}/\mathsf{log}$. Let M be a nondeterministic Turing machine with logarithmic advice that decides L and let a_n be the advice given to M for inputs of length n. We define a collection of programs M_{n,a_n} for L as follows. On input x the machine M_{n,a_n} first checks whether the length of the input is n. For this we need to code the number n into M_{n,a_n}. If $|x| \neq n$, then M_{n,a_n} rejects. Otherwise, M_{n,a_n} simulates M on input x with advice a_n which is also coded into M_{n,a_n}. Essentially, the machines M_{n,a_n} are constructed by hardwiring n and a_n into M, and thus the size of M_{n,a_n} is logarithmic in n. Therefore $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.

For the second inclusion, let $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. Then there exist a constant c and a polynomial p such that for all x we have $nic^p(x : L) \leq c \log |x| + c$. We construct a nondeterministic Turing machine M with polynomial advice that accepts exactly L. The advice of M for length n consists of all nondeterministic Turing machines $M_1, \dots, M_m$ of size at most $c \log n + c$ which are consistent with L. Note that for each input length n, there are only polynomially many machines of the appropriate size $\leq c \log n + c$. Hence polynomial advice suffices to encode the whole list $M_1, \dots, M_m$. On input x, the machine M simulates each M_i on

x for at most $p(|x|)$ steps. If any of the M_i accepts, then M accepts as well, otherwise it rejects.

We claim, that $L(M) = L$. For, if $x \in L$, then there is a nondeterministic L-consistent Turing machine M_i such that $M_i(x)$ accepts and $|M_i| \leq c \log |x| + c$. Thus, also $M(x)$ accepts. If, on the other hand, M accepts x, then so does some M_i which is consistent with L. Therefore, $x \in L$ because $L(M_i) \subseteq L$. □

In fact, the inclusions in Proposition 3.4.3 are proper as we will show in Theorem 3.4.5 below. For the proof we need the following notion:

Definition 3.4.4 (Buhrman, Fortnow, Laplante [BFL01]) *For a time bound t, the* nondeterministic decision complexity *of x, denoted $CND^t(x)$, is the minimal size of a t-time-bounded nondeterministic Turing machine M with $L(M) = \{x\}$.*

As already noted in [AKMT00], the *CND* measure provides an upper bound to the *nic* measure, i.e., for any language L and time bound t there is a constant $c > 0$ such that $nic^t(x : L) \leq CND^t(x) + c$ for all $x \in \Sigma^*$. By a simple counting argument, it follows that for any length n there exist strings x of length n with $CND(x) \geq n$, where $CND(x)$ is the minimal size of a nondeterministic Turing machine M with $L(M) = \{x\}$ (i.e., the time-unbounded *CND* measure).

Inspired by a similar result in [OKSW94], we now prove the following separations:

Theorem 3.4.5

1. *For every constant $c > 0$, $\mathsf{NP}/n^c \not\supseteq \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.*

2. $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}] \not\supseteq \mathsf{P}/\mathsf{lin}$.

Proof. For the first item, let $0 < c < d$ be natural numbers. Diagonalizing against all NP machines and all advice strings, we inductively define a set A with $A \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$, but $A \notin \mathsf{NP}/n^c$. Let $(N_i)_{i \in \mathbb{N}}$ be an enumeration of all NP machines, in which every machine occurs infinitely often. In step n we diagonalize against the machine N_n and every advice string of length $\leq n^c$ which N_n might use for length n. Let $x_1, \ldots, x_{2^n}$ be the lexicographic enumeration of all strings in Σ^n and let $S_n = \{x_1, \ldots, x_{n^d}\} \subseteq \Sigma^n$. For each string w of length at most n^c, let $A_w = \{x \in S_n : N_n(x) \text{ accepts under advice } w\}$. Since there are only 2^{n^c} such sets, but 2^{n^d} subsets of S_n, there must be one which is not equal to any A_w. For every n, let A_n be one such set, and let $A = \bigcup_n A_n$. By construction, $A \notin \mathsf{NP}/n^c$.

We still have to show $A \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. For each string s, let $\widetilde{s}$ be the substring of s which has all leading zeros deleted. For each n and each $a \in A_n$, let $M_{n,\widetilde{a}}$ be the following machine: on input x, the machine $M_{n,\widetilde{a}}$ checks whether $|x| = n$ and $\widetilde{x} = \widetilde{a}$. If this test is positive, then $M_{n,\widetilde{a}}$ accepts, otherwise it rejects. The machine $M_{n,\widetilde{a}}$ is of size $O(\log n)$, as both n and $\widetilde{a}$ are of length $O(\log n)$

(Observe that the first n^d elements in the lexicographic order of Σ^n have no 1's appearing before the last $\log n^d$ bits). Thus $A \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.

For the second item, let A be a set that contains exactly one element x per length with $CND(x) \geq |x|$. Obviously, $A \in \mathsf{P}/\mathsf{lin}$ because A contains exactly one string per length and this element can be given as advice. On the other hand, $A \notin \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. Assume on the contrary, that $A \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. Then there are a constant c and a polynomial p, such that for each $x \in A$, there is an A-consistent p-time-bounded machine M_x of size $\leq c \log |x| + c$ which accepts x. We modify M_x to a machine M'_x such that $L(M'_x) = \{x\}$ and $|M'_x| \leq c' \log |x| + c'$ for some constant c'. This machine M'_x works as follows: on input y, the machine M'_x first checks whether $|y| = |x|$. If not, it rejects. Otherwise, it simulates $M_x(y)$. Thus for all $x \in A$, $CND(x) \leq c' \log |x| + c'$, contradicting the choice of A. □

From Theorem 3.4.5 we infer that both inclusions in Proposition 3.4.3 are strict:

Corollary 3.4.6 $\mathsf{NP}/\mathsf{log} \subsetneq \mathsf{NIC}[\mathsf{log}, \mathsf{poly}] \subsetneq \mathsf{NP}/\mathsf{poly}$.

3.5 Promise Classes

Complexity classes are usually defined by a machine model on which resource bounds are imposed. A complexity class is *syntactic* if the machines can be appropriately standardized such that there exists an easy test which verifies that all these standardized machines define indeed languages from the complexity class (cf. [Pap94]). *Promise classes* (also called *semantic classes*) are the counterpart of syntactic classes because here we are lacking an easy syntactic machine model. Randomized classes like BPP or RP, the class of all proof systems for a given language, or disjoint NP-pairs are examples of promise classes.

Our basic model of computation are polynomial-time Turing machines and transducers. Tacitly we assume these machines to be suitably encoded by strings. We also assume that they always have a polynomial-time clock attached bounding their running time such that this running time is easy to detect from the code of the machine.

For a language L and a complexity class C, the set of all C-*easy subsets* of L consists of all sets $A \subseteq L$ with $A \in \mathsf{C}$. A class C of languages has a *recursive* P-*presentation* (resp. NP-presentation) if there exists a recursively enumerable list $N_1, N_2, \ldots$ of (non-)deterministic polynomial-time clocked Turing machines such that $L(N_i) \in \mathsf{C}$ for $i \in \mathbb{N}$, and, conversely, for each $A \in \mathsf{C}$ there exists an index i with $A = L(N_i)$. We remark that in this definition, for our purpose it would also suffice to replace $A = L(N_i)$ by the weaker requirement $A \subseteq L(N_i)$.

3.5.1 The General Concept of a Promise Class

Following the approach of Köbler, Messner, and Torán [KMT03], we define promise classes in a very general way. A promise R is described as a binary predicate between nondeterministic polynomial-time Turing machines N and strings x, i.e., $R(N, x)$ means that N obeys promise R on input x. A machine N is called an *R-machine* if N obeys R on any input $x \in \Sigma^*$. Given a promise predicate R, we define the language class $\mathsf{C}_R = \{L(N) \mid N \text{ is an R-machine}\}$ and call it the promise class generated by R. Instead of R-machines we will also speak of C_R-machines. Similarly, we define function promise classes by replacing $L(N)$ by the function computed by N (cf. [KMT03]). For functions we use the following variant of many-one reductions (cf. [KMT03]): $f \leq g$ if there exists a polynomial-time computable function t such that $f(x) = g(t(x))$ for all x in the domain of f.

In this general framework it is natural to impose further restrictions on promise classes. One assumption which we will make throughout this work is the presence of *universal machines*, i.e., we only consider promise conditions R such that there exists a universal machine U_R which, given an R-machine N, input x, and time bound 0^m, efficiently simulates $N(x)$ for m steps such that U_R obeys promise R on $\langle N, x, 0^m \rangle$.

Occasionally, we will need that C-machines can perform nondeterministic polynomial-time computations without violating the promise. We make this precise via the following notion from [KMT03]: for a complexity class A and a promise class C defined via promise R, we say that A-*assertions are useful for* C if for any language $A \in \mathsf{A}$ and any nondeterministic polynomial-time Turing machine N the following holds: if N obeys promise R on any $x \in A$, then there exists a language $C \in \mathsf{C}$ such that $C \cap A = L(N) \cap A$. A similar definition also applies for function classes. Namely, A-assertions are useful for a function class C if for any language $A \in \mathsf{A}$ and any polynomial-time clocked Turing transducer N it holds: if N obeys promise R on any input $x \in A$, then there exists a function $f \in C$ such that $N(x) = f(x)$ for any $x \in A$. In the following we will only consider promise classes C for which P-assertions are useful. If also NP-assertions are useful for C, then we say that C *can use nondeterminism*.

The set of all proof systems for a language L is an example for a promise function class, where the promise for a given function f is $rng(f) = L$. We define a larger class $PS(L)$ where we only concentrate on correctness but not on completeness of proof systems. This is made precise in the following definition.

Definition 3.5.1 *For a language L, the promise function class $PS(L)$ contains all polynomial-time computable functions f with $rng(f) \subseteq L$.*

3.5.2 Representations

In order to verify a promise, we need appropriate encodings of promise conditions. In the next definition we explain how a promise condition for a machine can be expressed in an arbitrary language.

Definition 3.5.2 *A promise R is* expressible in a language L *if there exists a polynomial-time computable function $corr : \Sigma^* \times \Sigma^* \times 0^* \to \Sigma^*$ such that the following conditions hold:*

1. Correctness: *For every Turing machine N, for every $x \in \Sigma^*$ and $m \in \mathbb{N}$, if $corr(x, N, 0^m) \in L$, then N obeys promise R on input x.*
2. Completeness: *For every R-machine N with polynomial time bound p, the set*
$$Correct(N) = \{ corr(x, N, 0^{p(|x|)}) \mid x \in \Sigma^* \}$$
is a subset of L.
3. Local recognizability: *For every Turing machine N, the set $Correct(N)$ is polynomial-time decidable.*

We say that the promise class C generated by R is expressible in L if R is expressible in L. If the elements $corr(x, N, 0^m)$ only depend on $|x|$, N, and m, but not on x, we say that C is expressible in L by a length-depending *promise.*

This definition applies to both language and function promise classes. One of the most important applications for the above concept of expressibility is to choose L as the set of propositional tautologies TAUT. Expressing promise conditions by propositional tautologies is a well known approach with a long history. For propositional proof systems, leading to the promise function class PS(TAUT), propositional expressions are constructed via the reflection principle of the proof system (cf. [Coo75, KP89]). Propositional expressions have also been used for other promise classes like disjoint NP-pairs and its generalizations [Bey07, BKM09]. Typically, these expressions are even length depending. We remark that Köbler, Messner, and Torán [KMT03] have used a related approach, namely the notion of a test set, to measure the complexity of promise conditions.

As a first example, consider the set of all P-easy subsets of a language L. The next lemma shows that this promise class is always expressible in L.

Lemma 3.5.3 *For every language L, the P-easy subsets of L are expressible in L.*

Proof. Let N be a deterministic polynomial-time Turing machine with running time p. We define the function $corr(N, x, 0^m)$ as

$$corr(x, N, 0^m) = \begin{cases} x & \text{if } N(x) \text{ accepts in} \leq m \text{ steps} \\ x_0 & \text{otherwise} \end{cases}$$

with some fixed element $x_0 \notin L$. □

Using expressibility of a promise class in a language L, we can verify the promise for a given machine with the help of short proofs in some proof system for L. This leads to the following concept:

Definition 3.5.4 *Let* C *be a promise class which is expressible in a language* L*. Let further* A *be a language from* C *and* P *be a proof system for* L*. We say that* A *is* representable in P *if there exists a* C*-machine* N *for* A *such that* $P \vdash_* Correct(N)$*. If these* P*-proofs of* $corr(x, N, 0^{p(|x|)})$ *can even be constructed from input* x *in polynomial time, then we say that* A *is* p-representable in P.

Furthermore, if every language $A \in \mathsf{C}$ *is (p-)representable in* P*, then we say that* C *is (p-)representable in* P*.*

Intuitively, representability of A in P means that we have short P-proofs of the promise condition of A (with respect to some C-machine for A). Given a proof system P for L and a promise class C which is expressible in L, it makes sense to consider the subclass of all languages or functions from C which are representable in P. This leads to the following definition:

Definition 3.5.5 *For a promise class* C *expressible in a language* L *and a proof system* P *for* L*, let* $\mathsf{C}(P)$ *denote the class of all* $A \in \mathsf{C}$ *which are representable in* P*.*

Note that for each $A \in \mathsf{C}$ there exists some proof system P for L such that $A \in \mathsf{C}(P)$, but in general $\mathsf{C}(P)$ will be a strict subclass of C which enlarges for stronger proof systems.

3.6 Optimal Proof Systems and Easy Subsets

In this section we search for characterizations for the existence of optimal or even p-optimal proof systems for arbitrary languages L and apply these results to concrete choices for L. We start with a criterion for the existence of p-optimal proof systems.

Theorem 3.6.1 *Let* L *be a language such that* $PS(L)$ *is expressible in* L*. Then* L *has a p-optimal proof system if and only if the* P*-easy subsets of* L *have a recursive* P*-presentation.*

Proof. Let f be a p-optimal proof system for L and let A be a polynomial-time computable subset of L. We can define a proof system f_A for L as follows:

$$f_A(x) = \begin{cases} f(y) & \text{if } x = 0y \\ a & \text{if } x = 1a \text{ and } a \in A \\ b & \text{otherwise} \end{cases}$$

where b is a fixed element in L. Because f is p-optimal, f_A is p-simulated by f via some polynomial-time computable function t_A.

As this can be done for all P-easy subsets A of L, we get a recursive P-presentation of L as follows. Let $(t_i)_{i\in\mathbb{N}}$ be an enumeration of all deterministic polynomial-time clocked Turing transducers. For $i \in \mathbb{N}$ consider the following set of algorithms M_i:

```
Input: $x$
IF $f(t_i(1x)) = x$ THEN accept ELSE reject
```

Apparently, these algorithms M_i can be computed by deterministic polynomial-time Turing machines. Further, each M_i only accepts inputs from L because if M_i accepts x, then we have an f-proof for x.

Now for each P-easy subset A of L, some machine computing the above function t_A appears in the enumeration t_i, and therefore A is accepted by M_i for the appropriate index i such that t_i computes t_A. Therefore M_i is a recursive P-presentation of the class of all P-easy subsets of L.

For the converse direction, let $(M_i)_{i\in\mathbb{N}}$ be a recursive P-presentation of the P-easy subsets of L. We construct a p-optimal proof system P_{opt} for L as follows. Inputs for P_{opt} are tuples

$$\langle \pi, P, 0^m, i, 0^n \rangle \ .$$

On such an input, P_{opt} first checks whether P is the encoding of a Turing transducer with a polynomial-time bound attached. If this is not the case, then P_{opt} outputs some fixed element $x_0 \in L$. Otherwise, P_{opt} spends n steps to compute the machine M_i from the enumeration $M_1, M_2 \dots$ If n steps do not suffice to construct M_i, we output again $x_0 \in L$. Otherwise, P_{opt} computes $corr(\pi, P, 0^m)$ and checks whether M_i accepts $corr(\pi, P, 0^m)$ in n steps. Again, if M_i does not stop in $\leq n$ steps, then we output x_0. If P and π pass the test, then P_{opt} simulates P on input π and outputs $P(\pi)$.

Apparently, P_{opt} can be computed in polynomial time. For each Turing transducer N with running time p and each input x with $N(x) \in L$, the element $corr(x, N, 0^{p(|x|)})$ is contained in some polynomial-time computable subset of L. Therefore, P_{opt} is a proof system for L, because by the correctness and completeness conditions from Definition 3.5.2, the range of P_{opt} is exactly L.

To prove the p-optimality of P_{opt}, let P be a proof system for L. Because by assumption $PS(P)$ is expressible in L, the set $Correct(P)$ is a P-easy subset of L (by the local recognizability condition from Definition 3.5.2). Hence there exists an index i such that M_i decides $Correct(P)$. Let c be a constant such that M_i can be computed from i in time c and let p and q be polynomial time bounds for P and M_i, respectively. Then P is easily seen to be p-simulated by

$$\pi \mapsto \langle \pi, P, 0^{p(|\pi|)}, i, 0^{q(|corr(\pi,P,0^{p(|\pi|)})|)+c} \rangle$$

which completes the proof. □

By a similar argument we can provide two characterizations for the existence of optimal proof systems.

Theorem 3.6.2 *Let L be a language such that $PS(L)$ is expressible in L. Then the following conditions are equivalent:*

1. *There exists an optimal proof system for L.*
2. *The NP-easy subsets of L have a recursive NP-presentation.*
3. *The P-easy subsets of L have a recursive NP-presentation.*

Given these general results, it is interesting to ask for which languages L the set $PS(L)$ of all proof systems for L is expressible in L. Our next lemma provides sufficient conditions:

Lemma 3.6.3 *Let L be a language fulfilling the following two conditions:*

1. *Natural numbers can be encoded by elements of L, i.e., there exists an injective function $Num : \mathbb{N} \to L$ which is both computable and invertible in polynomial time.*
2. *L possesses an AND-function, i.e., there exists a function $AND : \Sigma^* \times \Sigma^* \to \Sigma^*$ which is both polynomial-time computable and polynomial-time invertible such that for all $x, y \in \Sigma^*$, $AND(x, y) \in L$ if and only if $x \in L$ and $y \in L$.*

Then $PS(L)$ is expressible in L.

Proof. We have to define the function *corr* according to Definition 3.5.2. Given a string x, an encoding of a polynomial-time computable Turing transducer N, and a number $m \in \mathbb{N}$, we first simulate $N(x)$ for $\leq m$ steps. Let y be the output of $N(x)$, if the simulation succeeded. Otherwise, we choose a fixed string $y \notin L$.

Next we interpret the binary encoding of N as a natural number (which we again denote by N) and compute $Num(N)$. We then define the function *corr* as

$$corr(x, N, 0^m) = AND(y, AND(Num(N), Num(m))) \ .$$

Clearly, *corr* is polynomial-time computable. To verify the conditions of Definition 3.5.2 for *corr*, we observe that correctness and completeness of *corr* follow because the string $w := AND(Num(N), Num(m))$ is contained in L for all $N, m \in \mathbb{N}$, and therefore $AND(y, w) \in L$ if and only if $y \in L$.

Local recognizability for *corr* follows as AND and Num are invertible in polynomial time and therefore for each polynomial-time Turing transducer N, the set $Correct(N)$ is in P. □

Using this lemma we can show L-expressibility of $PS(L)$ for many interesting choices of L:

Proposition 3.6.4 *For any of the following languages L, the set $PS(L)$ is expressible in L:*

- SAT_i *for $i \in \mathbb{N}$ (the satisfiability problem for quantified propositional formulas with i quantifier alternations, starting with existential quantifiers),*
- TAUT_i *for $i \in \mathbb{N}$ (quantified propositional tautologies with i quantifier alternations, starting with universal quantifiers),*
- QBF *(quantified propositional tautologies),*
- *the graph isomorphism problem* GI*, its complement* $\overline{\mathrm{GI}}$*, and the complement* $\overline{\mathrm{GA}}$ *of the graph automorphism problem.*

Proof. We have to check the conditions from the previous lemma. For languages consisting of formulas like SAT_i, TAUT_i, or QBF, the AND-function is provided by the Boolean connective $\wedge$. The function *Num* can be defined for example by

$$n \mapsto (p \vee \neg p) \wedge \cdots \wedge (p \vee \neg p) \qquad (n \text{ times}) \enspace ,$$

where p is a fixed propositional variable.

For GI, $Num(n)$ can be implemented by a pair (K_n, K_n) of cliques of size n. For $\overline{\mathrm{GI}}$ we take (K_n, K_{n+1}), and for $\overline{\mathrm{GA}}$ we take an easy rigid graph with n vertices. It is well known that GI has an AND-function (cf. [KST93]). For the AND-functions of $\overline{\mathrm{GI}}$ and $\overline{\mathrm{GA}}$ we can take the OR-functions of GI and GA (cf. [KST93]). □

For GI, which like any problem in NP has an optimal proof system, we obtain the following characterization on the existence of a p-optimal proof system.

Corollary 3.6.5 GI *has a p-optimal proof system if and only if there exists a recursive* P*-presentation of all polynomial-time computable subsets of* GI*.*

Let us remark that in Lemma 3.6.3, instead of an AND-function we could also use a padding function for L. In this way we obtain a similar result as Corollary 3.6.5 for GA (which is not known to possess an AND-function).

Chapter 4

Proof Systems that Take Advice

> Altes Fundament ehrt man, darf aber das Recht nicht aufgeben, irgendwo wieder einmal von vorn zu gründen.
>
> Johann Wolfgang Goethe

Propositional proof complexity studies the question how difficult it is to prove propositional tautologies. In the classical Cook-Reckhow model, proofs are verified in deterministic polynomial time [CR79]. While this is certainly the most useful setting for practical applications, it is nevertheless interesting to ask if proofs can be shortened when we provide more resources for their verification. In this direction, Cook and Krajíček [CK07] have recently initiated the study of proof systems which use advice for the verification of proofs. Their results show that, like in the classical Cook-Reckhow setting, these proof systems enjoy a close connection to theories of bounded arithmetic (cf. Chapter 5).

In this chapter we aim at a rigorous development of the theory of proof systems with advice. In our investigation we particularly focus on the following fundamental questions for this new model:

Q1: Given a language L, do there exist polynomially bounded proof systems with advice for L?
Q2: For propositional proof systems, does advice help to shorten proofs?
Q3: Do there exist optimal proof systems with advice for L?

The organization of the chapter is as follows. In Section 4.1 we introduce our general model for proof systems with advice. Sections 4.2, 4.3, and 4.4 contain our results on questions Q1, Q2, and Q3, respectively. In Section 4.5 we demonstrate that the connection between the existence of optimal proof systems and the existence of complete sets for promise classes as shown in [KMT03, BS11] also holds in the stronger advice model.

4.1 Proof Systems with Advice

Our general model of computation for proof systems f with advice is a polynomial-time Turing transducer with several tapes: an input tape containing the proof π, possibly several work tapes for the computation of the machine, an output tape where we output the proven element $f(\pi)$, and an advice tape containing the advice. We start with a quite flexible definition of proof systems with advice for arbitrary languages, generalizing the notion of propositional proof systems with advice of Cook and Krajíček [CK07].

Definition 4.1.1 *For a function $k : \mathbb{N} \rightarrow \mathbb{N}$, a proof system f for L is a* proof system with k bits of advice*, if there exist a polynomial-time Turing transducer M, an advice function $h : \mathbb{N} \rightarrow \Sigma^*$, and an advice selector function $\ell : \Sigma^* \rightarrow 1^*$ such that*

1. *ℓ is computable in polynomial time,*

2. *M computes the proof system f with the help of the advice h, i.e., for all $\pi \in \Sigma^*$, $f(\pi) = M(\pi, h(|\ell(\pi)|))$, and*

3. *for all $n \in \mathbb{N}$, the length of the advice $h(n)$ is bounded by $k(n)$.*

We will abbreviate the phrase "proof system with k bits of advice" by ps/k. For a class F of functions, we denote by ps/F the class of all ps/k with $k \in F$.

We say that f *uses k bits of input advice* if ℓ has the special form $\ell(\pi) = 1^{|\pi|}$. On the other hand, in case $\ell(\pi) = 1^{|f(\pi)|}$ for all π in the domain of f, then f is said to *use k bits of output advice.* By this definition, proof systems with input advice use non-uniform information depending on the length of the proof, while proof systems with output advice use non-uniform information depending on the length of the proven formula.

We note that proof systems with advice are a quite powerful concept, as for every language $L \subseteq \Sigma^*$ there exists a proof system for L with only one bit of advice. In contrast, the class of all languages for which proof systems without advice exist coincides with the class of all recursively enumerable languages.

The above definition of a proof system with advice allows a very liberal use of advice, in the sense that for each input, the advice string used is determined by the advice selector function ℓ. In Section 5.5 we will consider concrete proof systems arising from generalizations of the extended Frege system *EF* which indeed require this general framework with respect to the advice.

In the next proposition we observe that proof systems with input advice are already as powerful as our general model of proof systems with advice.

Proposition 4.1.2 *Let $k : \mathbb{N} \rightarrow \mathbb{N}$ be a monotone function, $L \subseteq \Sigma^*$, and f be a ps/k for L. Then there exists a proof system f' for L with k bits of input advice such that f and f' are p-equivalent.*

Proof. We choose a polynomial-time computable bijective pairing function $\langle \cdot, \cdot \rangle$ on $\mathbb{N}$ such that $\langle n_1, n_2 \rangle \geq n_1 + n_2$ for all numbers n_1 and n_2. Let f be a *ps*/k for L with advice function h and advice selector ℓ. We define a proof system f' for L with input advice as follows: on input π' of length n the function f' first computes the two unique numbers n_1 and n_2 such that $n = \langle n_1, n_2 \rangle$. It then interprets the first n_1 bits $\pi'_1 \ldots \pi'_{n_1}$ of π' as an f-proof π and checks whether $\ell(\pi) = 1^{n_2}$. If this is the case, $f'(\pi') = f(\pi)$, otherwise f' outputs a fixed element $x_0 \in L$. Obviously, $f'(\pi')$ is computable with advice $h(|\ell(\pi)|) = h(n_2)$ whose length is bounded by $k(n_2) \leq k(n)$. This shows that f' is a *ps*/k for L with input advice.

The p-simulation of f by f' is computed by the function $\pi \mapsto \pi' = \pi 1^m$ where $m = \langle |\pi|, |\ell(\pi)| \rangle - |\pi|$. The converse simulation $f' \leq_p f$ is given by

$$\pi' \mapsto \begin{cases} \pi = \pi'_1 \ldots \pi'_{n_1} & \text{if } |\pi'| = \langle n_1, n_2 \rangle \text{ and } \ell(\pi) = 1^{n_2} \\ \pi_0 & \text{otherwise,} \end{cases}$$

where π_0 is a fixed f-proof of x_0. □

4.2 Polynomially Bounded Proof Systems with Advice

For any language L, we now investigate the question whether L has a polynomially bounded proof system with advice. We obtain different characterizations of this question, depending on

- whether we use input or output advice,
- which amount of advice the proof system may use, and
- the complexity of the proven language L.

In Section 4.2.1 we analyze the situation for arbitrary languages and in Section 4.2.2 we obtain stronger results for languages in coNP.

4.2.1 Results for Arbitrary Languages

We first consider proof systems with output advice. Similarly as in the classical result by Cook and Reckhow [CR79], we obtain the following equivalence:

Theorem 4.2.1 *Let $L \subseteq \Sigma^*$ be a language and let $k : \mathbb{N} \to \mathbb{N}$ be a function. Then L has a polynomially bounded ps/k with output advice if and only if $L \in \mathsf{NP}/k$.*

Proof. For the forward implication, let P be a polynomially bounded ps/k with output advice for L and let p be a bounding polynomial for P. We construct an NP/k machine M which uses the same advice as P and decides L. On input x, the machine M guesses a P proof w of size $\leq p(|x|)$ and checks whether $P(w) = x$. If so, M accepts, otherwise M rejects.

For the backward implication, let N be an NP/k machine deciding L with advice function h. We define a proof system P for L with k bits of output advice. Again, both P and N use the same advice. On input $\pi = \langle w, x\rangle$ the proof system P checks whether w is an accepting computation of N on input x with advice $h(|x|)$. If so, then $P(\pi) = x$. Otherwise, $P(\pi)$ is undefined. □

Given this result, we can now concentrate on input advice. In view of Theorem 4.4.1 below, input advice appears to be a stronger concept than output advice (as we probably cannot expect a similar result as Theorem 4.4.1 for output advice, cf. Corollary 4.2.5 and Proposition 4.2.8 below for results supporting this claim). Surprisingly, the advantage of input advice seems to vanish when we allow a polynomial amount of advice.

Theorem 4.2.2 *Let $L \subseteq \Sigma^*$ be any language. Then L has a polynomially bounded ps/poly with output advice if and only if L has a polynomially bounded ps/poly with input advice.*

Proof. The forward direction is a simple application of Proposition 4.1.2.

For the backward implication, let f_{in} be a ps/poly with input advice for L bounded by some polynomial p. Let a_n be the polynomially length-bounded advice used by f_{in} on inputs of length n.

We define a polynomially bounded ps/poly f_{out} for L with output advice as follows. Inputs x for f_{out} are interpreted as pairs $x = \langle \pi, y\rangle$. If $|\pi| \leq p(|y|)$ and $f_{in}(\pi) = y$, then $f_{out}(x) = y$. Otherwise, f_{out} is undefined. The computation of f_{out} uses all advice strings for f_{in} up to length $p(|y|)$ as advice. This still results in polynomial-size output advice for f_{out}.

The system f_{out} is correct, because f_{in} is correct. It is complete, because every $y \in L$ has a proof π_y with $|\pi_y| \leq p(|y|)$, implying that $f_{out}(\langle \pi_y, y\rangle) = y$. Hence, f_{out} is a polynomially bounded ps/poly with output advice. □

By Theorems 4.2.1 and 4.2.2, the existence of polynomially bounded ps/poly with input advice for L is equivalent to $L \in \mathsf{NP}/\mathsf{poly}$. Next, we consider proof systems with only a logarithmic amount of advice. In this case, we get a similar equivalence as before, where the class $\mathsf{NP}/\mathsf{poly}$ is replaced by the instance complexity class $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.

Theorem 4.2.3 *For every language L the following conditions are equivalent:*

1. *L has a polynomially bounded $ps/1$ with input advice.*

2. *L has a polynomially bounded ps/log with input advice.*

3. *$L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.*

Proof. The implication *1 ⇒ 2* follows by definition.

To prove the implication *2 ⇒ 3*, let f be a polynomially bounded ps/log with input advice and bounding polynomial p. For each x we have to construct a program M which is consistent with L and correctly decides x. If $x \notin L$, then M can just always reject. If $x \in L$, then there exists an f-proof π of x of length $\leq p(|x|)$. Let a be the advice for f on inputs of length $|\pi|$. To construct the machine M for x, we hardwire the values of $|x|$, $|\pi|$, and a into M. On input y the machine M checks whether $|y| = |x|$. If not, it rejects. Otherwise M guesses an f-proof π' of length $|\pi|$ for y and verifies that $f(\pi') = y$ using the advice a. If this test is positive, then M accepts, otherwise M rejects. Clearly, M accepts exactly all elements from L of length $|x|$ which have f-proofs of length $|\pi|$. In particular, M accepts x. Additionally, M is a polynomial-time nondeterministic program of length at most $c + \log|x| + \log|\pi| + |a|$ for some constant c. Therefore $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.

For the remaining implication *3 ⇒ 1*, let us assume that there are a polynomial p and a constant c, such that for every x, $nic^p(x : L) \leq c \cdot \log(|x|) + c$. We define a polynomially bounded $ps/1$ f for L with input advice as follows. Proofs in f take the form $\pi = \langle x, w, 1^M \rangle$, where $\langle \cdot, \ldots, \cdot \rangle$ is a polynomial-time computable and length-injective tupling function. The advice for f certifies whether or not M is a polynomial-time Turing machine that is consistent with L. If this is the case and w is an accepting computation of M on input x, then $f(\pi) = x$. Otherwise, $f(\pi)$ is undefined. Note that in the proof π we described the machine M in tally form. Together with the length-injectivity of the tupling function this allows the advice to refer to the machine M (but not to the input x which is given in binary notation).

Now, since $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$, for every $x \in L$ there is an L-consistent Turing machine M_x with running time p which accepts x and $|M_x| \leq c \cdot \log|x| + c$. Thus every element $x \in L$ has a polynomial-size f-proof $\langle x, w, 1^{M_x} \rangle$ where w is an accepting path of $M_x(x)$. □

In fact, we can prove a more general version of the preceding theorem, where we replace polynomial upper bounds for the proof length by arbitrary upper bounds. In this way we obtain:

Theorem 4.2.4 *For any language L and any function $t : \mathbb{N} \to \mathbb{N}$, $t \in n^{\Omega(1)}$, the following conditions are equivalent:*

1. *L has a $t^{O(1)}$-bounded $ps/1$ with input advice.*

2. *L has a $t^{O(1)}$-bounded ps/log with input advice.*

Table 4.1: Languages with polynomially bounded proof systems

	input advice	output advice
ps/poly	$\mathsf{NP/poly}$	$\mathsf{NP/poly}$
ps/log	$\mathsf{NIC[log, poly]}$	$\mathsf{NP/log}$
$ps/1$	$\mathsf{NIC[log, poly]}$	$\mathsf{NP}/1$
$ps/0$	NP	

3. $L \in \mathsf{NIC}[O(\log t), t^{O(1)}]$.

For a language L we now consider the following three assertions:

A1: L has a polynomially bounded ps/log with output advice.
A2: L has a polynomially bounded ps/log with input advice.
A3: L has a polynomially bounded ps/poly with output advice.

By our results so far, assertions A1, A2, and A3 are equivalent to the statement that L is contained in the classes $\mathsf{NP/log}$, $\mathsf{NIC[log, poly]}$, and $\mathsf{NP/poly}$, respectively. As these classes form a chain of inclusions by Proposition 3.4.3, we get the implications A1 $\Rightarrow$ A2 $\Rightarrow$ A3 for every L. Moreover, by Corollary 3.4.6, the inclusions $\mathsf{NP/log} \subsetneq \mathsf{NIC[log, poly]} \subsetneq \mathsf{NP/poly}$ are proper. Hence we obtain:

Corollary 4.2.5 *There exist languages L for which* A2 *is fulfilled, but* A1 *fails. Likewise, there exist languages L for which* A3 *is fulfilled, but* A2 *fails.*

Table 4.1 provides an overview of our results on question Q1 obtained so far, showing which languages possess polynomially bounded proof systems with advice. It is interesting to note that all language classes appearing in this table form a chain of strict inclusions (cf. Corollary 3.4.6).

4.2.2 Polynomially Bounded Proof Systems for TAUT

From a practical point of view, it is most interesting to investigate what precisely happens for $L =$ TAUT (or more generally for problems in coNP). Even though by Corollary 3.4.6, $\mathsf{NP/log}$ and $\mathsf{NIC[log, poly]}$ are distinct, they do not differ inside coNP, as the next theorem shows.

Theorem 4.2.6 *Let $L \in \mathsf{coNP}$. Then $L \in \mathsf{NP/log}$ if and only if $L \in \mathsf{NIC[log, poly]}$. Moreover, if $L \in \mathsf{NP/log}$, then the advice can be computed in $\mathsf{FP}^{\mathsf{NP[log]}}$.*

Proof. By Proposition 3.4.3 we only have to prove the backward implication. For this let L be a language from coNP. Assuming $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$, there exists a polynomial p and a constant c such that $nic^p(x : L) \leq c \log |x| + c$ for all $x \in \Sigma^*$. Let Π^n be the set of all p-time bounded nondeterministic machines M with $|M| \leq c \log n + c$. Let further a_n be the number of machines from Π^n that are not consistent with $L \cap \Sigma^{\leq n}$. As the cardinality of Π^n is bounded by a polynomial in n, the length of the number a_n is logarithmic in n.

We now construct a nondeterministic Turing machine N that uses $c \log n + c + 1$ bits of advice for inputs of length n and decides L. The advice of N for input length n will be the number a_n. On input x of length n, the machine N nondeterministically chooses a_n pairwise distinct machines $M_1, \dots, M_{a_n} \in \Pi^n$ and strings $x_1, \dots, x_{a_n} \in \Sigma^{\leq n}$. Next, N verifies that $x_1, \dots, x_{a_n}$ do not belong to L. As $L \in \mathsf{coNP}$, this can be done in nondeterministic polynomial time. Then N checks whether for each $i = 1, \dots, a_n$ the machine M_i accepts the input x_i. If any of the tests so far failed, N rejects. Otherwise, if all these tests were positive, we know that every machine in $\Pi^n \setminus \{M_1, \dots, M_{a_n}\}$ is consistent with $L \cap \Sigma^{\leq n}$. After this verification has successfully taken place, N simulates all remaining machines $M \in \Pi^n \setminus \{M_1, \dots, M_{a_n}\}$ on input x. If one of these simulations accepts, then also N accepts x, otherwise N rejects.

Since there are only consistent machines left after a_n machines have been deleted, N never accepts any $x \notin L$. On the other hand, the assumption $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$ guarantees that for every $x \in L$ there is a machine in Π^n which is consistent with L and accepts x. Therefore N correctly decides L, and thus $L \in \mathsf{NP}/\mathsf{log}$, as claimed.

For the additional claim in the theorem, we again use the above construction. Starting from a language $L \in \mathsf{NP}/\mathsf{log}$, we first obtain the collection of machines by Proposition 3.4.3, witnessing $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. This collection of machines is then transformed again into an NP-machine with logarithmic advice by the construction above. By this procedure we bring both the machine as well as the advice into a well-defined normal form. Now it suffices to observe that using binary search we can compute the advice a_n with at most logarithmically many queries of the form "Do there exist at least m logarithmic-size machines which are inconsistent with $L \cap \Sigma^{\leq n}$?" As this is an NP question, the advice can be computed in $\mathsf{FP}^{\mathsf{NP}[\mathsf{log}]}$. □

By Theorem 4.2.2 we already know that an arbitrary language L has a polynomially bounded ps/poly with input advice if and only if L has a polynomially bounded ps/poly with output advice. As a corollary to Theorem 4.2.6 we obtain the same equivalence for logarithmic advice, but only for coNP languages.

Corollary 4.2.7 *Let L be a language from coNP. Then L has a polynomially bounded ps/log with input advice if and only if L has a polynomially bounded ps/log with output advice.*

Table 4.2: Consequences of the existence of polynomially bounded proof systems

Assumption *if* TAUT *has a polynomially bounded . . .*		**Consequence** *then* PH *collapses to . . .*
ps/poly	(input or output advice)	$\mathsf{S}_2^{\mathsf{NP}} \subseteq \Sigma_3^{\mathsf{p}}$
ps/log	(input or output advice)	$\mathsf{P}^{\mathsf{NP}[\mathsf{log}]}$
$ps/O(1)$	(input advice)	$\mathsf{P}^{\mathsf{NP}[\mathsf{log}]}$
$ps/O(1)$	(output advice)	$\mathsf{P}^{\mathsf{NP}[O(1)]} = \mathsf{BH}$
$ps/0$	(no advice)	NP

Descending to constant advice, this equivalence seems to fail. Using Theorem 3.3.2 we prove that the assertions of the existence of polynomially bounded proof systems with input and output advice appear to be of different strength, as otherwise the equivalence of two collapses of PH of presumably different strength follows.

Proposition 4.2.8 *Assume that* TAUT *having a polynomially bounded* $ps/1$ *with input advice implies that* TAUT *has a polynomially bounded* $ps/1$ *with output advice. Then* $\mathsf{PH} \subseteq \mathsf{BH}$ *already implies* $\mathsf{PH} \subseteq \mathsf{D^p}$.

Proof. If the polynomial hierarchy collapses to the Boolean hierarchy, then PH in fact collapses to some level BH_k of BH. By Theorem 3.3.2, this means that $\mathsf{coNP} \subseteq \mathsf{NP}/k'$ for some constant k'. Hence by Theorem 4.2.1, TAUT has a polynomially bounded ps/k' P with output advice. By a result of Cook and Krajíček [CK07] (cf. also Theorem 4.4.1 below), this proof system P is simulated by a proof system P' which only uses 1 bit of input advice. As P is polynomially bounded, this is also true for P'. By our assumption, TAUT also has polynomially bounded $ps/1$ with output advice. By Theorem 4.2.1 this implies $\mathsf{coNP} \subseteq \mathsf{NP}/1$ and therefore $\mathsf{PH} \subseteq \mathsf{D^p}$ by Theorem 3.3.2. □

We remark that in Proposition 4.2.8 we have to state the result for TAUT or some other coNP-complete set, but do not obtain the same statement for any coNP language.

So far we have provided different characterizations of question Q1 whether polynomially bounded proof systems with advice exist. At this point it is natural to ask, how likely these assumptions actually are, i.e., what consequences follow from the assumption that such proof systems exist. For TAUT we obtain a series of collapse consequences of presumably different strength as shown in Table 4.2.

The first line in Table 4.2 follows from Theorems 4.2.1 and 4.2.2 and a result of Cai, Chakaravarthy, Hemaspaandra, and Ogihara [CCHO05], who have shown that $\mathsf{coNP} \subseteq \mathsf{NP/poly}$ implies $\mathsf{PH} \subseteq \mathsf{S}_2^{\mathsf{NP}}$. For the second line, the distinction between input and output advice is again irrelevant (Corollary 4.2.7). Here we use a result of Arvind, Köbler, Mundhenk, and Torán [AKMT00], who showed that TAUT $\in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$ implies $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}[\mathsf{log}]}$. Finally, the constant-advice case (lines 3 and 4) follows from Theorem 3.3.2 in conjunction with Theorems 4.2.1 and 4.2.3. In comparison, the classical Cook-Reckhow Theorem states that TAUT has an advice-free polynomially bounded proof system if and only if $\mathsf{PH} \subseteq \mathsf{NP}$ (line 5).

4.3 Simplifying the Advice in Propositional Proof Systems

In this section we again concentrate on propositional proof systems and prove results which contribute to an answer to question Q2. Apart from TAUT, our results proved here generalize to other languages, but different properties of TAUT play a role in these generalizations.

4.3.1 Transferring Advice from the Proof to the Formula

There are two natural ways to enhance proof systems with advice by either supplying non-uniform information to the proof (input advice) or to the proven formula (output advice). Intuitively, input advice is the stronger model: proofs can be quite long and formulas of the same size typically require proofs of different size. Hence, supplying advice depending on the proof size is not only more flexible, but also results in more advice per formula. This view is also supported by previous results: there exist optimal proof systems with input advice [CK07] (see also Theorem 4.4.1 below), whereas for output advice a similar result cannot be obtained by current techniques (cf. 4.4.2 below). Further evidence is provided by the existence of languages that have polynomially bounded proof systems with logarithmic input advice, but do not have such systems with output advice (Corollary 4.2.5).

In our next result we show how input advice can be transformed into output advice. We obtain this simplification of advice under the assumption of weak, but non-trivial upper bounds to the proof size. More precisely, from a propositional proof system which uses logarithmic input advice and has sub-exponential size proofs of all tautologies, we construct a system with polynomial output advice which obeys almost the same upper bounds. The result holds for TAUT and more generally for all languages L which have a polynomial-time computable AND-function:

Definition 4.3.1 *A language L possesses a* linear AND-function *if there exists a function AND which is both polynomial-time computable and polynomial-time* *invertible such that for all $x_1, \dots, x_n \in \Sigma^*$, $AND(x_1, \dots, x_n) \in L$ if and only if $x_i \in L$ for all $i = 1, \dots, n$. We further require that $|AND(x_1, \dots, x_n)| \leq d \cdot \sum_{i=1}^{n} |x_i|$ for some constant d.*

For the proof of the next result we use a new technique by Buhrman and Hitchcock [BH08] who show that sets of sub-exponential density are not NP-hard unless coNP ⊆ NP/poly.

Theorem 4.3.2 *Let L be a language with a linear AND-function. Let further $t(n) \in 2^{O(\sqrt{n})}$ and let f be a $t(n)$-bounded proof system for L with polylogarithmic input advice. Then there exists an $s(n)$-bounded proof system g for L with polynomial output advice where $s(n) \in O(t(d \cdot n^2))$ and where d is the constant from Definition 4.3.1.*

Proof. Let $t(n) \leq 2^{c \cdot \sqrt{n}}$ for some constant c and let f be a $t(n)$-bounded propositional proof system with polylogarithmic input advice. We say that π is a *conjunctive f-proof* for a string $x \in L$ if there exist strings $y_1, \dots, y_n$ with $|y_i| = |x| = n$ such that $f(\pi) = AND(y_1, \dots, y_n)$ and x is among the y_i. For a number $m \geq 1$, we denote by $\sharp_m^n$ the number of strings $x \in L \cap \Sigma^n$ which have conjunctive f-proofs of size exactly m. By counting we obtain

$$(\sharp_m^n)^n \geq |\{\, (x_1, \dots, x_n) \mid AND(x_1, \dots, x_n) \text{ has an } f\text{-proof of size } m \text{ and } |x_i| = n \text{ for } 1 \leq i \leq n \,\}| \ . \tag{4.1}$$

As f is t-bounded, every $x \in L \cap \Sigma^n$ has a conjunctive f-proof of size at most $t(d \cdot n^2)$ where d is the constant from the AND-function of L as in Definition 4.3.1. Let $\sharp^n = \max\{\, \sharp_m^n \mid m \leq t(d \cdot n^2) \,\}$. Using (4.1) we obtain

$$\begin{aligned} |L \cap \Sigma^n|^n \leq \sum_{m=1}^{t(d \cdot n^2)} (\sharp_m^n)^n \ &\leq \ (\sharp^n)^n \cdot t(d \cdot n^2) \\ \leq (\sharp^n)^n \cdot 2^{c \cdot \sqrt{d \cdot n^2}} \ &= \ (\sharp^n \cdot 2^{c \cdot \sqrt{d}})^n \ . \end{aligned}$$

Thus, setting $\delta = 2^{-c \cdot \sqrt{d}}$, we get $\sharp^n \geq \delta \cdot |L \cap \Sigma^n|$. Therefore, by definition of $\sharp^n$ there exists a number $m_{n,0} \leq t(d \cdot n^2)$ such that $\sharp_{m_{n,0}}^n \geq \delta \cdot |L \cap \Sigma^n|$, i.e., a δ-fraction of all strings from L of length n has a conjunctive f-proof of size $m_{n,0}$.

Consider now the set $L_0^{=n}$ of all strings from L of length n which do not have conjunctive f-proofs of size $m_{n,0}$. Repeating the above argument for $L_0^{=n}$ yields a proof length $m_{n,1} \leq t(d \cdot n^2)$ such that $\sharp_{m_{n,1}}^n \geq \delta \cdot |L_0^{=n}|$. Iterating this argument we obtain a sequence $m_{n,0}, m_{n,1}, \dots, m_{n,\ell(n)}$, where

$$\ell(n) = \left\lceil \frac{\log |L \cap \Sigma^n|}{\log(1-\delta)^{-1}} \right\rceil \leq \left\lceil \frac{n}{\log(1-\delta)^{-1}} \right\rceil ,$$

such that every $x \in L \cap \Sigma^n$ has a conjunctive f-proof of size $m_{n,i}$ for some $i \in \{0, \dots, \ell(n)\}$.

We will now define a proof system g which uses polynomial output advice and obeys the same proof lengths as f. Assume that f is computed by the polynomial-time Turing transducer M_f with advice function h_f. The system g will be computed by a polynomial-time Turing transducer M_g using the advice function

$$h_g(n) = \left\langle m_{n,0}, h_f(m_{n,0}), \ldots, m_{n,\ell(n)}, h_f(m_{n,\ell(n)}) \right\rangle \ .$$

The machine M_g works as follows: On input π', M_g first checks whether the proof π' has the form

$$\langle x, y_1, \ldots, y_n, \pi, i \rangle \ ,$$

where $x, y_1, \ldots, y_n$ are strings of length n such that $x \in \{y_1, \ldots, y_n\}$, π is a string (an f-proof), and i is a number $\leq \ell(n)$. If this test fails, then $g(\pi')$ is undefined. Then M_g uses its advice to check whether $|\pi| = m_{n,i}$. If so, then M_g simulates M_f on input π using advice $h_f(m_{n,i})$ (which is available through the advice function h_g). If the output of this simulation is $AND(y_1, \ldots, y_n)$, then M_g outputs x, otherwise $g(\pi')$ is undefined.

By our analysis above, g is a proof system for L (it is correct and complete). The advice only depends on the length n of the proven string, so g uses output advice. To estimate the advice length, let $|h_f(m)| \leq \log^a m$ for some constant a. Then

$$|h_g(n)| \leq \sum_{i=0}^{\ell(n)} (|m_{n,i}| + |h(m_{n,i})|) \leq (\ell(n)+1)\left(n/\delta + \log^a(2^{n/\delta})\right) = n^{O(1)} \ .$$

The size of a g-proof $\langle x, y_1, \ldots, y_n, \pi, i \rangle$ for $x \in L \cap \Sigma^n$ is dominated by $|\pi| \leq t(d \cdot n^2)$, and therefore g is $s(n)$-bounded for some $s(n) \in O(t(d \cdot n^2))$. □

In some sense, Theorem 4.3.2 transfers the results of Theorem 4.2.2 and Corollary 4.2.7 to super-polynomial proof lengths. However, while Theorem 4.2.2 has an easy proof and holds for all languages, the last construction is rather non-trivial and uses the assumption that L has a linear AND-function.

4.3.2 Substituting Advice by Weak Oracles

From a practical point of view, proof systems with advice are susceptive to criticism: advice can be arbitrarily complex (even non-recursive) and thus verifying proofs with the help of advice does not form a feasible model to use in practice. Our next result shows that for propositional proof systems, logarithmic advice can be replaced by a sparse NP-oracle without increasing the proof length.

Theorem 4.3.3 *Let L be a language from* coNP. *Then the following holds:*

1. *Every proof system for L with logarithmic advice is simulated by a proof system for L computable in polynomial time with access to a sparse* NP-*oracle.*

2. *Conversely, every proof system for L computable in polynomial time with access to a sparse* NP*-oracle is simulated by a proof system for L with logarithmic advice.*

Proof. For the first claim, let f be a proof system for L computed by the polynomial-time Turing transducer M_f with advice function h_f where $|h_f(n)| \leq c{\cdot}\log n$ for some constant c. Without loss of generality, we may assume that f uses input advice (Proposition 4.1.2). We choose a length-injective polynomial-time computable pairing function $\langle\cdot\rangle$ and consider the set

$$A = \left\{ \langle 1^n, a\rangle \mid a \in \Sigma^{\leq c\cdot\log n} \text{ and for some } \pi \in \Sigma^n,\ M_f(\pi, a) \notin L \right\} ,$$

where $M_f(\pi, a)$ denotes the output of M_f on input π using advice a. Intuitively, A collects all incorrect advice strings for M_f on length n. By construction, A is sparse. Further, $A \in \mathsf{NP}$ because on input $\langle 1^n, a\rangle$ we can guess $\pi \in \Sigma^n$ and nondeterministically verify $M_f(\pi, a) \notin L$. Because $L \in \mathsf{coNP}$ this verification is possible in nondeterministic polynomial time.

We now construct a polynomial-time oracle Turing transducer M_g which under oracle A computes a proof system $g \geq f$. Proofs in g will be of the form $\langle \pi, \varphi\rangle$. On such input, M_g queries all strings $\langle 1^{|\pi|}, a\rangle$, $a \in \Sigma^{\leq c\cdot\log|\pi|}$. For each negative answer, M_g simulates M_f on input π using a as advice. If any of these simulations outputs φ, then M_g also outputs φ, otherwise $g(\langle \pi, \varphi\rangle)$ is undefined. Because M_g performs at most polynomially many simulations of M_f, the machine M_g runs in polynomial time. Correctness and completeness of g follow from the fact that M_f is simulated with all correct advice strings, and the original advice used by M_f is among these (as also other advice strings are used, g might have shorter proofs than f, though).

For the second claim, let f be a proof system for L computed by the oracle transducer M_f under the sparse NP-oracle A. Let M_A be an NP-machine for A and let $p(n)$ be a polynomial bounding the cardinality of $A \cap \Sigma^{\leq n}$ as well as the running times of M_A and M_f. With these conventions, there are at most $q(n) = p(p(n))$ many strings in A that M_f may query on inputs of length n.

We now define a machine M_g, an advice function h_g, and an advice selector ℓ_g which together yield a proof system $g \geq f$ for L with logarithmic advice. The advice function will be $h_g(n) = |A \cap \Sigma^{\leq p(n)}|$. As A is sparse this results in logarithmic advice. Proofs in the system g are of the form

$$\pi_g = \left\langle a_1, \ldots, a_{q(n)}, w_1, \ldots, w_{q(n)}, \pi_f \right\rangle$$

where $\pi_f \in \Sigma^n$ (an f-proof), $a_1, \ldots, a_{q(n)} \in \Sigma^{\leq p(n)}$ (elements from A), and $w_1, \ldots, w_{q(n)} \in \Sigma^{\leq q(n)}$ (computations of M_A). Given such a proof π_g, the advice selector chooses the advice corresponding to $|\pi_f|$, i.e., we set $\ell_g(\pi_g) = |\pi_f|$. The machine M_g works as follows: it first uses its advice to obtain the number $m = h_g(|\pi_f|)$ and checks whether $a_1, \ldots, a_m$ from the proof π_g are pairwise distinct and for each $i = 1, \ldots, m$, the string w_i is an accepting computation of M_A on input a_i. If all these simulations succeed, then we know that

$A \cap \Sigma^{\leq p(n)} = \{a_1, \ldots, a_m\}$. Hence M_g can now simulate M_f on π_f and give correct answers to all oracle queries made in this computation. □

As a consequence, we get the following simplicity result stating that we can bound the complexity of the non-uniform part (the advice) when proving coNP languages:

Corollary 4.3.4 *Every ps/*log *f for a language in* coNP *is simulated by a ps/*log *g whose advice function h is computable in* $\mathsf{FP}^{\mathsf{NP} \cap \mathsf{Sparse}[\log]}$*, i.e., h is computable in polynomial time with a logarithmic number of queries to a sparse* NP*-oracle.*

Proof. The claim follows by first applying item 1 and then item 2 of Theorem 4.3.3 and observing that the advice function of the resulting proof system (denoted h_g in the proof above) is computable using binary search with logarithmically many questions to the sparse NP-set $\{\langle 1^m, 1^n \rangle \mid m \leq |A \cap \Sigma^{\leq p(n)}|\}$. □

We remark that the condition $L \in \mathsf{coNP}$ in Theorem 4.3.3 is only needed for the first item whereas the second item holds for all languages. Further, by an easy modification in the above proofs it follows that instead of a sparse NP-set it also suffices to use a tally NP-set as the oracle. Let us remark that Balcázar and Schöning [BS92] have shown a similar trade-off between advice and oracle access in complexity theory: $\mathsf{coNP} \subseteq \mathsf{NP}/\mathsf{log}$ if and only if $\mathsf{coNP} \subseteq \mathsf{NP}^S$ for some sparse $S \in \mathsf{NP}$. We complete the picture by showing that the simulations in the previous theorem cannot be strengthened to a full equivalence between the two concepts:

Proposition 4.3.5 *For every nonempty language L there exist proof systems with constant advice which cannot be computed with access to a recursive oracle.*

Proof. Let us first consider the case that L is recursively enumerable and let f be a polynomial-time computable proof system for L. With each infinite sequence $a = (a_i)_{i \in \mathbb{N}}$, $a_i \in \{0, 1\}$, we associate the proof system

$$f_a(\pi) = \begin{cases} f(\pi') & \text{if either } \pi = 0\pi' \text{ or } (\pi = 1\pi' \text{ and } a_{|\pi|} = 0) \\ \text{undefined} & \text{if } \pi = 1\pi' \text{ and } a_{|\pi|} = 1. \end{cases}$$

Because of the first line of its definition, f_a is a complete proof system for L. As different sequences a and b yield different proof systems f_a and f_b, there exist uncountably many different propositional proof systems with one bit of advice. On the other hand, there are only countably many proof systems computed by oracle Turing machines under recursive oracles. Hence the claim follows.

Now consider the case that L is not recursively enumerable. Yet, L has a proof system with one bit of advice which is computed by the machine M

$$M(w) = \begin{cases} x & \text{if } h(|w|) = 1 \text{ and } w = 1^x \text{ (the string } x \text{ coded in unary)} \\ \text{undef.} & \text{otherwise} \end{cases}$$

where h is the advice function for M defined as

$$h(n) = \begin{cases} 1 & \text{if } n = |1^x| \text{ and } x \in L \\ 0 & \text{otherwise,} \end{cases}$$

i. e., $h(n)$ is the characteristic function of L where the input is coded in unary. On the other hand, if L is not recursively enumerable, then L does not have a proof system which is computable in polynomial time under a recursive oracle. Hence the claim also holds in this case. □

For polynomial instead of logarithmic advice, we obtain a similar result as Theorem 4.3.3, but there are two differences. On the one hand, the result holds for arbitrary languages, whereas Theorem 4.3.3 only holds for languages in coNP. Also, we will now get a full equivalence between the two concepts (compare with Proposition 4.3.5). On the other hand, the oracle will still be sparse, but we cannot bound its complexity—it will be as complex as the original advice.

Proposition 4.3.6 *Let L be an arbitrary language and let f be a proof system for L. Then f is a ps/poly if and only if f can be computed in polynomial time with access to a sparse oracle.*

Proof. For the forward direction, let f be a proof system for L computed by the polynomial-time Turing transducer M_f with advice function h_f where $|h_f(n)| \leq p(n)$ for some polynomial p. We choose a length-injective polynomial-time computable pairing function $\langle \cdot \rangle$ and consider the set

$$A = \{ \langle 1^n, a \rangle \mid a \text{ is a prefix of } h_f(n) \} \quad .$$

Now, f can be computed in polynomial time with oracle access to A by first computing the relevant advice using prefix search and then simulating M_f.

Conversely, if f is computed in polynomial time $q(n)$ under a sparse oracle B, then f is computable by a *ps*/poly with input advice using as advice an encoding of the set $B \cap \Sigma^{\leq q(n)}$. □

4.4 Optimal Proof Systems

We now come to question Q3 on the existence of optimal proof systems. We recall from Section 2.2 that an *optimal* proof system for a language L simulates every other proof system for L.

4.4.1 Optimal Proof Systems with Advice

While in the classical setting, the existence of optimal proof systems is a prominent open question [KP89] (cf. Section 2.2 for a discussion), Cook and Krajíček [CK07] have shown that there exists a propositional proof system with one bit of input advice which simulates all classical Cook-Reckhow proof systems. The proof of this result easily generalizes to arbitrary languages L, thus yielding:

Theorem 4.4.1 *For every language L there exists a proof system P with one bit of input advice such that P simulates all ps/log for L. Moreover, P p-simulates all advice-free proof systems for L.*

Proof. Let $\langle \cdot, \dots, \cdot \rangle$ be a polynomial-time computable tupling function on Σ^* which is length injective, i.e., $|\langle x_1, \dots, x_n \rangle| = |\langle y_1, \dots, y_n \rangle|$ implies $|x_i| = |y_i|$ for $i = 1, \dots, n$. We define the proof system P as follows. P-proofs are of the form $w = \langle \pi, 1^T, 1^a, 1^m \rangle$ with $\pi, T, a \in \Sigma^*$ and $m \in \mathbb{N}$ (here 1^T and 1^a denote unary encodings of T and a, respectively).

The proof system P uses one bit $h(|w|)$ of advice, where $h(|w|) = 1$ if and only if the transducer T with advice a only outputs elements from L for inputs of length $|\pi|$. Note that by the length injectivity of $\langle \cdot, \dots, \cdot \rangle$, the advice bit can in fact refer to T, a, and $|\pi|$. Now, if $h(|w|) = 1$ and T on input π with advice a outputs y after at most m steps, then $P(w) = y$. Otherwise, $P(w)$ is undefined.

In case Q is a proof system computed by some polynomial-time transducer T without (i.e. zero bits of) advice, then Q is p-simulated by P via the polynomial-time computable function $\pi \mapsto \langle \pi, 1^T, 1^\varepsilon, 1^{p(|\pi|)} \rangle$, where p is a polynomial bound for the running time of T (and ε is the empty string). On the other hand, if T uses advice $h(|\ell(\pi)|)$ of at most logarithmic length, then Q is simulated by P via the function $\pi \mapsto \langle \pi, 1^T, 1^{h(|\ell(\pi)|)}, 1^{p(|\pi|)} \rangle$. □

It is a natural question whether we can improve this construction to obtain proof systems with output advice that still have the same optimality conditions. While we must leave this question open, our next result shows that it seems unlikely to give an affirmative answer with currently available techniques, as otherwise collapse assumptions of presumably different strength would be equivalent. This result indicates that, by current knowledge, input advice for propositional proof systems is indeed a more powerful concept than output advice.

Theorem 4.4.2 *Let $k \geq 1$ be a constant and assume that* TAUT *has a ps/k with output advice that simulates every ps/1 for* TAUT. *Then the following conditions are equivalent:*

1. *The polynomial hierarchy collapses to BH_{2^k}.*

2. *The polynomial hierarchy collapses to BH.*

3. $\mathsf{coNP} \subseteq \mathsf{NP}/\mathsf{log}$.

4. $\mathsf{coNP} \subseteq \mathsf{NP}/k$.

Proof. The equivalence of 1 and 4 was shown by Buhrman, Chang, and Fortnow (Theorem 3.3.2), and clearly, item 1 implies item 2. It therefore remains to prove the implications $2 \Rightarrow 3$ and $3 \Rightarrow 4$.

For the implication $2 \Rightarrow 3$, let us assume $\mathsf{PH} \subseteq \mathsf{BH}$. We choose a Σ_2^{p}-complete problem L, which by assumption is contained in $\mathsf{BH}_{k'}$ for some number k'. By Theorem 3.3.2 this implies $\mathsf{coNP} \subseteq \mathsf{NP}/k'$ and hence $\mathsf{coNP} \subseteq \mathsf{NP}/\mathsf{log}$.

For the final implication $3 \Rightarrow 4$, we assume $\mathsf{coNP} \subseteq \mathsf{NP}/\mathsf{log}$. By Theorem 4.2.1 this guarantees the existence of a polynomially bounded system P for TAUT with $O(\log n)$ bits of output advice. By Theorem 4.4.1, P is simulated by a proof system P' with only one bit of input advice. Hence also P' is polynomially bounded. Now we use the hypothesis of the existence of a proof system Q with k bits of output advice which simulates all $ps/1$ for TAUT. In particular, $P' \leq Q$ and therefore Q is a polynomially bounded ps/k with output advice. Using again Theorem 4.2.1 we obtain $\mathsf{coNP} \subseteq \mathsf{NP}/k$. □

With respect to the optimal proof system from Theorem 4.4.1 we obtain:

Corollary 4.4.3 *The optimal $ps/1$ for* TAUT *from Theorem 4.4.1 is not equivalent to a $ps/1$ with output advice, unless* $\mathsf{PH} \subseteq \mathsf{BH}$ *implies* $\mathsf{PH} \subseteq \mathsf{D}^{\mathsf{p}}$.

Of course, rather than indicating that proof systems with constant output advice cannot be optimal for the class of all ps/log, this corollary points towards our current limitations to disprove such a result.

Combining Theorems 4.3.3 and 4.4.1, we can reformulate the optimality result for coNP languages in the oracle model:

Corollary 4.4.4 *Let $L \in \mathsf{coNP}$. Then there exists a proof system f for L which simulates every polynomial-time computable proof system for L. The system f is computable in polynomial time under a sparse* NP*-oracle.*

Our next result shows that if advice does not help to shorten proofs (even for easy languages), then optimal propositional proof systems exist.

Theorem 4.4.5 *If every polynomially bounded proof system for a* coNP *set that uses one bit of output advice can be simulated by a proof system without advice, then the class of all polynomial-time computable propositional proof systems contains an optimal system.*

Proof. Book [Boo74] showed that $\mathsf{NE} = \mathsf{coNE}$ if and only if any tally set $A \in \mathsf{coNP}$ belongs to NP. The former, however, implies the existence of an optimal proof system by a result of Krajíček and Pudlák [KP89]. Therefore it suffices to show that the assumption implies that any tally set $A \in \mathsf{coNP}$ belongs to NP. Clearly, any tally set $A \in \mathsf{coNP}$ has a polynomially bounded proof system f with one bit of output advice because we can define $f(x) = 1^{|x|}$ if the advice $h(|x|)$ equals 1 and leave it undefined otherwise. Here, the advice function h is the characteristic function of the set $\{n \in \mathbb{N} \mid 1^n \in A\}$. Now let g be a proof system without advice that simulates f. Then it follows that g is polynomially bounded and hence $A \in \mathsf{NP}$. □

4.4.2 On p-optimal Proof Systems with Advice

In this section we will investigate the question whether there exist p-optimal propositional proof systems with advice. With respect to p-optimality, there are two natural options how to define this concept in the presence of advice: we may also allow the simulation functions to take advice or we can consider advice-free simulations. With respect to the first option, the proof of Theorem 4.4.1 immediately yields the following result:

Theorem 4.4.6 *For every language L there exists a proof system for L with one bit of input advice which p-simulates all proof systems for L with $k(n)$ bits of input advice for $k(n) = O(\log n)$. The p-simulation is computed by a polynomial-time algorithm using $k(n)$ bits of advice.*

In the above theorem, in addition to the proof system also the simulation functions are allowed to use advice. Our next result shows that for advice-free simulation functions such a result does not hold. We just state the result for propositional proof systems but the proof easily generalizes to other languages.

Proposition 4.4.7 *Let $k : \mathbb{N} \to \mathbb{N}$ be an arbitrary function. Then there does not exist a ps/k for* TAUT *which p-simulates every ps/1 for* TAUT *with output advice.*

Proof. Let f be a propositional proof system without advice. We will define an uncountable family of propositional proof systems with one bit of output advice. With each infinite sequence $a = (a_i)_{i \in \mathbb{N}}$ with $a_i \in \{0, 1\}$, we associate the following proof system

$$f_a(\pi) = \begin{cases} f(\pi') & \text{if } \pi = 0\pi' \\ f(\pi') \wedge \top & \text{if } \pi = 1\pi' \text{ and } a_{|f(\pi') \wedge \top|} = 1 \\ f(\pi') \vee \bot & \text{if } \pi = 1\pi' \text{ and } a_{|f(\pi') \vee \bot|} = 0. \end{cases}$$

Because of the first line of its definition, f_a is a complete proof system. Further, f_a uses one bit of output advice, as the length of $f_a(\pi)$ does not depend on the advice bit (because $f(\pi') \wedge \top$ and $f(\pi') \vee \bot$ are of the same length). As all advice bits from the sequence a are coded into the proof system f_a according to lines 2 and 3 of its definition, different sequences a and b also yield different proof systems f_a and f_b. Therefore there exist uncountably many different *ps*/1 for TAUT with output advice.

On the other hand, there are only countably many Turing machines which can compute potential p-simulations between proof systems. Simulating two different proof systems f_a and f_b by one fixed proof system g requires two different simulation functions. Hence the claim follows. □

Proposition 4.4.7 immediately yields that none of the classes of propositional proof systems with advice can have a p-optimal proof system.

Corollary 4.4.8 *Let $k : \mathbb{N} \to \mathbb{N}$ be a function such that $k(n) > 0$ for infinitely many $n \in \mathbb{N}$. Then the class of all ps/k for* TAUT *does not contain a p-optimal proof system. Similarly, the class of all ps/k for* TAUT *with output advice does not contain a p-optimal proof system.*

The previous corollary contains strong negative information on the existence of p-optimal proof systems with advice. In order to still obtain positive results in the spirit of p-optimality, we make the following less restrictive definition.

Definition 4.4.9 *Let L be any language and let $k : \mathbb{N} \to \mathbb{N}$ be any function. Then the class of all ps/k for L has a* p-optimal machine *if there exists a deterministic polynomial-time Turing machine M and a polynomial-time computable advice selector function $\ell : \Sigma^* \to 1^*$ such that for all ps/k f there exists an advice function $h : \mathbb{N} \to \Sigma^*$ and a polynomial-time computable function t such that for all $\pi \in \Sigma^*$*

1. $f(\pi) = M(t(\pi), h(|\ell(t(\pi))|))$ *(the p-simulation),*
2. *for all $n \in \mathbb{N}$, $|h(n)| \leq k(n)$ (the advice bound), and*
3. $M(\pi, h(|\ell(\pi)|)) \in \text{TAUT}$ *(the correctness).*

Let us provide some motivation for this definition. Proof systems with advice essentially consist of three components: the uniform polynomial-time Turing machine, the uniform advice selector function, and the nonuniform advice. As we cannot control the nonuniform component (which causes the absence of p-optimal proof systems by Proposition 4.4.7), it makes sense to ask for a p-optimal system where only the uniform part is fixed, but the nonuniform advice remains variable. This constellation is precisely described by the above notion of a p-optimal machine. In the remaining part of this section we will investigate the question whether p-optimal machines exist for several measures of advice.

In the next definition we single out a large class of natural functions which we will use as advice bounds in Theorem 4.4.11 below.

Definition 4.4.10 *A monotone function $k : \mathbb{N} \to \mathbb{N}$ is* polynomially monotone *if there exists a polynomial p, such that for each $m, n \in \mathbb{N}$, $m \geq p(n)$ implies $k(m) > k(n)$.*

Monotone polylogarithmic functions and monotone polynomials (non-constant) are examples for polynomially monotone functions. If we consider proof systems with a polynomially monotone amount of advice, then we obtain p-optimal machines for each such class. This is the content of the next theorem which we prove by the same technique as was used for Theorem 4.4.6.

Theorem 4.4.11 *Let L be any language and let $k(n)$ be a polynomially monotone function. Then the class of all ps/k for L has a p-optimal machine.*

Proof. Fix L and let k be a function as above. Since k is polynomially monotone we can find a polynomial-time computable function $\ell : \Sigma^* \to 1^*$ such that for each $x \in \Sigma^*$ we have $k(|\ell(x)|) \geq k(|x|)+1$. Moreover, we can choose the function ℓ such that ℓ is injective on lengths, i.e., for all $x, y \in \Sigma^*$, $|\ell(x)| = |\ell(y)|$ implies $|x| = |y|$. Let $\|\cdot\|$ be an encoding of deterministic polynomial-time clocked Turing transducers by natural numbers. Without loss of generality we may assume that every machine M has running time $|x|^{\|M\|}$. Further, we need a polynomial-time computable function $\langle\cdot,\cdot,\cdot\rangle$ mapping triples of $\mathbb{N}$ bijectively to $\mathbb{N}$.

We will construct a polynomial-time Turing machine P which together with the above advice selector function ℓ serves as a p-optimal machine for the class of all ps/k for L. Let Q be a system from the class of all ps/k for L with advice function h_Q. By Proposition 4.1.2 we may assume that Q has input advice. First we will define a polynomial-time computable function t_Q translating Q-proofs into P-proofs and then we will describe how P works. We set $t_Q(\pi) = \pi 1^m$ where m is determined from the equation $m + |\pi| = \langle |\pi|, 1^{\|Q\|}, |\pi|^{\|Q\|}\rangle$.

Now we define the machine P: upon input x we first compute the unique numbers m_1, m_2, m_3 such that $|x| = \langle m_1, m_2, m_3\rangle$. Let $\pi = x_1 \dots x_{m_1}$ be the first m_1 bits of x. Then we determine the machine Q from the encoding $|m_2| = \|Q\|$. By the construction of ℓ, the machine P receives at least one more bit of advice than Q. For the p-simulation of Q, the machine P uses the advice function $h_{P,Q}(|\ell(t_Q(\pi))|) = \mathit{Correct}^\frown h_Q(|\pi|)$, where *Correct* is a bit certifying that under the advice $h_Q(|\pi|)$, the machine Q encoded by $|m_2|$ is indeed a correct proof system for L on proof length $|\pi|$. Because ℓ is injective on lengths, the bit *Correct* can indeed refer to the correctness of Q on proof length $|\pi|$. Therefore, if the first advice bit of P is 1, P simulates Q on input π for m_3 steps, where it passes the last $k(|\pi|)$ advice bits of P to Q. Otherwise, if the first advice bit of P is 0, P outputs $\top$. Except for the first bit, P receives the same advice as Q. Further, the machine P p-simulates every ps/k Q with input advice via the polynomial-time computable function t_Q. By Proposition 4.1.2, P also p-simulates every general ps/k for L. Thus, P and ℓ yield a p-optimal machine. □

In a similar way we get:

Proposition 4.4.12 *For each language L and each constant $k \geq 0$ there exists a machine P using $k+1$ bits of input advice that p-simulates every proof system for L with k bits of input advice.*

Proof. The proof uses the same construction as in the proof of Theorem 4.4.11 where the last k advice bits of the new machine P are the advice bits for the machine Q which we simulate if the first of the $k+1$ advice bits certifies that Q is correct, i.e., it only produces elements from L. □

Regarding the two previous results there remains the question whether for constant k the class of all general ps/k for a language L also has a p-optimal machine with exactly k bits. Going back to the proof of Proposition 4.4.12, we observe that the machine with $k+1$ advice bits, which p-simulates each ps/k, does not really need the full power of these $k+1$ bits, but in fact only needs 2^k+1 different advice strings. For $L = \mathrm{TAUT}$, assuming the existence of a p-optimal proof system without advice, we can manage to reduce the amount of the necessary advice to exactly k bits, thus obtaining a p-optimal machine for the class of all general ps/k.

Theorem 4.4.13 *Assume that there exists a p-optimal proof system for* TAUT*. Then for each constant $k \geq 1$ the class of all ps/k for* TAUT *has a p-optimal machine.*

Proof. By Theorem 3.6.1 we know that the existence of p-optimal propositional proof systems can be characterized as follows:

> *There exists a p-optimal propositional proof system if and only if there exists a recursive enumeration M_i, $i \in \mathbb{N}$, of deterministic polynomial-time clocked Turing machines such that*
>
> 1. *for every $i \in \mathbb{N}$ we have $L(M_i) \subseteq \mathrm{TAUT}$ and*
> 2. *for every polynomial-time decidable subset $L \subseteq \mathrm{TAUT}$ there exists an index i such that $L \subseteq L(M_i)$.*

Assume now that M_i is an enumeration of the easy subsets of TAUT as above. For every proof system Q with k bits of input advice we construct a sequence of propositional formulas

$$Prf^{Q}_{m,n,k}(\pi, \varphi, a) \ ,$$

asserting that the computation of Q at input π of length m leads to the output φ of length n under the k advice bits of a. We also choose a propositional formula $Taut_n(\varphi)$ stating that the formula encoded by φ is a propositional tautology. As Q is a ps/k for TAUT, the formulas

$$Correct^{Q}_{m,n,k} = (\exists a)(\forall \pi, \varphi)\Big(Prf^{Q}_{m,n,k}(\pi, \varphi, a) \to Taut_n(\varphi)\Big)$$

are true quantified Boolean formulas for every $n, m \geq 0$. Since the advice length k is a constant, the quantifier $(\exists a)$ can be replaced by a constant-size disjunction, making it Π^q_1; by prenexing and stripping the universal quantifiers, we obtain a usual Boolean formula. Because the resulting formulas can be constructed in polynomial time from Q, there exists an index $i \in \mathbb{N}$ such that M_i accepts the set of propositional translations of $\{\mathit{Correct}^Q_{m,n,k} \mid m, n \geq 0\}$.

Now we construct a p-optimal machine P with k advice bits as follows: at input x we compute the unique numbers $m_1, \dots, m_5$ such that $|x| = \langle m_1, \dots, m_5 \rangle$. As in the proof of Theorem 4.4.11, we set $\pi = x_1 \dots x_{m_1}$ and $\|Q\| = m_2$. The machine P then simulates $Q(\pi)$ with its own k advice bits for m_3 steps. If the simulation does not terminate, then P outputs $\top$. Otherwise, let φ be the output of this simulation. But before also P can output φ, we have to check the correctness of Q for the respective input and output length. To do this, P simulates the machine M_{m_4} on input $\mathit{Correct}^Q_{m_1,|\varphi|,k}$ for at most m_5 steps. If M_{m_4} accepts, then we output φ, and $\top$ otherwise.

The advice which P receives is the correct advice for Q, in case that M_{m_4} certifies that such advice indeed exists. To show the p-optimality of P, let Q be a ps/k for TAUT with input advice and let M_i be the machine accepting $\{\mathit{Correct}^Q_{m,n,k} \mid m, n \geq 0\}$. Then the system Q is p-simulated by the machine P via the mapping $\pi \mapsto \pi 1^m$ where $m = \langle |\pi|, \|Q\|, p(|\pi|), i, p(\ell) \rangle - |\pi|$, where p is a polynomial bounding the running time of both M_i and Q, and $\ell = \max_{i \leq p(|\pi|)}(|\mathit{Correct}^Q_{|\pi|,i,k}|)$. □

4.5 Applications to Promise Problems

4.5.1 Hard Problems under Advice

Our next result shows that the relation between optimal proof systems and complete sets for promise classes can be transferred to the advice setting. Thus we derive from the optimal proof system with 1 bit of advice the following strong information on complete problems in the presence of advice.

Theorem 4.5.1 *Let C be a promise complexity class and let L be a language such that C is expressible in L by a length-depending promise. Then $\mathsf{C}/1$ contains a problem (or function) using one bit of advice which is many-one hard for C.*

Proof. We choose a polynomial-time computable and invertible tupling function $\langle \cdot \rangle$ on Σ^* which is injective on lengths, i.e., for all $x_1, \dots, x_n, y_1, \dots, y_n \in \Sigma^*$, $|\langle x_1, \dots, x_n \rangle| = |\langle y_1, \dots, y_n \rangle|$ implies $|x_i| = |y_i|$ for $i = 1, \dots, n$.

We now define the problem (or function) A_{C} with one advice bit which will be many-one hard for C. Inputs are of the form

$$\langle x, 0^N, 0^m \rangle$$

where x is the input, 0^N is the unary encoding of a Turing machine N, and 0^m is the time bound for N. On such an input, A_{C} first computes the string

$corr(x, N, 0^m)$. Then A_{C} uses its advice bit to verify whether or not $corr(x, N, 0^m)$ is in L (for this step we could have also used the optimal proof system for L with one bit of advice from Theorem 4.4.1). If $corr(x, N, 0^m) \in L$, then A_{C} simulates N on input x for at most m steps and produces the corresponding output (in case the simulation does not terminate it rejects or outputs some fixed element). As $\langle \cdot, \dots, \cdot \rangle$ is length injective and *corr* is length depending, the element $corr(x, N, 0^m)$ is uniquely determined by $|\langle x, 0^N, 0^m \rangle|$ and therefore the advice bit of A_{C} can in fact refer to $corr(x, N, 0^m)$.

If A is a problem (or function) from C and N is a C-machine for A with polynomial running time p, then A many-one reduces to A_{C} via $x \mapsto \langle x, 0^N, 0^{p(|x|)} \rangle$. Hence A_{C} is many-one hard for C. □

Let us state a concrete application of this general result. There are many reductions for disjoint NP-pairs (cf. [GSS05]). The strongest of these, defined in [KMT03], is the following version of a many-one reduction. Let (A, B) and (C, D) be disjoint NP-pairs. Then (A, B) *strongly many-one reduces* to (C, D) if there exists a polynomial-time computable function f such that $f(A) \subseteq C$, $f(B) \subseteq D$, and $f(\overline{A \cup B}) \subseteq \overline{C \cup D}$. As disjoint NP-pairs are expressible in TAUT by a length-depending promise [Bey07], we obtain:

Corollary 4.5.2 *There exist a disjoint pair (A, B) and a sequence $(a_n)_{n \in \mathbb{N}}$ with the following properties:*

1. *A and B are computable in nondeterministic polynomial time with advice a_n for inputs of length n.*

2. *The set $\{\langle a_n, 0^n \rangle \mid n \in \mathbb{N}\}$ is computable in coNP.*

3. *Every disjoint NP-pair is strongly many-one reducible to (A, B).*

4.5.2 Hard Problems under a Tally NP-Oracle

We now show that for promise classes with a coNP-promise, instead of using advice it also suffices to use a weak oracle to obtain similar results as in the previous section.

Recall that a set $A \subseteq \Sigma^*$ is *sparse* if there exists a polynomial p such that for each $n \in \mathbb{N}$, $|A \cap \Sigma^n| \leq p(n)$. A sparse set A is called *tally* if $A \subseteq \{1^n \mid n \in \mathbb{N}\}$. We denote the set of all sparse and tally sets by Sparse and Tally, respectively. Sparse NP-sets appear to be very weak if used as oracles. For instance, TAUT $\notin \mathsf{NP}^S$ with a sparse NP-oracle S, unless the polynomial hierarchy collapses to its second level [Kad89].

Theorem 4.5.3 *Let C be a promise language (or function) class which is representable in some language from coNP. Then exists a tally NP-oracle A such that C^A contains a language (or function) which is many-one hard for C.*

Proof. As in the proof of Theorem 4.5.1 we choose a polynomial-time computable tupling function $\langle \cdot, \dots, \cdot \rangle$ which is injective on lengths.

The oracle set A now contains all machines which violate the promise condition on some given length, i.e.,

$$A = \{ \langle 1^N, 1^n, 1^t \rangle \mid N, n, t \in \mathbb{N},\ N \text{ is a nondeterministic Turing machine and there exists some } x \in \Sigma^n \text{ such that } \mathit{corr}(x, N, 1^t) \notin L \} \ ,$$

where L is the coNP-set in which the promise of C is expressible.

By definition, the set A is tally. Let us verify that $A \in \mathsf{NP}$. Because of the length injectivity of the tupling function, a number $m \in \mathbb{N}$ already uniquely determines the tuple $\langle 1^N, 1^n, 1^t \rangle$ with $|\langle 1^N, 1^n, 1^t \rangle| = m$. Therefore, on input 1^m we can first determine the entries N, n, t and then verify that N indeed encodes a nondeterministic Turing machine. Next we guess a string $x \in \Sigma^n$ and compute $\mathit{corr}(x, N, 1^t)$. As $L \in \mathsf{coNP}$, we can check $\mathit{corr}(x, N, 1^t) \notin L$ in nondeterministic polynomial time.

The hard set for C will now contain elements $\langle 1^N, x, 1^t \rangle$. On such input, we first query the string $\langle 1^N, 1^{|x|}, 1^t \rangle$ to the oracle A. If the answer is negative, then we simulate M on input x for at most t steps and answer according to the output of this simulation. If the answer is positive or if the simulation does not terminate in t steps, then we reject. It is easy to verify that this yields a hard set (or function) for C. □

As disjoint NP-pairs have a coNP-promise (cf. [Bey07]), we obtain:

Corollary 4.5.4 *There exists a strongly many-one hard disjoint* NP*-pair under a tally* NP*-oracle, i.e., there exists a tally set* $A \in \mathsf{NP}$ *and a disjoint pair* (C_1, C_2) *such that the following holds:*

1. *the components* C_1 *and* C_2 *are computable in* NP^A *with only one query to the oracle* A*, and*

2. *every disjoint* NP*-pair strongly many-one reduces to* (C_1, C_2)*.*

It is known that there is a close connection between disjoint NP-pairs and functions from NPSV, single-valued functions computable in nondeterministic polynomial time (cf. [Sel94, BKM09, GSSZ04] for definitions and background information). Using this correspondence we can formulate Corollary 4.5.4 differently as:

Corollary 4.5.5 *There exists a tally* NP*-set* A *and a function* $f \in \mathsf{NPSV}^A$ *such that every function from* NPSV *is many-one reducible to* f*.*

From Theorem 4.5.4 we also get a sufficient condition for the existence of complete disjoint NP-pairs:

Corollary 4.5.6 *If* $\mathsf{NP} = \mathsf{NP}^{\mathsf{NP}\cap\mathsf{Tally}}$*, then there exist* $\leq_s$*-complete disjoint* NP*-pairs.*

We can rephrase this corollary using the notion of low sets from [Sch83]. Recall that a set $A \in \mathsf{NP}$ is *low* for the nth level $\mathsf{\Sigma_n^p}$ of the polynomial hierarchy if $(\mathsf{\Sigma_n^p})^A \subseteq \mathsf{\Sigma_n^p}$. Intuitively, if a set A is low for $\mathsf{\Sigma_n^p}$, then A is useless as an oracle for the class $\mathsf{\Sigma_n^p}$. All sets $A \in \mathsf{NP}$ which are low for $\mathsf{\Sigma_n^p}$ are collected in the nth level $\mathsf{L_n}$ of the low hierarchy. Using this terminology, we can express Corollary 4.5.6 differently as:

Corollary 4.5.7 *If* $\mathsf{NP} \cap \mathsf{Tally} \subseteq \mathsf{L_1}$*, then there exist* $\leq_s$*-complete disjoint* NP*-pairs.*

Whether or not $\mathsf{NP} \cap \mathsf{Tally} \subseteq \mathsf{L_1}$ is open, but Ko and Schöning [KS85] have shown that $\mathsf{NP} \cap \mathsf{Sparse} \subseteq \mathsf{L_2}$.

Chapter 5

Proof Systems with Advice and Bounded Arithmetic

„Seid doch nicht so frech, Epigramme!“ Warum nicht? Wir sind nur Überschriften; die Welt hat die Kapitel des Buchs.

Johann Wolfgang Goethe, *Venezianische Epigramme*

There is a deep and fruitful connection between propositional proofs and weak subsystems of Peano arithmetic, called *bounded arithmetic* (cf. Krajíček's monograph [Kra95] and the new book of Cook and Nguyen [CN10] on the subject). This connection also holds in the presence of advice and was the main motivation of Cook and Krajíček [CK07] for their investigation. The connection extends to classical problems in computational complexity, and, in fact, between each two of the three fields of propositional proof complexity, bounded arithmetic, and computational complexity there exist strong relations, we refer again to [Kra95, CN10] for a detailed account of these beautiful connections.

In this chapter we will first give an example of the link between computational complexity and bounded arithmetic. This link works in both directions, as we employ complexity-theoretic techniques in bounded arithmetic and obtain in this way a result in bounded arithmetic (a Karp-Lipton collapse result as already briefly mentioned in Section 3.3) which again sheds light on open problems in computational complexity. In our analysis, there is an underlying connection to proof systems with advice, but this is not very visible at first sight. We will comment and explain on this connection at the end of this chapter in Section 5.5.

5.1 A Strong Karp-Lipton Collapse Result in Bounded Arithmetic

The classical Karp-Lipton Theorem states that $\mathsf{NP} \subseteq \mathsf{P/poly}$ implies a collapse of the polynomial hierarchy PH to its second level [KL80]. Subsequently, these collapse consequences have been improved by Köbler and Watanabe [KW98] to $\mathsf{ZPP}^{\mathsf{NP}}$ and by Sengupta and Cai to $\mathsf{S}_2^{\mathsf{p}}$ (cf. [Cai07]). This currently forms the strongest known collapse result of this kind.

Recently, Cook and Krajíček [CK07] have considered the question which collapse consequences can be obtained if the assumption $\mathsf{NP} \subseteq \mathsf{P/poly}$ is provable in some weak arithmetic theory. This assumption seems to be stronger than in the classical Karp-Lipton results, because in addition to the inclusion $\mathsf{NP} \subseteq \mathsf{P/poly}$ we require an easy proof for it. In particular, Cook and Krajíček showed that if $\mathsf{NP} \subseteq \mathsf{P/poly}$ is provable in PV, then PH collapses to the Boolean hierarchy BH, and this collapse is provable in PV. For stronger theories, the collapse consequences become weaker. Namely, if PV is replaced by S_2^1, then $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}[O(\log n)]}$, and for S_2^2 one gets $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}}$ [CK07]. Still all these consequences are presumably stronger than in Sengupta's result above, because $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{S}_2^{\mathsf{p}}$. The situation is summarized in Table 5.1.

Cook and Krajíček [CK07] ask whether under the above assumptions, their collapse consequences for PH are optimal in the sense that also the converse implications hold. Here we give an affirmative answer to this question for the theory PV. Thus PV proves $\mathsf{NP} \subseteq \mathsf{P/poly}$ if and only if PV proves $\mathsf{PH} \subseteq \mathsf{BH}$. To show this result we use the assertion $\mathsf{coNP} \subseteq \mathsf{NP}/O(1)$ as an intermediate assumption. Surprisingly, Cook and Krajíček [CK07] have shown that provability of this assumption in PV is equivalent to the provability of $\mathsf{NP} \subseteq \mathsf{P/poly}$ in PV. While such a trade-off between nondeterminism and advice seems rather unlikely to hold unconditionally, Buhrman, Chang, and Fortnow [BCF03] proved that $\mathsf{coNP} \subseteq \mathsf{NP}/O(1)$ holds if and only if PH collapses to BH. Their proof in [BCF03] refines the hard/easy argument of Kadin [Kad88]. We formalize this technique in PV and thus obtain that $\mathsf{coNP} \subseteq \mathsf{NP}/O(1)$ is provable in PV if and only if PV proves $\mathsf{PH} \subseteq \mathsf{BH}$. Combined with the mentioned results of Cook and Krajíček [CK07], this implies that $PV \vdash \mathsf{PH} \subseteq \mathsf{BH}$ is equivalent to $PV \vdash \mathsf{NP} \subseteq \mathsf{P/poly}$.

Let us remark that this result can also be obtained in a less direct way by combining results of Zambella [Zam96] with recent advances of Jeřábek [Jeř09].

Table 5.1: Stronger assumptions yield stronger Karp-Lipton collapses

PH collapses to	BH	$\subseteq$	$\mathsf{P}^{\mathsf{NP}[O(\log)]}$	$\subseteq$	P^{NP}	$\subseteq$	$\mathsf{S}_2^{\mathsf{p}}$
in the theory	PV		S_2^1		S_2^2		ZFC

The alternative argument proceeds as follows: By a result of Zambella [Zam96] (cf. Theorem 5.3.2), if PV proves $\mathsf{PH} \subseteq \mathsf{BH}$, then Buss' hierarchy of arithmetic theories S_2 collapses to PV. Recently, Jeřábek [Jeř09] proved that the assumption $S_2 = PV$ implies that $PV \vdash \mathsf{coNP} \subseteq \mathsf{NP}/O(1)$. Using the above mentioned result by Cook and Krajíček [CK07] it follows that $PV \vdash \mathsf{NP} \subseteq \mathsf{P}/\mathit{poly}$.

Comparing the two proofs, let us mention that Jeřábek's proof yields a more general result as it also holds for higher levels of the polynomial hierarchy (namely, if T^i_2 proves that the polynomial hierarchy collapses to the Boolean hierarchy over $\Sigma^{\mathsf{p}}_{\mathsf{i+1}}$, then T^i_2 proves $\Sigma^{\mathsf{p}}_{\mathsf{i+1}} \subseteq \Delta^{\mathsf{p}}_{\mathsf{i+1}}/\mathsf{poly}$, cf. [Jeř09]). On the other hand, Jeřábek's result is reached via a new sophisticated and quite elaborate technique called "approximate counting by hashing", whereas our direct proof for the base case $i = 0$ is conceptually much more straightforward (it also uses the mentioned result of Zambella, though).

In addition, using Jeřábek's result one can even obtain the consequence $PV \vdash \mathsf{NP} \subseteq \mathsf{P}/\mathit{poly}$ under the assumption $PV \vdash \mathsf{PH} \subseteq \Sigma^{\mathsf{p}}_2$, which at first sight seems to be weaker than $PV \vdash \mathsf{PH} \subseteq \mathsf{BH}$. This is so, because Zambella [Zam96] actually establishes $PV \vdash \mathsf{PH} \subseteq \Sigma^{\mathsf{p}}_2$ as a sufficient condition for the collapse $S_2 = PV$. Thus we conclude that PV proves $\mathsf{PH} \subseteq \mathsf{BH}$ if and only if PV proves $\mathsf{PH} \subseteq \Sigma^{\mathsf{p}}_2$. This is interesting, as such a result is not known to hold without reference to bounded arithmetic.

In summary, combining our results with previous results from [CK07, Jeř09, Zam96], we obtain that provability in PV of each of the following four things is equivalent to the other three:

1. $\mathsf{NP} \subseteq \mathsf{P}/\mathit{poly}$,

2. $\mathsf{coNP} \subseteq \mathsf{NP}/O(1)$,

3. $\mathsf{PH} = \mathsf{BH}$, and

4. $\mathsf{PH} = \Sigma^p_2$.

5.2 Representing Complexity Classes by Bounded Formulas

The relations between computational complexity and bounded arithmetic are rich and varied, and we refer to [Kra95, CN10] for background information. Here we will use the two-sorted formulation of arithmetic theories [Coo05, CN10]. In this setting we have two sorts: numbers and finite sets of numbers, which are interpreted as strings. Number variables will be denoted by lower case letters $x, y, n, \ldots$ and string variables by upper case letters $X, Y, \ldots$ The two-sorted vocabulary includes the symbols $+, \cdot, \leq, 0, 1$, and the function $|X|$ for the length of strings.

Our central arithmetic theory will be the theory VPV, which is the two-sorted analogue of Cook's PV [Coo75]. In addition to the above symbols, the language of VPV contains names for all polynomial-time computable functions (where the running time is measured in terms of the length of the inputs with numbers coded in unary). The theory VPV is axiomatized by definitions for all these functions as well as by the number induction scheme for open formulas.

Bounded quantifiers for strings are of the form $(\forall X \le t)\varphi$ and $(\exists X \le t)\varphi$, abbreviating $(\forall X)(|X| \le t \to \varphi)$ and $(\exists X)(|X| \le t \wedge \varphi)$, respectively (where t is a number term not containing X). We use similar abbreviations for $=$ instead of $\le$. By counting alternations of quantifiers, a hierarchy Σ_i^B, Π_i^B of bounded formulas is defined. The first level Σ_1^B contains formulas of the type $(\exists X_1 \le t_1) \dots (\exists X_k \le t_k)\varphi$ with only bounded number quantifiers occurring in φ. Similarly, Π_1^B*-formulas* are of the form $(\forall X_1 \le t_1) \dots (\forall X_k \le t_k)\varphi$.

As we want to investigate the provability of various complexity-theoretic assumptions in arithmetic theories, we need to formalize complexity classes within bounded arithmetic. To this end we associate with each complexity class C a class of arithmetic formulas $\mathcal{F}_{\mathsf{C}}$. The formulas $\mathcal{F}_{\mathsf{C}}$ describe C, in the sense that for each $A \subseteq \Sigma^*$ we have $A \in \mathsf{C}$ if and only if A is definable by an $\mathcal{F}_{\mathsf{C}}$-formula $\varphi(X)$ with a free string variable X.

It is well known that Σ_1^B-formulas describe NP-sets in this sense, and this connection extends to the formula classes Σ_i^B and Π_i^B and the respective levels $\mathsf{\Sigma_i^p}$ and $\mathsf{\Pi_i^p}$ of the polynomial hierarchy. Given this connection, we can model the levels BH_k of the Boolean hierarchy by formulas of the type

$$\varphi_1(X) \wedge \neg(\varphi_2(X) \wedge \dots \neg(\varphi_{k-1}(X) \wedge \neg\varphi_k(X)) \dots) \tag{5.1}$$

with Σ_1^B-formulas $\varphi_1, \dots, \varphi_k$.

Another way to speak about complexity classes in arithmetic theories is to consider complete problems for the respective classes. For the satisfiability problem SAT we can build an open formula $Sat(T, X)$, stating that T codes a satisfying assignment for the propositional formula coded by X. In VPV we can prove that $(\exists T \le |X|)\, Sat(T, X)$ is NP-complete, in the sense that every Σ_1^B-formula φ is provably equivalent to $(\exists T \le t(|X|)\, Sat(T, F_\varphi(X))$ for some polynomial-time computable function F_φ and an appropriate number term t.

Using this fact, we can express the classes BH_k in VPV equivalently as:

Lemma 5.2.1 *For every formula φ describing a language from BH_k as in* (5.1) *there is a polynomial-time computable function $F : \Sigma^* \to (\Sigma^*)^k$ such that VPV proves the equivalence of φ and*

$$\begin{aligned} (\exists T_1, T_3, \dots, T_{2\cdot\lfloor k/2\rfloor+1} \le t)(\forall T_2, T_4, \dots, T_{2\cdot\lfloor k/2\rfloor} \le t) \\ (\dots((Sat(T_1, \pi_1(F(X))) \wedge \neg Sat(T_2, \pi_2(F(X)))) \\ \vee Sat(T_3, \pi_3(F(X)))) \wedge \dots \wedge_k \neg^{k+1} Sat(T_k, \pi_k(F(X)))) \end{aligned} \tag{5.2}$$

where $\wedge_k = \wedge$ if k is even and $\vee$ otherwise, $\neg^k = \neg \ldots \neg$ (k-times), and t is a number term bounding $|F(X)|$. We will abbreviate (5.2) *by $BL_k(F(X))$.*

Similarly, we can define the class $\mathsf{P}_{tt}^{\mathsf{NP}[k]}$ by all formulas of the type

$$\begin{aligned}&(\exists T_1 \ldots T_k \leq t)(Sat(T_1, F_1(X)) \wedge \cdots \wedge Sat(T_k, F_k(X)) \wedge \varphi_1(X)) \vee \cdots \vee \\ &(\forall T_1 \ldots T_k \leq t)(\neg Sat(T_1, F_1(X)) \wedge \cdots \wedge \neg Sat(T_k, F_k(X)) \wedge \varphi_{2^k}(X))\end{aligned} \tag{5.3}$$

where $\varphi_1, \ldots, \varphi_{2^k}$ are open formulas, $F_1, \ldots, F_k$ are polynomial-time computable functions, and t is a number term bounding $|F_i(X)|$ for $i = 1, \ldots, k$. In (5.3), every combination of negated and unnegated *Sat*-formulas appears in the disjunction.

With these arithmetic representations we can prove inclusions between complexity classes in arithmetic theories. Let A and B be complexity classes represented by the formula classes $\mathcal{A}$ and $\mathcal{B}$, respectively. Then we use $VPV \vdash \mathcal{A} \subseteq \mathcal{B}$ to abbreviate that for every formula $\varphi_{\mathcal{A}} \in \mathcal{A}$ there exists a formula $\varphi_{\mathcal{B}} \in \mathcal{B}$, such that $VPV \vdash \varphi_{\mathcal{A}}(X) \leftrightarrow \varphi_{\mathcal{B}}(X)$.

In the following, we will use the same notation for complexity classes and their respective representations. Hence we can write statements like $VPV \vdash \mathsf{PH} \subseteq \mathsf{BH}$, with the precise meaning explained above. For example, using Lemma 5.2.1 it is straightforward to verify:

Lemma 5.2.2 *For every number k we have $VPV \vdash \mathsf{BH}_k \subseteq \mathsf{P}_{tt}^{\mathsf{NP}[k]}$.*

Finally, we will consider complexity classes that take advice. Let $\mathcal{A}$ be a class of formulas. For a constant $k \geq 0$, $VPV \vdash \mathcal{A} \subseteq \mathsf{NP}/k$ abbreviates that, for every $\varphi \in \mathcal{A}$ there exist Σ_1^B-formulas $\varphi_1, \ldots, \varphi_{2^k}$, such that

$$VPV \vdash (\forall n) \bigvee_{1 \leq i \leq 2^k} (\forall X)\left(|X| = n \rightarrow (\varphi(X) \leftrightarrow \varphi_i(X))\right) . \tag{5.4}$$

Similarly, using the self-reducibility of SAT, we can formalize the assertion $VPV \vdash \mathsf{NP} \subseteq \mathsf{P}/poly$ as

$$VPV \vdash (\forall n)(\exists C \leq t(n))(\forall X \leq n)(\forall T \leq n)(Sat(T, X) \rightarrow Sat(C(X), X))$$

where t is a number term and $C(X)$ is a term expressing the output of the circuit C on input X (cf. [CK07]).

5.3 The Karp-Lipton Collapse Result in *VPV*

In this section we will prove that, in VPV, the Karp-Lipton collapse $\mathsf{PH} \subseteq \mathsf{BH}$ of Cook and Krajíček [CK07] is optimal in the sense that $VPV \vdash \mathsf{NP} \subseteq \mathsf{P/poly}$ is equivalent to $VPV \vdash \mathsf{PH} \subseteq \mathsf{BH}$. We will use Theorem 3.3.2 of Buhrman, Chang, and Fortnow [BCF03]. For convenience we state Theorem 3.3.2 again:

Theorem 5.3.1 (Buhrman, Chang, Fortnow [BCF03]) *For every constant k we have $\mathsf{coNP} \subseteq \mathsf{NP}/k$ if and only if $\mathsf{PH} \subseteq \mathsf{BH}_{2^k}$.*

While the forward implication of this result is comparatively easy and was shown to hold relative to VPV by Cook and Krajíček [CK07], the backward implication was proven in [BCF03] by a sophisticated hard/easy argument. In the sequel, we will formalize this argument in VPV, thereby answering a question of Cook and Krajíček [CK07], who asked whether $VPV \vdash \mathsf{PH} \subseteq \mathsf{BH}$ already implies $VPV \vdash \mathsf{coNP} \subseteq \mathsf{NP}/O(1)$.

The assumption of a provable collapse of PH to BH also allows us to use stronger tools in our arguments than are known to be available in VPV.

Theorem 5.3.2 (Zambella [Zam96]) *If $PV \vdash \mathsf{PH} \subseteq \mathsf{BH}$, then $PV = S_2$.*

This enables us to use the bounded replacement principle and bounded minimization properties which are presumably not available in VPV. The bounded replacement principle implies that each Σ_i^B class, defined in terms of alternating bounded string quantifiers, is up to provable equivalence closed under bounded number quantifiers (here we only need this for Σ_1^B, so we only require Σ_1^B or equivalently Σ_0^B replacement, cf. [CN10]). The bounded minimization principle states that if a bounded property is ever satisfied, then there exists a lexicographically minimal satisfying element. We will frequently make use of these principles and their consequences later on.

Assuming $VPV \vdash \mathsf{PH} \subseteq \mathsf{BH}$, we claim that there is some constant ℓ such that $VPV \vdash \mathsf{PH} \subseteq \mathsf{BH}_\ell$. This follows, because $\mathsf{PH} \subseteq \mathsf{BH}$ implies $\mathsf{PH} = \mathsf{BH} = \Sigma_2^{\mathsf{p}}$. Therefore every problem in PH can be reduced to a fixed Σ_2^{p}-complete problem. Since this problem is contained in some level BH_ℓ of BH, it can be reduced to an appropriate BH_ℓ-complete problem as well. Thus $\mathsf{PH} \subseteq \mathsf{BH}_\ell$.

Therefore, BH_ℓ is provably closed under complementation in VPV, i.e., there exists a polynomial-time computable function h such that

$$VPV \vdash BL_\ell(X_1, \dots, X_\ell) \leftrightarrow \neg BL_\ell(h(X_1, \dots, X_\ell)) \ . \tag{5.5}$$

Given such a function h, we define the notion of a *hard sequence*. This concept was defined by Chang and Kadin [CK96] as a generalization of the notion of hard strings from [Kad88]. Hard strings were first used to show that $\mathsf{BH} \subseteq \mathsf{D^p}$ implies a collapse of PH [Kad88].

Definition 5.3.3 *Let h be a polynomial-time computable function satisfying (5.5). A sequence $\bar{x} = (x_1, \dots, x_r)$ of strings is a* hard sequence *of order r for length n, if $x_1, \dots, x_r$ are unsatisfiable formulas of length n, and for all $(\ell - r)$-tuples $\bar{u}$ of formulas of length n, the formula $\pi_{\ell-r+i}(h(\bar{u}, \bar{x}))$ is unsatisfiable for each $i = 1, \dots, r$.*

A hard sequence $\bar{x}$ of order r for length n is not extendable *if, for every unsatisfiable formula x of length n the sequence $x^\frown\bar{x}$ is not hard. Finally, a* maximal hard sequence *is a hard sequence of maximal order.*

Maximal hard sequences are obviously not extendable. Note that by definition, the empty sequence is a hard sequence for every length.

To use this definition in VPV, we have to formalize the notions of hard sequences, non-extendable hard sequences, and maximal hard sequences by bounded predicates HS, $NEHS$, and $MaxHS$, respectively.

Definition 5.3.4 *Let* $VPV \vdash BL_\ell(\bar{X}) \leftrightarrow \neg BL_\ell(h(\bar{X}))$ *for some polynomial-time computable function* h *and every* ℓ*-tuple* $\bar{X}$ *of strings.*

For r*-tuples* X *of strings of length* n*, the following predicate* $HS_\ell(X; n, r)$ *expresses that* X *is a hard sequence.* $HS_\ell(X; n, r)$ *is defined as*

$$\begin{aligned}&(\forall i < r)(\forall T \le n)\neg Sat(T, \pi_{i+1}(X)) \wedge \\ &(\forall U \in (\Sigma^n)^{\ell-r})(\forall i < r)(\forall T \le n)(\neg Sat(T, \pi_{\ell-r+i+1}(h(U, X)))) \ .\end{aligned}$$

Similarly, we formalize non-extendable hard sequences by the following predicate $NEHS_\ell(X; n, r)$*, defined as*

$$HS_\ell(X; n, r) \wedge (\forall S = n)\neg HS_\ell(S^\frown X; n, r + 1) \ .$$

Finally, maximal hard sequences are expressed via the following bounded formula $MaxHS_\ell(X; n, r)$

$$HS_\ell(X; n, r) \wedge (\forall S \in (\Sigma^n)^{r+1})\neg HS_\ell(S; n, r + 1) \ .$$

We remark that HS_ℓ is a Π^B_1-predicate, while $NEHS_\ell$ and $MaxHS_\ell$ are Π^B_2-formulas. Maximal hard sequences allow us to define the unsatisfiability of propositional formulas by a Σ^B_1-formula, as stated in the following lemma.

Lemma 5.3.5 *Let* h *be a polynomial-time computable function which for some constant* ℓ *satisfies* (5.5)*. Then* VPV *proves the formula*

$$\begin{aligned}&(\forall n)(\forall X = n)(\forall r < \ell)(\forall H \in (\Sigma^n)^{\ell-r-1})\ (MaxHS_\ell(H; n, \ell - r - 1) \rightarrow \\ &[(\forall T \le n)\neg Sat(T, X) \leftrightarrow (\exists T \le n)(\exists U \in (\Sigma^n)^r)Sat(T, \pi_{r+1}(h(U, X, H)))]) \ .\end{aligned}$$

Proof. We will argue in the theory VPV. Let $H \in (\Sigma^n)^{\ell-r-1}$ be given such that $MaxHS(H; n, \ell - r - 1)$ is fulfilled. Assume $(\forall T \le n)\neg Sat(T, X)$. Then

$$(\forall T \le n)(\forall U \in (\Sigma^n)^r)\neg Sat(T, \pi_{r+1}(h(U, X, H)))$$

implies $\neg NEHS(H; n, \ell - r - 1)$, which in turn implies $\neg MaxHS(H; n, \ell - r - 1)$. Thus it holds that

$$(\exists T \le n)(\exists U \in (\Sigma^n)^r)Sat(T, \pi_{r+1}(h(U, X, H))) \ . \qquad (5.6)$$

On the other hand, assume that (5.6) holds. Then we obtain

$$(\forall T \leq n)\neg Sat(T, X)$$

from $BL_\ell(X_1, \ldots, X_\ell) \leftrightarrow \neg BL_\ell(h(X_1, \ldots, X_\ell))$ in a straightforward calculation showing that by the maximality of H, the formula X cannot be satisfiable if $\pi_{r+1}(h(U, X, H))$ is. □

By the preceding lemma, given maximal hard sequences we can describe Π^B_1-formulas by Σ^B_1-formulas. Most of the proof of the main theorem (Theorem 5.3.8) will go into the construction of such sequences. As we want to use a similar technique as in Theorem 3.3.2, we will now only consider levels of the Boolean hierarchy that are powers of 2. Thus we assume that $\ell = 2^k$ for some k. It will turn out that, assuming $VPV \vdash \mathsf{PH} \subseteq \mathsf{BH}_{2^k}$, we can construct 2^k Σ^B_1-formulas, whose disjunction decides the elements of a maximal hard sequence as in (5.4).

Therefore our aim is to give an NP/k definition of a maximal hard sequence. We will give such a definition for a bitwise encoding of a maximal hard sequence below.

Definition 5.3.6 *Let h be a polynomial-time computable function which for some constant $\ell = 2^k$ satisfies (5.5). We define the predicate $HardSeqBits_{\ell,1}(\langle 1^n, i\rangle)$ by*

$$\begin{aligned}&(\exists H \in (\Sigma^n)^{<\ell})[MaxHS_\ell(H; n, |H|)\wedge \\ &\quad (\forall S \in (\Sigma^n)^{<\ell})(MaxHS_\ell(S; n, |S|) \to \langle H\rangle \leq_{lex} \langle S\rangle) \wedge \pi^*_i(\langle H\rangle) = 1]\ .\end{aligned}$$

Here $\leq_{lex}$ denotes the lexicographic ordering on the strings.

Analogously we define $HardSeqBits_{\ell,0}(\langle 1^n, i\rangle)$ with a 0 substituted for the 1 in the last line of the above formula.

Informally, $HardSeqBits_{\ell,1}(\langle 1^n, i\rangle)$ holds, if the i^{th} bit of the encoding of the lexically smallest maximal hard sequence for length n is 1. $HardSeqBits_{\ell,0}(\langle 1^n, i\rangle)$ holds, if the i^{th} bit of the encoding of the lexically smallest maximal hard sequence for length n is 0. Observe that we need the Π^B_2 minimization principle, as stated after Theorem 5.3.2, to prove the existence of such a minimal H. Let $s_\ell(n)$ be a number term such that sequences with at most ℓ elements from Σ^n are coded by strings of size $\leq s_\ell(n)$ via the tupling function $\langle\cdot\rangle$. We choose this function in such a way that for all bit positions $1 \leq i, j \leq s_\ell(n)$ we get $|\langle 1^n, i\rangle| = |\langle 1^n, j\rangle|$. Thus the length of $\langle 1^n, i\rangle$ only depends on n.

Lemma 5.3.7 *For every k, n, and i, if $VPV \vdash \mathsf{PH} \subseteq \mathsf{BH}_{2^k}$, then*

$$VPV \vdash HardSeqBits_{2^k,1}(\langle 1^n, i\rangle) \in \mathsf{NP}/k\ .$$

The same holds for $HardSeqBits_{2^k,0}$.

Proof. We will only argue for $\mathit{HardSeqBits}_{2^k,1}$ as the proof for $\mathit{HardSeqBits}_{2^k,0}$ follows along the same lines. As $\mathit{HardSeqBits}_{2^k,1}$ is definable by a bounded formula, the assumption $\mathit{VPV} \vdash \mathsf{PH} \subseteq \mathsf{BH}_{2^k}$ together with Lemma 5.2.2 guarantees that the predicate $\mathit{HardSeqBits}_{2^k,1}(\langle 1^n, i\rangle)$ is VPV-provably equivalent to the $\mathsf{P}_{tt}^{\mathsf{NP}[2^k]}$-formula

$$\begin{aligned}
&(\exists T_1 \dots T_{2^k} \leq t_\ell(n))\\
&[\mathit{Sat}(T_1, F_1(\langle 1^n, i\rangle)) \wedge \dots \wedge \mathit{Sat}(T_{2^k}, F_{2^k}(\langle 1^n, i\rangle)) \wedge \varphi_1(\langle 1^n, i\rangle)] \ \vee\\
&\qquad\qquad\qquad\qquad\qquad\qquad\vdots\\
&(\forall T_1 \dots T_{2^k} \leq t_\ell(n))\\
&[\neg\mathit{Sat}(T_1, F_1(\langle 1^n, i\rangle)) \wedge \dots \wedge \neg\mathit{Sat}(T_{2^k}, F_{2^k}(\langle 1^n, i\rangle)) \wedge \varphi_{2^{2^k}}(\langle 1^n, i\rangle)] \ ,
\end{aligned} \tag{5.7}$$

for appropriate polynomial-time computable functions $F_1, \dots, F_{2^k}$, open formulas $\varphi_1, \dots, \varphi_{2^{2^k}}$, and the VPV-number term $t_\ell(n) = |F_1(\langle 1^n, 1\rangle)|$. By padding and because $|\langle 1^n, i\rangle|$ is already determined by n, we can choose the F_j in such a way that the size of the formulas $F_j(\langle 1^n, i\rangle)$ is also determined by n. Thus every formula $F_j(\langle 1^n, i\rangle)$ is of size $t_\ell(n)$.

Our goal is to find Σ_1^B-formulas $\psi_{HSB,1}^0, \dots, \psi_{HSB,1}^{2^k-1}$, such that for every n there is a z, such that for every i, $\mathit{HardSeqBits}_{2^k,1}(\langle 1^n, i\rangle) \leftrightarrow \psi_{HSB,1}^z(\langle 1^n, i\rangle)$. Let Φ be the set of all formulas $F_j(\langle 1^n, i\rangle)$ where $i, n \geq 0$ and $1 \leq j \leq 2^k$. Since we have to evaluate the Sat-formulas only for arguments from Φ, by the proof of Lemma 5.3.5 it suffices to consider only sequences H with elements from Φ. The parameter z in the formulas $\psi_{HSB,1}^z$ will be the order of a maximal hard sequence for length n, if we only allow formulas from Φ in the sequence.

Let now n be given and let H be a tuple of sequences with elements from Φ. Assume further that H contains a hard sequence of order z. Then we can give a Σ_1^B-formula $\psi_{HB,1}^z(\langle 1^n, i\rangle, H)$ that is VPV-equivalent to $\mathit{HardSeqBits}_{2^k,1}(\langle 1^n, i\rangle)$ for every i of suitable length by $\psi_{HB,1}^z(\langle 1^n, i\rangle, H) =_{\mathit{def}}$

$$\begin{aligned}
&(\exists I = 2^k)(\forall i_H \leq |H|)(((|\pi_{i_H}(H)| = z \wedge \mathit{HS}_{2^k}(\pi_{i_H}(H); n, z)) \rightarrow [\\
&\quad(\exists \bar{U} \in (\Sigma^{t_\ell(n)})^{2^k-z-1})(\forall j < 2^k)(\exists T \leq t_\ell(n))[\\
&\quad(\pi_j^*(I) = 1 \rightarrow \mathit{Sat}(T, F_{j+1}(\langle 1^n, i\rangle))) \wedge\\
&\quad(\pi_j^*(I) = 0 \rightarrow \mathit{Sat}(T, \pi_{2^k-z}(h(\bar{U}, F_{j+1}(\langle 1^n, i\rangle), \pi_{i_H}(H))))) \wedge\\
&\quad\varphi_{\ell(I)}(X)]]) \ .
\end{aligned} \tag{5.8}$$

Here, ℓ is a polynomial-time computable function that takes I to the number of the respective line in (5.7). I codes the satisfiability of that line, i.e., $\pi_j^*(I) = 1$ if and only if $F_j(\langle 1^n, i\rangle)$ is satisfiable. This is verified in lines 3 and 4 of (5.8) by a maximal hard sequence. Line 5 then queries the appropriate $\varphi_{\ell(I)}$. The verification is due to Lemma 5.3.5, because we only consider maximal hard sequences in line 4 (by line 1 and the assumption that z is the proper advice).

Observe that HS_{2^k} is Π^B_1 and therefore, using replacement, $\psi^z_{HB,1}$ is equivalent to a Σ^B_1-formula. Abusing notation we will identify $\psi^z_{HB,1}$ with its equivalent Σ^B_1-formula.

Due to (5.8) we will focus on the definition of such a tuple H of sequences, one of which is maximal. First, observe that there are only few possible elements of the sequences to be included in H. Namely, for each n there are just polynomially many propositional formulas coded by the $F_j(\langle 1^n, i\rangle)$. Let $p_F(n)$ be a polynomial bounding this number. Thus, there exist at most $q_F(n) = 2^k \cdot p_F(n)^{2^k}$ sequences of length at most 2^k with elements among the $F_j(\langle 1^n, i\rangle)$. Therefore, even if H contains all such sequences, it will still be polynomial in size. So, we will just give a definition of H that guarantees that H contains every sequence of order less than 2^k. Then H trivially contains every maximal hard sequence.

To this end let $\psi_{all}(H) =_{def}$

$$\begin{aligned}
&(\exists i_\varepsilon \le |H|)(\varepsilon = \pi_{i_\varepsilon}(H)) \ \wedge \\
&(\forall i_H \le |H|)(\forall i_F \le p_F(n)) \\
&\quad (|\pi_{i_H}(H)| < 2^k \to (\exists j \le |H|) F_{p(i_F)}(\langle 1^n, q(i_F)\rangle)^\frown \pi_{i_H}(H) = \pi_j(H)) \ .
\end{aligned}$$

The formula $\psi_{all}(H)$ states in the first line, that H includes the empty sequence ε. The next two lines ensure that, if some sequence s in H does not have maximal length, then H includes every sequence of the type $F_j(\langle 1^n, i\rangle)^\frown s$. Thus, H contains every sequence of length less than 2^k, in particular every maximal hard sequence. Here, p and q are polynomial-time computable functions, such that $F_{p(i)}(\langle 1^n, q(i)\rangle)$ is an enumeration of

$$F_1(\langle 1^n, 0\rangle), \ldots, F_1(\langle 1^n, s_{2^k}(n)\rangle), \ldots, F_{2^k}(\langle 1^n, 0\rangle), \ldots, F_{2^k}(\langle 1^n, s_{2^k}(n)\rangle) \ .$$

By the arguments above, we can define $HardSeqBits_{2^k,1}(\langle 1^n, i\rangle)$ by using $\psi_{all}(H)$ in addition to $\psi^z_{HB,1}(\langle 1^n, i\rangle, H)$. Thus let $\psi^z_{HSB,1}(\langle 1^n, i\rangle) =_{def}$

$$(\exists H \in ((\Sigma^{t_\ell(n)})^{\le 2^k})^{q_F(n)}) \ \psi_{all}(H) \wedge \psi^z_{HB,1}(\langle 1^n, i\rangle, H) \ .$$

Then it holds, that

$$VPV \vdash (\forall n) \bigvee_{0 \le z < 2^k} (\forall i \le s_{2^k}(n))(HardSeqBits_{2^k,1}(\langle 1^n, i\rangle) \leftrightarrow \psi^z_{HSB,1}(\langle 1^n, i\rangle)))$$

which concludes the proof of the lemma. □

The above lemma provides the appropriate tools to prove the converse implication to the Karp-Lipton collapse result of Cook and Krajíček [CK07].

Theorem 5.3.8 *If $VPV \vdash \mathsf{PH} \subseteq \mathsf{BH}_{2^k}$, then $VPV \vdash \mathsf{coNP} \subseteq \mathsf{NP}/k$.*

Proof. Assuming $VPV \vdash \mathsf{PH} \subseteq \mathsf{BH}_{2^k}$, there exists a polynomial-time computable function h, such that for tuples $\bar{X} = (X_1, \dots, X_{2^k})$ we have $VPV \vdash BL_{2^k}(\bar{X}) \leftrightarrow \neg BL_{2^k}(h(\bar{X}))$. Thus, by Lemma 5.3.5, given a maximal hard sequence for length n, we can define $(\forall T \leq n)\neg Sat(T, X)$ by a Σ_1^B-formula. In Lemma 5.3.7 we constructed such a sequence using k bits of advice.

Let $\psi^z_{HSB,1}$ be the Σ_1^B-formula from the proof of Lemma 5.3.7 and let $\psi^z_{HSB,0}$ be its counterpart coding the zeros of the hard sequence.

By Lemma 5.3.7 the theory VPV proves the formulas

$$(\forall n) \bigvee_{0 \leq z < 2^k} (\forall i \leq s_{2^k}(n))\, (\mathit{HardSeqBits}_{2^k,j}(\langle 1^n, i\rangle) \leftrightarrow \psi^z_{HSB,j}(\langle 1^n, i\rangle, Y))$$

for $j \in \{0, 1\}$. As in Lemma 5.3.7, z is the order of a maximal hard sequence for length n. Observe that z, acting as the advice, can be nonuniformly obtained from n.

Provided the right z, there is a Σ_1^B-formula $\mathit{EasyUnSat}_z(X)$ that, for every X of length n, is VPV-equivalent to $(\forall T \leq n)\neg Sat(T, X)$. This is due to Lemma 5.3.5. The formula $\mathit{EasyUnSat}_z(X)$ is defined as

$$\begin{aligned}&(\exists C \leq s_{2^k}(|X|))\ (\forall i < |C|)[\bigwedge_{j \in \{0,1\}} (\pi_i^*(C) = j \to \varphi^z_{HSB,j}(\langle 1^{|X|}, i\rangle, Y)) \wedge\\ &(\exists T \leq |X|)(\exists \bar{U} \in (\Sigma^{|X|})^{2^k-1-|enc(C)|})\ Sat(T, \pi_{2^k-|enc(C)|}(h(\bar{U}, X, enc(C))))]\ .\end{aligned}$$

By the first line of this formula, C is the encoding of some maximal hard sequence. As in Lemma 5.3.5, C is used to define $\neg Sat$ by a Σ_1^B-formula (second line). Thus, we have

$$VPV \vdash (\forall n) \bigvee_{0 \leq z < 2^k} (\forall X = n)[(\forall T \leq n)\neg Sat(T, X) \leftrightarrow \mathit{EasyUnSat}_z(X)]\ .$$

This conclude the proof. □

With this result we can now prove the optimality of the following Karp-Lipton collapse result of Cook and Krajíček [CK07]:

Theorem 5.3.9 (Cook, Krajíček [CK07]) *If VPV proves $\mathsf{NP} \subseteq \mathsf{P/poly}$, then $\mathsf{PH} \subseteq \mathsf{BH}$, and this collapse is provable in VPV.*

To show the converse implication, we use the following surprising trade-off between advice and nondeterminism in VPV:

Theorem 5.3.10 (Cook, Krajíček [CK07]) *$VPV \vdash \mathsf{NP} \subseteq \mathsf{P/poly}$ if and only if $VPV \vdash \mathsf{coNP} \subseteq \mathsf{NP}/O(1)$.*

We remark that the proof of Theorem 5.3.10 uses strong witnessing arguments in form of the Herbrand Theorem and the KPT witnessing theorem [KPT91]. Thus it seems unlikely that a similar result holds without assuming provability of $\mathsf{NP} \subseteq \mathsf{P/poly}$ and $\mathsf{coNP} \subseteq \mathsf{NP}/O(1)$ in some weak arithmetic theory. Theorem 5.3.9 can be obtained as a consequence of Theorem 5.3.10 and a complexity-theoretic proof of $\mathsf{coNP} \subseteq \mathsf{NP}/O(1) \Rightarrow \mathsf{PH} \subseteq \mathsf{BH}$ (cf. [BCF03, CK07]).

Combining Theorems 5.3.8, 5.3.9, and 5.3.10 we can now state the optimality of the Karp-Lipton collapse $\mathsf{PH} \subseteq \mathsf{BH}$ in VPV.

Theorem 5.3.11 *The theory VPV proves $\mathsf{NP} \subseteq \mathsf{P/poly}$ if and only if VPV proves that the polynomial hierarchy collapses to the Boolean hierarchy.*

5.4 Karp-Lipton Results in Stronger Theories

We continue with some discussion about Karp-Lipton collapse results in stronger arithmetic theories. As already mentioned in the beginning, in addition to Theorem 5.3.9, Cook and Krajíček obtain two further results of this kind.

Theorem 5.4.1 (Cook, Krajíček [CK07])

1. *If S^1_2 proves $\mathsf{NP} \subseteq \mathsf{P/poly}$, then $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}[O(\log n)]}$, and this collapse is provable in the theory S^1_2.*
2. *If S^2_2 proves $\mathsf{NP} \subseteq \mathsf{P/poly}$, then $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}}$, and this collapse is provable in the theory S^2_2.*

It remains as an open problem whether also $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}[O(\log n)]}$ and $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}}$ are optimal within S^1_2 and S^2_2, respectively. For S^1_2 this corresponds to the problem whether $\mathsf{coNP} \subseteq \mathsf{NP}/O(\log n)$ is equivalent to $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}[O(\log n)]}$. Buhrman, Chang, and Fortnow [BCF03] conjecture

$$\mathsf{coNP} \subseteq \mathsf{NP}/O(\log n) \Longleftrightarrow \mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}}$$

(cf. also [FK05]). This seems unlikely, as Cook and Krajíček [CK07] noted that $\mathsf{coNP} \subseteq \mathsf{NP}/O(\log n)$ implies $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}[O(\log n)]}$. However, it does not seem possible to extend the technique from [BCF03] to prove the converse implication. Is even $\mathsf{coNP} \subseteq \mathsf{NP/poly} \Longleftrightarrow \mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}}$ true, possibly with the stronger hypothesis that both inclusions are provable in S^2_2? Currently, $\mathsf{coNP} \subseteq \mathsf{NP/poly}$ is only known to imply $\mathsf{PH} \subseteq \mathsf{S}_2^{\mathsf{NP}}$ [CCHO05].

5.5 Classical Proof Systems with Advice

We will now come back to proof systems with advice and how they relate to theories of bounded arithmetic. Assumptions of the form $\mathsf{coNP} \subseteq \mathsf{NP}/O(1)$ play a dominant role in the above Karp-Lipton results. These hypotheses essentially ask whether advice is helpful to decide propositional tautologies. More precisely, the assertion $\mathsf{coNP} \subseteq \mathsf{NP}/O(1)$ states that there exists a polynomially bounded propositional proof system with constant (output) advice (cf. Theorem 4.2.1). Moreover, as in Theorem 5.3.10, Cook and Krajíček considered the provability of the statement $\mathsf{coNP} \subseteq \mathsf{NP}/O(1)$ in theories of bounded arithmetic. This can be rephrased as asking for a proof system for which the theory (VPV in case of Theorem 5.3.10) proves the polynomial boundedness of the proof system. We will see below that this proof system is in fact a variant of the extended Frege system which uses constant advice. These connections to bounded arithmetic were the prior motivation of Cook and Krajíček [CK07] to investigate propositional proof systems taking advice.

We will now describe the extended Frege systems with advice as defined by Cook and Krajíček [CK07]. The systems are inspired by the proof of Theorem 5.3.10. In that proof, disjunctions stemming from the KPT-witnessing theorem [KPT91] were exploited to get a Σ_1^B-description of $(\forall T \leq n)\neg Sat(T, X)$. Before we can define the proof systems, we need to define these disjunctions.

Let $\varphi(X, Y, Z)$ be some open formula in the language of VPV. Let $|Y|$ and $|Z|$ be bounded by $n = |X|$, via $s(n)$ and $t(n)$, respectively. Then $\langle\varphi\rangle_{n,s(n),t(n)}$ denotes the propositional translation of φ for inputs of the respective lengths. We will assume, that the bounds on $|Y|$ and $|Z|$ are implicit in the formula and therefore just write $\langle\varphi\rangle_n$.

The propositional disjunctions to be considered are of the form

$$\langle\varphi\rangle_n(p, C_1(p), q^1) \vee \langle\varphi\rangle_n(p, C_2(p, q^1), q^2) \vee \dots \\ \vee\langle\varphi\rangle_n(p, C_\ell(p, q^1, \dots, q^{\ell-1}), q^\ell),$$

where p is an n-tuple of propositional atoms, the q^i are $t(n)$-tuples of propositional atoms, and the C_i are circuits with $s(n)$ output bits and input bits as indicated. We will refer to such a disjunction as an *(n, ℓ)-disjunction from φ.*

Definition 5.5.1 (Cook, Krajíček [CK07]) *Let $k(n)$ be any function and $\ell(n) =_{\text{def}} 2^{k(n)}$. An extension $EF_{\varphi,F}$ of EF by k advice bits is determined by an open VPV-formula $\varphi(X, Y, Z)$ and a polynomial-time computable function F that, on input 1^n and $1^{\ell(n)}$, outputs an EF-proof of an (n, ℓ)-disjunction from φ.*

For each n the advice specifies the least $i \leq \ell(n)$, such that the disjunction $B_{n,i}$ of the first i disjuncts of the (n, ℓ)-disjunction from φ which is computed by F is a tautology.

A proof of a propositional formula A in $EF_{\varphi,F}$ is a triple $(1^n, 1^{\ell(n)}, W)$, with $n = |A|$ where W is an EF-proof of $B'_{n,i} \to A$. Here, $B'_{n,i}$ is some simple substitution instance of $B_{n,i}$ (i.e. it is only allowed to substitute atoms with constants or atoms).

Cook and Krajíček proved that one of these systems is polynomially bounded if VPV proves $\mathsf{coNP} \subseteq \mathsf{NP}/O(1)$:

Theorem 5.5.2 (Cook and Krajíček [CK07]) *If the theory VPV proves* $\mathsf{coNP} \subseteq \mathsf{NP}/O(1)$*, then there exists an extended Frege system with advice as in Definition 5.5.1 which is polynomially bounded.*

We will now describe a general method of how one can define classical proof systems that use advice. A priori it is not clear how systems like Resolution or Frege can sensibly use advice, but a canonical way to implement advice into them is to enhance these systems by further axioms which can be decided in polynomial time with advice. We first consider again the extended Frege system *EF* as the base system and give a definition of *EF* with advice which generalizes Definition 5.5.1.

Definition 5.5.3 *Let* Φ *be a set of tautologies that can be decided in polynomial time with* $k(n)$ *bits of advice. We define the system* $EF + \Phi/k$ *as follows. An* $EF + \Phi/k$*-proof of a formula* φ *is a pair* $\langle \pi, \psi_0 \rangle$*, where* π *is an EF-proof of an implication* $\psi \to \varphi$ *and* ψ *is a simple substitution instance of* $\psi_0 \in \Phi$ *(simple substitutions only replace some of the variables by constants).*

If π is an $EF + \Phi/k$-proof of a formula φ, then the advice used for the verification of π neither depends on $|\pi|$ nor on $|\varphi|$, but on the length of the substitution instance ψ from Φ, which is used in π. As $|\psi|$ can be easily determined from π, $EF + \Phi/k$ are systems of the type ps/k (in fact, this was the motivation for our general Definition 4.1.1).

If we require that the length of ψ in the implication $\psi \to \varphi$ is determined by the length of the proven formula φ, then the advice only depends on the output and hence we get a ps/k with output advice. This is the case for Cook and Krajíček's extensions of *EF* defined in Definition 5.5.1. Thus, the definition of extended Frege systems with advice of Cook and Krajíček [CK07] is a special case of Definition 5.5.3.

Our next result shows that the optimal propositional proof system from Theorem 4.4.1 is equivalent to an extended Frege system with advice as in Definition 5.5.3.

Theorem 5.5.4

1. *There exists a set* $\Psi \in \mathsf{P}/1$ *such that* $EF + \Psi/1$ *is optimal for the class of all* ps/log *for* TAUT.

2. *In contrast, none of the constant advice extensions of EF from Definition 5.5.1 simulates every propositional* $ps/1$*, unless items 1 to 4 from Theorem 4.4.2 are equivalent.*

Proof. For item 1 we choose the system P using 1 bit of input advice which is optimal for the class of all propositional ps/log according to Theorem 4.4.1. We define the set $\Psi \in \mathsf{P}/1$ as the collection of all formulas

$$RFN^P_{m,n,1} = Prf^P_{m,n,1}(\pi, \varphi, a) \to Taut_n(\varphi)$$

which describe the correctness of P, similarly as in the proof of Theorem 4.4.13. In contrast to the formulas $\mathit{Correct}^P_{m,n,k}$ from the proof of Theorem 4.4.13, the correct advice bit a is already substituted into the formula $\mathit{Prf}^P_{m,n,1}(\pi, \varphi, a)$. Therefore, the set

$$\Psi = \{\mathit{RFN}^P_{m,n,1}(\pi, \varphi, a) \mid m, n \geq 0\}$$

is not necessarily in P, but only in $\mathsf{P}/1$.

To show the optimality of $\mathit{EF} + \Psi/1$ it suffices to prove $P \leq \mathit{EF} + \Psi/1$. For this let π be a P-proof of φ. Substituting the propositional encodings of π and φ into $\mathit{RFN}^P_{|\pi|,|\varphi|,1}$, we obtain the formula

$$\mathit{Prf}^P_{m,n,1}(\pi, \varphi, a) \to \mathit{Taut}_n(\varphi) \ .$$

Now $\mathit{Prf}^P_{m,n,1}(\pi, \varphi, a)$ is a tautological formula, where all relevant variables have been substituted by constants (only auxiliary variables describing the computation of P remain free, but these variables are determined by π). Therefore, we can derive $\mathit{Prf}^P_{m,n,1}(\pi, \varphi, a)$ in a polynomial-size EF-proof, and modus ponens yields $\mathit{Taut}_n(\varphi)$. By induction on the formula φ, we can devise polynomial-size EF-proofs of

$$\mathit{Taut}_n(\varphi) \to \varphi \ .$$

Hence one further application of modus ponens gives the formula φ, and thus we have constructed a polynomial-size $\mathit{EF} + \Psi/1$-proof of φ.

As the extensions of *EF* from Definition 5.5.1 use a constant amount of output advice, the second item follows by Theorem 4.4.2. □

Comparing the definition of *EF* with advice from Definitions 5.5.3 and 5.5.1, we remark that both definitions are parameterized by a set of tautologies Φ, and hence they both lead to a whole class of proof systems rather than *the* extended Frege system with advice. The drawback of our Definition 5.5.3 is, that even in the base case, where no advice is used, we do not get *EF*, but again all extensions $\mathit{EF} + \Phi$ with polynomial-time computable $\Phi \subseteq \mathrm{TAUT}$. It is known that each advice-free propositional proof system is p-simulated by such an extension of *EF* [Kra95]. In contrast, Cook and Krajíček's extended Frege systems with advice lead exactly to *EF*, if no advice is used. On the other hand, even with advice these systems appear to be strictly weaker than the systems from Definition 5.5.3, as indicated by item 2 of Theorem 5.5.4.

Finally, we will outline how other classical proof systems like Resolution can be equipped with advice. Let $\Phi = \{\varphi_n \mid n \geq 0\}$ be a sequence of tautologies in conjunctive normal form. Then φ_n can be written as a set of clauses Δ_n. Assume further that Φ can be decided in polynomial time with $k(n)$ bits of advice. A *Resolution system with advice* $\mathit{Res} + \Phi$ is then defined as follows: Let ψ be a formula in disjunctive normal form and let Γ be the set of clauses for $\neg\psi$. A $\mathit{Res} + \Phi$-proof of ψ is a Resolution refutation of the set $\Delta \cup \Gamma$ where Δ is some simple substitution instance of Δ_n for some n.

Chapter 6

Prover-Delayer Games

> Auf wieviel verschiedene Arten kann man einen Vierlochknopf annähen ? : nun kriegte ich Unterricht [. . .] einfach rundrum (≫Daß also n Quadrat entsteht≪). Oder so, als Andreaskreuz. Oder als 2 Parallele; als Z; als U; als – – ≫Na ? Na ?≪ – – Tja; also nu wußt' ich weißgott keine Möglichkeiten mehr : ≫Iss doch ausgeschlossen !≪ : ≫Haha !≪ : bis sie s herablassend zeigte : ≫Als Gänsefüßchen ! ! – : ⩚ !≪ und nähte ihren triumfierend gleich als solches an : tatsächlich; man lernt nie aus !
>
> Arno Schmidt, *Seelandschaft mit Pocahontas*

Proving lower bounds by games is a very fruitful technique in proof complexity [PB94, Pud99, PI00, AD08]. In particular, the Prover-Delayer game of Pudlák and Impagliazzo [PI00] is one of the canonical tools to study lower bounds in tree-like Resolution [PI00, BSIW04] and tree-like $Res(k)$ [EGM04]. The Prover-Delayer game of Pudlák and Impagliazzo arises from the well-known fact [Kra95] that a tree-like Resolution proof for a formula F can be viewed as a decision tree which solves the search problem of finding a clause of F falsified by a given assignment. In the game, Prover queries a variable and Delayer either gives it a value or leaves the decision to Prover and receives *one* point. The number of Delayer's points at the end of the game is then proportional to the height of the proof tree. It is easy to argue that showing lower bounds by this game only works if (the graph of) every tree-like Resolution refutation contains a balanced sub-tree as a minor, and the height of that sub-tree then gives the size lower bound.

In this chapter we develop a new asymmetric Prover-Delayer game which extends the game of Pudlák and Impagliazzo to make it applicable to obtain lower bounds to tree-like proofs when the proof trees are very unbalanced. In Chapter 7 we will use the new asymmetric game to obtain lower bounds in tree-like Parameterized Resolution, a proof system in the context of parameterized proof complexity recently introduced by Dantchev, Martin, and Szeider [DMS07]. The lower bounds we will obtain there for tree-like Parameterized Resolution are of the form $n^{\Omega(k)}$ (n is the formula size and k the parameter), but the tree-like Parameterized Resolution refutations of the formulas in question only contain balanced sub-trees of height k.

In this chapter we will first explain the original Prover-Delayer game of Pudlák and Impagliazzo [PI00], introduce the refined asymmetric Prover-Delayer game, and apply this new game to (non-parameterized) tree-like Resolution.

One of the best studied principles is the pigeonhole principle. Iwama and Miyazaki [IM99] and independently Dantchev and Riis [DR01] show that the pigeonhole principle requires tree-like Resolution refutations of size roughly $n!$ while its tree-like Resolution proofs only contain balanced sub-trees of height n. Therefore the game of Pudlák and Impagliazzo only yields a $2^{\Omega(n)}$ lower bound which is weaker than the optimal bound $2^{\Omega(n \log n)}$ established by Iwama and Miyazaki. In Section 6.4 we provide a new and easier proof of this lower bound by our asymmetric Prover-Delayer game.

6.1 Tree-Like Resolution and Decision Trees

A Resolution refutation of F can be depicted as a directed graph were vertices are labeled with the clauses of the refutation and a Resolution step

$$\frac{C \quad D}{E}$$

yields edges (C, E) and (D, E). As this graph is acyclic, we also refer to the general Resolution system as dag-like Resolution. If the graph is a tree we call the refutation tree like. When we allow only tree-like refutations we get the *tree-like Resolution* system. In tree-like Resolution, each derived clause can be used at most once as a prerequisite of the Resolution rule.

A tree-like refutation of a set of clauses F can equivalently be described as a *Boolean decision tree*. A Boolean decision tree for F is a binary tree where inner nodes are labeled with variables from F and leafs are labeled with clauses from F. Each path in the tree corresponds to a partial assignment where a variable x gets value 0 or 1 according to whether the path branches left or right at the node labeled with x. The condition on the decision tree is that each path α must lead to a clause which is falsified by the assignment corresponding to α. Therefore, a Boolean decision tree solves the *search problem* for F which, given an assignment α, asks for a clause falsified by α. It is easy to verify that each tree-like Resolution refutation of F yields a Boolean decision tree for F and vice versa, where the size of the Resolution proof equals the number of nodes in the decision tree. In the sequel, we will therefore concentrate on Boolean decision trees when we prove upper or lower bounds to tree-like Resolution.

6.2 The Prover-Delayer Game of Pudlák and Impagliazzo

In this section we introduce the Prover-Delayer game of Pudlák and Impagliazzo [PI00]. Let F be a set of clauses in n variables $x_1, \dots, x_n$. In the game, Prover and Delayer build a (partial) assignment to $x_1, \dots, x_n$. The game is over as soon as the partial assignment falsifies a clause from F. The game proceeds in rounds. In each round, Prover suggests a variable x_i, and Delayer either chooses a value 0 or 1 for x_i or leaves the choice to the Prover. In this last case, if the Prover sets the value, then the Delayer gets exactly 1 point. It is clear that Prover can always win the game on unsatisfiable formulas, but the question is how many points Delayer can earn before the end of this game.

The connection of this game to size of proofs in tree-like Resolution is given by Theorem 6.2.1.

Theorem 6.2.1 (Pudlák, Impagliazzo [PI00]) *Let F be an unsatisfiable formula in CNF. If F has a tree-like Resolution refutation of size at most S, then the Delayer gets at most $\log S$ points in each Prover-Delayer game played on F.*

We omit the proof as Theorem 6.2.1 will be a special case of Theorem 6.3.1 below.

We illustrate the applicability of the game by showing a lower bound to the pigeonhole principle. The *weak pigeonhole principle* PHP^m_n with $m > n$ uses variables $x_{i,j}$ with $i \in [m]$ and $j \in [n]$, indicating that pigeon i goes into hole j. PHP^m_n consists of the clauses

$$\bigvee_{j \in [n]} x_{i,j} \quad \text{for all pigeons } i \in [m]$$

and $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ for all choices of distinct pigeons $i_1, i_2 \in [m]$ and holes $j \in [n]$. We prove that PHP^m_n with $m > n$ is hard for tree-like Resolution. Showing the lower bound by the Prover-Delayer game requires to find a suitable strategy for the Delayer.

Theorem 6.2.2 *Any tree-like Resolution refutation of PHP^m_n for $m > n$ has size $2^{\Omega(n)}$.*

Proof. Let us say that a hole j is occupied if there exists $i \in [m]$ such that $x_{i,j}$ was assigned to 1 in the game. We also assume that Prover never asks the same variable twice.

Then Delayer uses the following strategy: If Prover asks variable $x_{i,j}$, then Delayer answers 0 if hole j is already occupied, otherwise she leaves the decision to Prover. As a first observation, the game never ends by falsifying a conflict clause $\neg x_{i_1,j} \vee \neg x_{i_2,j}$. Therefore the game stops at one of the big clauses $\bigvee_{j \in [n]} x_{i,j}$, i. e., for some $i \in [m]$ all variables $x_{i,j}$ with $j \in [n]$ have been assigned to 0 by either Prover or Delayer.

We claim that Delayer earns at least n points in the game. If variable $x_{i,j}$ was set to 0 by Prover, then Delayer earns 1 point for this. On the other hand, if $x_{i,j}$ was assigned 0 by Delayer, then according to Delayer's strategy, there was some other pigeon $i' \neq i$ sitting in hole j, i. e., $x_{i',j}$ was assigned to 1. This decision was certainly made by Prover, as Delayer never sets a variable to 1 in her strategy. Thus, in total Delayer earns a point for each variable $x_{i,j}$ with $j \in [n]$. By Theorem 6.2.1 the lower bound follows. □

Let us note that the lower bound $2^{\Omega(n)}$ is can be improved. As shown in [IM99, DR01], the optimal lower bound for PHP_n^m in tree-like Resolution is $n!$ which is asymptotically equal to $2^{\Omega(n \log n)}$. This lower bound can be shown via a refinement of the Prover-Delayer game—the asymmetric Prover-Delayer game—which we introduce in the next section.

6.3 The Asymmetric Prover-Delayer Game

We now introduce the asymmetric Prover-Delayer game. Let F be a set of clauses in n variables $x_1, \dots, x_n$. In the asymmetric game, Prover and Delayer again build a (partial) assignment to $x_1, \dots, x_n$. The game is over as soon as the partial assignment falsifies a clause from F. The game proceeds in rounds. In each round, Prover suggests a variable x_i, and Delayer either chooses a value 0 or 1 for x_i or leaves the choice to the Prover. In this last case, if the Prover sets the value, then the Delayer gets some points. The number of points Delayer earns depends on the variable x_i, the assignment α constructed so far in the game, and two functions $c_0(x_i, \alpha)$ and $c_1(x_i, \alpha)$. More precisely, the number of points that Delayer will get is

$$\begin{array}{rl} 0 & \text{if Delayer chooses the value,} \\ \log c_0(x_i, \alpha) & \text{if Prover sets } x_i \text{ to 0, and} \\ \log c_1(x_i, \alpha) & \text{if Prover sets } x_i \text{ to 1.} \end{array}$$

Moreover, the functions $c_0(x, \alpha)$ and $c_1(x, \alpha)$ are chosen in such a way that for each variable x and assignment α

$$\frac{1}{c_0(x, \alpha)} + \frac{1}{c_1(x, \alpha)} = 1 \tag{6.1}$$

holds. Let us call this game the (c_0, c_1)-game on F.

The connection of this game to size of proofs in tree-like Resolution is given by Theorem 6.3.1.

Theorem 6.3.1 *Let F be unsatisfiable formula in CNF and let c_0 and c_1 be two functions satisfying* (6.1) *for all partial assignments α to the variables of F. If F*

has a tree-like Resolution refutation of size at most S, then the Delayer gets at most $\log S$ *points in each* (c_0, c_1)*-game played on* F.

Proof. Let F be an unsatisfiable CNF in variables $x_1, \dots, x_n$ and let Π be a tree-like Resolution refutation of F. Assume now that Prover and Delayer play a game on F where they successively construct an assignment α. Let α_i be the partial assignment constructed after i rounds of the game, i. e., α_i assigns i variables a value 0 or 1. By p_i we denote the number of points that Delayer has earned after i rounds, and by Π_{α_i} we denote the sub-tree of the decision tree of Π which has as its root the node reached in Π along the path specified by α_i.

We use induction on the number of rounds in the game to prove the following claim:

$$|\Pi_{\alpha_i}| \le \frac{|\Pi|}{2^{p_i}} \text{ for any round } i.$$

To see that the theorem follows from this claim, let α be an assignment constructed during the game yielding p_α points to the Delayer. As a contradiction has been reached in the game, the size of Π_α is 1, and therefore by the inductive claim

$$1 \le \frac{|\Pi|}{2^{p_\alpha}} \ ,$$

yielding $p_\alpha \le \log |\Pi|$ as desired.

In the beginning of the game, Π_{α_0} is the full tree and the Delayer has 0 points. Therefore the claim holds.

For the inductive step, assume that the claim holds after i rounds and Prover asks for a value of the variable x in round $i+1$. If the Delayer chooses the value, then $p_{i+1} = p_i$ and hence

$$|\Pi_{\alpha_{i+1}}| \le |\Pi_{\alpha_i}| \le \frac{|\Pi|}{2^{p_i}} = \frac{|\Pi|}{2^{p_{i+1}}} \ .$$

If the Delayer defers the choice to the Prover, then the Prover uses the following strategy to set the value of x. Let $\alpha_i^{x=0}$ be the assignment extending α_i by setting x to 0, and let $\alpha_i^{x=1}$ be the assignment extending α_i by setting x to 1. Now, Prover sets $x = 0$ if $|\Pi_{\alpha_i^{x=0}}| \le \frac{1}{c_0(x,\alpha_i)}|\Pi_{\alpha_i}|$, otherwise he sets $x = 1$. Because $\frac{1}{c_0(x,\alpha_i)} + \frac{1}{c_1(x,\alpha_i)} = 1$, we know that if Prover sets $x = 1$, then $|\Pi_{\alpha_i^{x=1}}| \le \frac{1}{c_1(x,\alpha_i)}|\Pi_{\alpha_i}|$. Thus, if Prover's choice is $x = j$ with $j \in \{0, 1\}$, then we get

$$|\Pi_{\alpha_{i+1}}| = |\Pi_{\alpha_i^{x=j}}| \le \frac{|\Pi_{\alpha_i}|}{c_j(x,\alpha_i)} \le \frac{|\Pi|}{c_j(x,\alpha_i)2^{p_i}} = \frac{|\Pi|}{2^{p_i+\log c_j(x,\alpha_i)}} = \frac{|\Pi|}{2^{p_{i+1}}} \ .$$

This completes the proof of the induction. □

Apparently, we get the game of Pudlák and Impagliazzo [PI00] as a special case of the our asymmetric game by setting $c_0(x, \alpha) = c_1(x, \alpha) = 2$ for all variables x and partial assignments α.

6.4 Tree-like Resolution Lower Bounds for the Pigeonhole Principle

Proving lower bounds in our new game, i. e., devising good Delayer strategies, entails first of all to finding suitable functions c_0 and c_1. Functions c_0 and c_1 can be interpreted in terms of *information content* of tree-like Resolution refutations. The points which Delayer scores in one round should be proportional to the fraction of the current refutation that Prover can avoid to check by deciding a value for the variable. This is easily understandable in the case of the original game: the only good strategy for Prover to set a variable is choosing the value that allows him to proceed the game in the smallest of the two sub-trees of the current refutation which is in fact of size smaller than 1/2 of the current refutation size.

We now use the asymmetric Prover-Delayer game to prove a better lower bound for the pigeonhole principle than the exponential bound obtained via the Pudlák-Impagliazzo game in Theorem 6.2.2. This lower bound is of the form $2^{\Omega(n \log n)}$ and was established by Iwama and Miyazaki [IM99] and independently Dantchev and Riis [DR01]. We give a new proof based on our asymmetric Prover-Delayer game which is conceptually simpler than the previous arguments.

Theorem 6.4.1 *Any tree-like Resolution refutation of PHP^m_n has size $2^{\Omega(n \log n)}$.*

Proof. Let α be a partial assignment to the variables $\{ x_{i,j} \mid i \in [m], j \in [n] \}$. Let

$$p_i(\alpha) = |\{ j \in [n] \mid \alpha(x_{i,j}) = 0 \text{ and } \alpha(x_{i',j}) \neq 1 \text{ for all } i' \in [m] \}| \enspace .$$

Intuitively, $p_i(\alpha)$ corresponds to the number of holes which are still free but are explicitly excluded for pigeon i by α (we do not count the holes which are excluded because some other pigeon is sitting there). We define

$$c_0(x_{i,j}, \alpha) = \frac{\frac{n}{2} + 1 - p_i(\alpha)}{\frac{n}{2} - p_i(\alpha)} \quad \text{and} \quad c_1(x_{i,j}, \alpha) = \tfrac{n}{2} + 1 - p_i(\alpha) \enspace .$$

For simplicity we assume that n is divisible by 2. During the game it will never be the case that Prover gets the choice when $p_i(\alpha) \geq \frac{n}{2}$. Therefore the functions c_0 and c_1 are always greater than zero when the Delayer gets points, thus the score function is always well defined. Furthermore notice that this definition satisfies (6.1).

We now describe Delayer's strategy in a (c_0, c_1)-game played on PHP^m_n. If Prover asks for a value of $x_{i,j}$, then Delayer decides as follows:

set $\alpha(x_{i,j}) = 0$ if there exists $i' \in [m] \setminus \{i\}$ such that $\alpha(x_{i',j}) = 1$ or
if there exists $j' \in [n] \setminus \{j\}$ such that $\alpha(x_{i,j'}) = 1$;
set $\alpha(x_{i,j}) = 1$ if $p_i(\alpha) \geq \frac{n}{2}$ and there is no $i' \in [m]$ with $\alpha(x_{i',j}) = 1$, and
let Prover decide otherwise.

Intuitively, Delayer leaves the choice to Prover as long as pigeon i does not already sit in a hole, hole j is still free, and there are at most $\frac{n}{2}$ excluded free holes for pigeon i.

Let us pause to give an intuitive explanation of why we choose the functions c_0 and c_1 and thus the points for Delayer as above. As a first observation, Delayer always earns more when Prover is setting a variable $x_{i,j}$ to 1 instead of setting it to 0. This is intuitively correct as the amount of freedom for Delayer to continue the game is by far more diminished by sending pigeon i to some hole j than by just excluding that hole j for pigeon i. In fact, our choice of scores can be completely explained by the following information-theoretic interpretation: When Prover sends a pigeon to a hole, Delayer should always get about $\log n$ points on that pigeon. For our Delayer strategy, sending pigeon i to a hole either means that Prover excluded $\frac{n}{2}$ holes for pigeon i or was setting pigeon i directly to a hole. When we play the game, in each round Delayer should get some number of points proportional to the progress Prover made towards fixing pigeon i to a hole. For instance, if Prover fixes i to a hole in the very beginning by answering 1 to $x_{i,j}$, Delayer should get the $\log n$ points immediately. On the other extreme, if Prover has already excluded $\frac{n}{2} - 1$ holes for pigeon i, then it does not matter whether Prover sets $x_{i,j}$ to 0 or 1 because after both answers pigeon i will be forced to a hole. Consequently, in the latter case, Delayer gets just 1 point regardless of whether Prover answers 0 or 1. This is exactly what our score function provides.

If Delayer uses the above strategy, then the small clauses $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ from PHP^m_n will not be violated in the game. Therefore, a contradiction will always be reached on one of the big clauses $\bigvee_{j \in [n]} x_{i,j}$. Let us assume now that the game ends by violating $\bigvee_{j \in [n]} x_{i,j}$, i. e., for pigeon i all variables $x_{i,j}$ with $j \in [n]$ have been set to 0. As soon as the number $p_i(\alpha)$ of excluded free holes for pigeon i reaches the threshold $\frac{n}{2}$, Delayer will not leave the choice to Prover. Instead, Delayer will try to place pigeon i into some hole. If Delayer still answers 0 to $x_{i,j}$ even after $p_i(\alpha) > \frac{n}{2}$, it must be the case that some other pigeon already sits in hole j, i. e., for some $i' \neq i$, $\alpha(x_{i',j}) = 1$. Therefore, at the end of the game at least $\frac{n}{2}$ variables have been set to 1. W. l. o. g. we assume that these are the variables x_{i,j_i} for $i = 1, \ldots, \frac{n}{2}$.

Let us check how many points Delayer earns in this game. We calculate the points separately for each pigeon $i = 1, \ldots, \frac{n}{2}$ and distinguish two cases: whether x_{i,j_i} was set to 1 by Delayer or Prover. Let us first assume that Delayer sets the variable x_{i,j_i} to 1. Then pigeon i was not assigned to a hole yet and, moreover, there must be $\frac{n}{2}$ unoccupied holes which are already excluded for pigeon i by α, i. e., there is some $J \subseteq [n]$ with $|J| = \frac{n}{2}$, $\alpha(x_{i',j'}) \neq 1$ for $i' \in [m]$, $j' \in J$, and $\alpha(x_{i,j'}) = 0$ for all $j' \in J$. All of these 0's have been assigned by Prover, as Delayer has only assigned a 0 to $x_{i,j'}$ when some other pigeon was already sitting in hole j', and this is not the case for the holes from J (at the moment when Delayer assigns the 1 to x_{i,j_i}). Thus, before Delayer sets $\alpha(x_{i,j_i}) = 1$, she has already earned points for all $\frac{n}{2}$ variables $x_{i,j'}$, $j' \in J$, yielding

$$\sum_{p=0}^{\frac{n}{2}-1} \log \frac{\frac{n}{2}+1-p}{\frac{n}{2}-p} \;=\; \log \prod_{p=0}^{\frac{n}{2}-1} \frac{\frac{n}{2}+1-p}{\frac{n}{2}-p} \;=\; \log\left(\frac{n}{2}+1\right)$$

points for the Delayer. Let us note that because Delayer never allows a pigeon to go into more than one hole, she will really get the number of points calculated above for *every* of the variables which she set to 1.

If, conversely, Prover sets variable x_{i,j_i} to 1, then Delayer gets $\log(\frac{n}{2}+1-p_i(\alpha))$ points for this, but she also received points for the $p_i(\alpha)$ variables set to 0 before by Prover. Thus, in this case Delayer earns on pigeon i

$$\begin{aligned} & \log(\frac{n}{2}+1-p_i(\alpha)) + \sum_{p=0}^{p_i(\alpha)-1} \log \frac{\frac{n}{2}+1-p}{\frac{n}{2}-p} \\ = \;& \log(\frac{n}{2}+1-p_i(\alpha)) + \log \frac{\frac{n}{2}+1}{\frac{n}{2}-p_i(\alpha)+1} \\ = \;& \log\left(\frac{n}{2}+1\right) \end{aligned}$$

points. In total, Delayer gets at least

$$\frac{n}{2}\log\left(\frac{n}{2}+1\right)$$

points in the game. Applying Theorem 6.3.1, we obtain $2^{\frac{n}{2}\log\left(\frac{n}{2}+1\right)}$ as a lower bound to the size of each tree-like Resolution refutation of PHP^m_n. □

By inspection of the above Delayer strategy it becomes clear that the lower bound from Theorem 6.4.1 also holds for the *functional pigeonhole principle* where in addition to the clauses from PHP^m_n we also include $\neg x_{i,j_1} \vee \neg x_{i,j_2}$ for all pigeons $i \in [m]$ and distinct holes $j_1, j_2 \in [n]$.

We remark that the choice of the score functions c_0 and c_1 in the proof of Theorem 6.4.1 is by no means unique. It is even possible to obtain the same asymptotic lower bound $2^{\Omega(n \log n)}$ by choosing simpler score functions c_0, c_1 which do not depend on the game played so far, i. e., c_0 and c_1 just depend on n, but are independent of the assignment α and the queried variable x. Namely, setting

$$c_1 = \frac{n}{\log n} \quad \text{and} \quad c_0 = \frac{c_1}{c_1-1} = 1 + \frac{1}{c_1-1} = \Omega(e^{\frac{1}{c_1-1}}) = 2^{\Omega(\frac{\log n}{n})}$$

we obtain score functions which satisfy (6.1) and lead to the following modified analysis in the proof of Theorem 6.4.1: if the Prover sets $x_{i,j}$ to 1, then Delayer earns at least $\log c_1 = \Omega(\log n)$ points. Otherwise, she still earns at least $\frac{n}{2}\log c_0 =$

$\Omega(\log n)$ points on pigeon i. Thus, in total Delayer earns $\frac{n}{2} \cdot \Omega(\log n)$ points during the game, yielding the lower bound.

Our first proof of Theorem 6.4.1 has the advantage that it yields more precise and better bounds, namely exactly $2^{\frac{n}{2}\log\left(\frac{n}{2}+1\right)}$ which is the same lower bound obtained by Dantchev and Riis [DR01]. There might also be scenarios where the adaptive definition of points according to our above information-theoretic interpretation indeed yields better asymptotic bounds.

Chapter 7

Parameterized Proof Complexity

> Kehr' ich in mich selbst zurück, wie man doch so gern tut bei jeder Gelegenheit, so entdecke ich ein Gefühl, das mich unendlich freut, ja, das ich sogar auszusprechen wage. Wer sich mit Ernst hier umsieht und Augen hat zu sehen, muß solid werden, er muß einen Begriff von Solidität fassen, der ihm nie so lebendig ward.[1]
>
> Johann Wolfgang Goethe, *Italienische Reise*

Parameterized complexity is a very successful branch of computational complexity where problems are analyzed in a finer way than in the classical approach [DF99, FG06, Nie06]. Instead of expressing the complexity of a problem as a function only of the input size, one parameter is part of the input instance, and one investigates the effect of the parameter on the complexity. In this setting, many classically intractable problems have efficient solutions for small choices of the parameter, even if the total size of the input is large.

Parameterized proof complexity has been recently introduced by Dantchev, Martin, and Szeider [DMS07]. After considering the notions of propositional *parameterized tautologies* and *fpt-bounded* proof systems, they laid the foundations to study complexity of proofs in a parameterized setting. In particular, they considered a parameterized version of Resolution which is the best studied and most important propositional proof system. In contrast to classical Resolution, *Parameterized Resolution* appears to be a relatively powerful proof system as a number of classically hard principles admit fpt-bounded proofs even in tree-like Parameterized Resolution as we will show in this chapter. In particular, we transfer the concept of a kernel from parameterized complexity to proof complexity and construct kernelizations for many classically hard principles as the class of

[1] Goethe in Rome on November 10, 1786. Chapters 6 and 7 of this dissertation were written during a research stay at Sapienza University Rome.

all CNF's of bounded width. For hardness results we use the asymmetric Prover-Delayer game from the previous chapter to model and study the complexity of proofs in tree-like Parameterized Resolution. Moreover, we obtain the first lower bound to the general dag-like Parameterized Resolution system for the pigeonhole principle and study a variant of the DPLL algorithm in the parameterized setting.

We start this chapter by providing some background information from parameterized complexity.

7.1 Fixed-Parameter Tractability

A *parameterized language* is a language $L \subseteq \Sigma^* \times \mathbb{N}$. For an instance (x, k), we call k the parameter of (x, k). In parameterized complexity, the classical notion of efficiency is replaced by the more liberal concept of fixed-parameter tractability.

Definition 7.1.1 *A parameterized language L is* fixed-parameter tractable *if L has a deterministic decision algorithm running in time $f(k) \cdot |x|^{O(1)}$ for some computable function f. The class of all fixed-parameter tractable languages is denoted by* FPT.

Many classically hard problems like vertex cover are fixed-parameter tractable for natural choices of the parameter.

Parameterized languages are compared via fpt-reductions defined as follows:

Definition 7.1.2 *A parameterized language $L_1 \subseteq \Sigma_1^* \times \mathbb{N}$* fpt-reduces *to a parameterized language $L_2 \subseteq \Sigma_2^* \times \mathbb{N}$ if there is a mapping $R : \Sigma_1^* \times \mathbb{N} \to \Sigma_2^* \times \mathbb{N}$ such that*

1. *For all $(x, k) \in \Sigma_1^* \times \mathbb{N}$, $(x, k) \in L_1$ if and only if $R(x, k) \in L_2$.*

2. *R is computable by a fixed-parameter algorithm, i. e., there exists a computable function f such that R is computable in time $f(k) \cdot |x|^{O(1)}$.*

3. *There is a computable function g such that whenever $R(x, k) = (x', k')$, then $k' \leq g(k)$.*

Besides FPT there is a wealth of complexity classes containing problems which are not believed to be fixed-parameter tractable. The most prominent classes lie in the *weft hierarchy* forming a chain

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[\mathsf{P}] \subseteq \text{para-NP} \ .$$

The important classes for us will be $\mathsf{W}[1], \mathsf{W}[2]$, its complements $\mathsf{coW}[1], \mathsf{coW}[2]$, and para-NP. So we will just define those and otherwise refer to the monograph [FG06].

The classes of the weft hierarchy are usually defined as the closure of a canonical problem under fpt-reductions. These canonical problems are mostly weighted versions of the satisfiability problem. The *weight of an assignment* α, denoted $w(\alpha)$, is the number of variables that α assigns to 1.

The canonical $\mathsf{W}[1]$-complete problem is the following:

Problem:	WEIGHTED d-CNF SAT (complete for $\mathsf{W}[1]$)
Input:	F, a CNF where all clauses have at most d literals.
Parameter:	k, a non-negative integer.
Question:	Does there exist a satisfying assignment for F with weight *equal* to k?

The unbounded width version of this problem is the canonical $\mathsf{W}[2]$-complete problem, defined as:

Problem:	WEIGHTED CNF SAT (complete for $\mathsf{W}[2]$)
Input:	F, a CNF.
Parameter:	k, a non-negative integer.
Question:	Does there exist a satisfying assignment for F with weight *equal* to k?

Instead of asking for a satisfying assignment α with $w(\alpha) = k$ we can also ask for α with $w(\alpha) \leq k$ and still get a $\mathsf{W}[2]$-complete problem. This observation was made by Dantchev et al. [DMS07], but they did not give a full proof. We present a full proof for completeness.

Proposition 7.1.3 (Dantchev, Martin, Szeider [DMS07]) *Deciding if an arbitrary CNF F has a satisfying assignment of weight at most k is $\mathsf{W}[2]$-complete.*

Proof. We refer to this problem as the "relaxed" version of WEIGHTED CNF SAT, which is in contrast referred as the "exact" version. Consider $k+1$ new variables $y_1, \ldots, y_{k+1}$. The formula $F \wedge (y_1 \vee y_2 \vee \cdots \vee y_{k+1})$ has a satisfying assignment of weight $k+1$ if and only if F has a satisfying assignment of weight at most k. Furthermore, translating back such assignment to an assignment for F is a simple projection. This proves that the relaxed version is reducible to the exact one, and thus the former is in $\mathsf{W}[2]$.

To prove completeness consider the following reduction from the exact version to the relaxed one. Let (F, k) be the input, where F is a CNF in variables $x_1, \ldots, x_n$. Consider the following CNF ψ which in addition to $x_1, \ldots, x_n$ uses new variables $y_{i,j}$ for $i \in [n]$ and $j \in [k]$:

$$\begin{array}{ll}
\bigvee_i y_{i,j} & \text{for any } j \in [k] \\
\neg y_{i,j} \vee \neg y_{i',j} & \text{for any } i \neq i' \in [n] \text{ and } j \in [k] \\
\neg y_{i,j} \vee \neg y_{i,j'} & \text{for any } i \in [n] \text{ and } j \neq j' \in [k] \\
y_{i,1} \vee\ y_{i,2} \vee\ \ldots \vee\ y_{i,k} \vee\ \neg x_i & \text{for any } i \in [n] \\
\neg y_{i,j} \vee\ x_i & \text{for any } i \in [n] \text{ and } j \in [k]
\end{array}$$

Formula ψ is satisfiable if and only if there is a set of k indices in $[n]$ matched with $[k]$. Variable x_i is true if and only if i is in such set. Thus any satisfying assignment for ψ has weight $2k$. The reduction from the exact version to the relaxed version of Weighted CNF Sat is given by $(F, k) \mapsto (F \wedge \psi, 2k)$. This proves that the relaxed version is W[2]-hard. □

In Theorem 7.6.2 below we show that for CNF's of constant width the problem of finding an assignment with weight *at most k* is in FPT. Thus W[1]-completeness really depends on the equality requirement (this observation was also made in [DF99, CF08]).

Observe that using unbounded clauses is crucial in the previous proof, and that this reduction does not work with bounded width CNF's. In general, any reduction from Weighted d-CNF Sat to its relaxed version would imply $\mathsf{W}[1] \subseteq \mathsf{FPT}$ and $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{o(n)})$ (for the latter implication see [DF99, Corollary 17.7]).

Using the relaxed version of Weighted CNF Sat, we obtain a complete problem for coW[2] as in the classical duality between tautologies and satisfiability:

Definition 7.1.4 (Dantchev, Martin, Szeider [DMS07]) *A* parameterized contradiction *is a pair (F, k) consisting of a propositional formula F and $k \in \mathbb{N}$ such that F has no satisfying assignment of weight $\leq k$. We denote the set of all parameterized contradictions by* PCon.

7.2 Parameterized Proof Systems

We start with the general definition of a parameterized proof system of Dantchev, Martin, and Szeider [DMS07].

Definition 7.2.1 (Dantchev, Martin, Szeider [DMS07]) *A* parameterized proof system *for a parameterized language $L \subseteq \Sigma^* \times \mathbb{N}$ is a function $P : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that $rng(P) = L$ and $P(x, k)$ can be computed in time $O(f(k)|x|^{O(1)})$ with some computable function f.*

In this definition, there are two parameters: one stemming from the parameterized language, but there is also a parameter in the proof. Verification of proofs then proceeds in fpt-time in the proof parameter. We contrast this with the classical concept of Cook and Reckhow where proofs are verified in polynomial time:

Definition 7.2.2 (Cook, Reckhow [CR79]) *A* proof system *for a language $L \subseteq \Sigma^*$ is a polynomial-time computable function $P : \Sigma^* \to \Sigma^*$ with $rng(P) = L$.*

This framework can also be applied to parameterized languages $L \subseteq \Sigma^* \times \mathbb{N}$. Thus, in contrast to the parameterized proof systems from Definition 7.2.1, we say that a *proof system for the parameterized language* L is just a polynomial-time computable function $P : \Sigma^* \to \Sigma^* \times \mathbb{N}$ with $rng(P) = L$. The difference to Definition 7.2.1 is that proofs are now verified in polynomial time. We will argue in Theorem 7.2.6 that this weaker and simpler notion is in fact equivalent to the notion of parameterized proof systems from Definition 7.2.1 when considering lengths of proofs.

For lengths of proofs it is appropriate to adjust the notion of short proofs to the parameterized setting. This was formalized by Dantchev, Martin, and Szeider as follows:

Definition 7.2.3 (Dantchev, Martin, Szeider [DMS07]) *A parameterized proof system P for a parameterized language L is* fpt-bounded *if there exist computable functions f and g such that every $(x, k) \in L$ has a P-proof (y, k') with $|y| \leq f(k)|x|^{O(1)}$ and $k' \leq g(k)$.*

Again, if we use polynomial-time computable proof systems for parameterized languages, this definition simplifies a bit as follows:

Definition 7.2.4 *A proof system P for a parameterized language L is* fpt-bounded *if there exists a computable function f such that every $(x, k) \in L$ has a P-proof of size at most $f(k)|x|^{O(1)}$.*

The main motivation behind the work of [DMS07] was that of generalizing the classical approach of Cook and Reckhow to the parameterized case and working towards a separation of parameterized complexity classes as FPT and W[P] by techniques developed in proof complexity. In fact, we will obtain an analogous result to the well-known Cook-Reckhow theorem.

For this we want to determine which parameterized languages admit fpt-bounded proof systems. Recall that by the theorem of Cook and Reckhow [CR79], the class of all languages with polynomially bounded proof systems coincides with NP. To obtain a similar result in the parameterized world, we use the following parameterized version of NP.

Definition 7.2.5 (Flum, Grohe [FG03]) *The class* para-NP *contains all parameterized languages which can be decided by a nondeterministic Turing machine in time $f(k)|x|^{O(1)}$ for some computable function f.*

The following result is a direct analogue of the classical theorem of Cook and Reckhow [CR79] for the parameterized setting. Moreover, it shows the equivalence of our two notions for proof systems for parameterized languages with respect to the existence of fpt-bounded systems.

Theorem 7.2.6 *Let $L \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized language. Then the following statements are equivalent:*

1. *There exists an fpt-bounded proof system for L.*
2. *There exists an fpt-bounded parameterized proof system for L.*
3. *$L \in \mathsf{para\text{-}NP}$.*

Proof. For the implication 1 ⇒ 2, let P be an fpt-bounded proof system for L. Then the system P' defined by $P'(y,k) = P(y)$ is an fpt-bounded parameterized proof system for L.

For the implication 2 ⇒ 3, let P be an fpt-bounded parameterized proof system for L such that every $(x,k) \in L$ has a P-proof (y,k') with $|y| \le f(k)p(|x|)$ and $k' \le g(k)$ for some computable functions f, g and some polynomial p. Let M be a Turing machine computing P in time $h(k)q(n)$ with computable h and a polynomial q. Then $L \in \mathsf{para\text{-}NP}$ by the following algorithm: on input (x,k) we guess a proof (y,k') with $|y| \le f(k)p(|x|)$ and $k' \le g(k)$. Then we verify that $P(y,k') = (x,k)$ in time $h(k')q(|y|)$ which by the choice of (y,k) yields an fpt running time. If the test is true, then we accept the input (x,k), otherwise we reject.

For the implication 3 ⇒ 1, let $L \in \mathsf{para\text{-}NP}$ and let M be a nondeterministic Turing machine for L running in time $f(k)p(n)$ where f is computable and p is a polynomial. Then we define the following proof system P for L:

$$P(x,k,w) = \begin{cases} (x,k) & \text{if } w \text{ is an accepting computation of } M \text{ on input } (x,k) \\ (x_0,k_0) & \text{otherwise} \end{cases}$$

where $(x_0,k_0) \in L$ is some fixed instance. Apparently, P can be computed in polynomial time. Moreover, P is fpt-bounded as every $(x,k) \in L$ has a P-proof of size $O(f(k)p(|x|))$. □

As in items 1 and 2 of Theorem 7.2.6, the two concepts of proof systems for parameterized languages also turn out to be equivalent with respect to other notions, for instance when defining parameterized simulations or considering the existence of optimal proof systems.

7.3 Parameterized Resolution

The classical proof system of Resolution was adapted by Dantchev, Martin, and Szeider [DMS07] to a parameterized proof system, called *Parameterized Resolution.* Parameterized Resolution is a refutation system for the set PCon of parameterized contradictions (cf. Definition 7.1.4). Given a set of clauses F in variables $x_1, \ldots, x_n$ with $(F, k) \in$ PCon, a *Parameterized Resolution refutation* of (F, k) is a Resolution refutation of

$$F \cup \{\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}} \mid 1 \leq i_1 < \cdots < i_{k+1} \leq n\} \ .$$

Thus, in Parameterized Resolution we have built-in access to all parameterized clauses of the form $\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$. All these clauses are available in the system, but when measuring the size of a derivation we only count those which appear in the derivation. Note that parameterized resolution is actually a proof system where verification proceeds in polynomial time.

As before, if refutations are tree-like we speak of *tree-like Parameterized Resolution.* Similarly as in classical tree-like Resolution (cf. Section 6.1), a tree-like Parameterized refutation of (F, k) can equivalently be described as a *Boolean decision tree.* A Boolean decision tree for (F, k) is a binary tree where inner nodes are labeled with variables from F and leafs are labeled with clauses from F or parameterized clauses $\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$. Each path in the tree corresponds to a partial assignment where a variable x gets value 0 or 1 according to whether the path branches left or right at the node labeled with x. The condition on the decision tree is that each path α must lead to a clause which is falsified by the assignment corresponding to α. It is easy to verify that each tree-like Parameterized Resolution refutation of (F, k) yields a Boolean decision tree for (F, k) and vice versa, where the size of the Resolution proof equals the number of nodes in the decision tree.

The main result of Dantchev, Martin, and Szeider [DMS07] on Parameterized Resolution is an extension of Riis' *gap theorem* [Rii01]. For this result, first-order sentences ψ are translated into sequences $\langle C_{n,\psi} \rangle_{n \in \mathbb{N}}$ of propositional formulas in CNF. Riis' gap theorem states that in tree-like Resolution, these translations of unsatisfiable first-order formulas have either polynomial-size refutations or need exponential-size refutations:

Theorem 7.3.1 (Riis [Rii01]) *Let ψ be a first-order sentence which fails in all finite models. Then for the propositional translations $\langle C_{n,\psi} \rangle_{n \in \mathbb{N}}$ of ψ, either of the following holds:*

1. *$C_{n,\psi}$ has tree-like Resolution refutations of size $n^{O(1)}$;*
2. *There exists a constant $\varepsilon > 0$ such that every tree-like Resolution refutation of $C_{n,\psi}$ has size at least $2^{\varepsilon \cdot n}$.*

Furthermore, case 2 holds if and only if ψ has an infinite model.

Dantchev, Martin, and Szeider prove that in the parameterized setting, the hard case in Riis' dichotomy (case 2) splits into two sub-cases:

Theorem 7.3.2 (Dantchev, Martin, Szeider [DMS07]) *Let ψ be a first-order sentence which fails in all finite models, but holds in some infinite model. Then for the sequence of parameterized contradictions $\langle (C_{n,\psi}, k)\rangle_{n\in\mathbb{N}}$ stemming from the translations of ψ, either of the following holds:*

2a. The parameterized contradictions $(C_{n,\psi}, k)$ have tree-like Parameterized Resolution refutations of size $2^{O(k)} \cdot n^{O(1)}$;

2b. There exists a constant $0 < \varepsilon \leq 1$ such that for every $n > k$, every tree-like Parameterized Resolution refutation of $(C_{n,\psi}, k)$ has size at least n^{k^ε}.

Furthermore, case 2b holds if and only if ψ has an infinite model whose induced hypergraph has no finite dominating set, where the hypergraph of a model M contains the elements of M as vertices and hyperedges are the tuples appearing in some relation in the model.

This gap theorem states that in tree-like Parameterized Resolution, translations of first-order formulas either have fpt-bounded refutations or require refutations of size similar to exhaustive search. Dantchev et al. provide examples of principles for all the three cases 1, 2a, and 2b.

7.4 A General Lower Bound for Parameterized Proof Systems

Before we analyze the complexity of of Parameterized Resolution in more detail, we will explain a very general lower bound argument which works for all parameterized proof systems where the parameterized axioms are given explicitly in the form $\neg x_1 \vee \cdots \vee \neg x_{k+1}$ as is the case in Parameterized Resolution. The lower bound exploits the fact that we can easily build minimally unsatisfiable parameterized contradictions where we need all these parameterized clauses for their refutation.

Proposition 7.4.1 *Parameterized Resolution is not fpt-bounded.*

Proof. Let $n, k \in \mathbb{N}$. We use propositional variables $x_{i,j}$ with $i \in [k+1]$ and $j \in [n]$. Consider the following formulas Θ_n^{k+1}:

$$\Theta_n^{k+1} := \bigwedge_{i\in[k+1]} \bigvee_{j\in[n]} x_{i,j} \quad .$$

We can think of these formulas as the collection of all the big clauses from the pigeonhole principle, but now we use $k+1$ pigeons and n holes. The formula Θ_n^{k+1} asserts that each of the $k+1$ pigeons goes into some hole. Clearly, the formulas (Θ_n^{k+1}, k) are parameterized contradictions. However, every Parameterized Resolution refutation of (Θ_n^{k+1}, k) has to involve all n^{k+1} parameterized axioms of the following set

$$\Pi := \{\, \neg x_{1,j_1} \vee \cdots \vee \neg x_{k+1,j_{k+1}} \mid j_1, \ldots, j_{k+1} \in [n] \,\} \ ,$$

because for each proper subset $\Pi' \subset \Pi$, the formula $\Theta_n^{k+1} \cup \Pi'$ is no classical contradiction and thus cannot be refuted in Resolution. This shows that any Parameterized Resolution refutation of (Θ_n^{k+1}, k) has size at least n^{k+1}. □

We remark that these lower bounds apply to all parameterized proof systems where the parameterized axioms are given explicitly. However, these lower bounds are somewhat trivial and not very informative about the actual strength of the systems. In the following sections we will therefore analyze the complexity of *natural* principles in tree-like and dag-like Parameterized Resolution. Let us remark though that Dantchev et al. [DMS07] also define parameterized proof systems where the parameterized axioms are given in a more succinct implicit encoding. Lower bounds for these systems, in particular for implicit Parameterized Resolution, are not yet known.

7.5 Tree-like Lower Bounds via Asymmetric Prover-Delayer Games

Dantchev, Martin, and Szeider [DMS07] proved that tree-like Parameterized Resolution is not fpt-bounded. However, their lower bound technique only works for formulas arising from propositional encodings of first-order principles having infinite models. To show hardness results in tree-like Parameterized Resolution for other sequences of parameterized contradictions we use our asymmetric Prover-Delayer game from Section 6.3. As in Section 6.3, we obtain the connection of the asymmetric Prover-Delayer game to the size of proofs in tree-like Parameterized Resolution. In fact, the asymmetric game is applicable to all tree-like proof systems. We state the result for tree-like Parameterized Resolution. Its proof is essentially the same as the proof of Theorem 6.3.1.

Theorem 7.5.1 *Let (F, k) be a parameterized contradiction and let c_0 and c_1 be two functions satisfying $\frac{1}{c_0(x,\alpha)} + \frac{1}{c_1(x,\alpha)} = 1$ for all partial assignments α to the variables of F. If (F, k) has a tree-like Parameterized Resolution refutation of size at most S, then the Delayer gets at most $\log S$ points in each (c_0, c_1)-game played on (F, k).*

We remark that for tree-like Parameterized Resolution it is indeed essential that we use the asymmetric game instead of the original Pudlák-Impagliazzo game. This is so because we aim for lower bounds of the form $n^{\Omega(k)}$ (n is the formula size and k the parameter), but, trivially, every tree-like Parameterized Resolution refutation only contains balanced sub-trees of height k as we can never set more than k variables to 1.

We illustrate the applicability of this technique for tree-like Parameterized Resolution by proving the hardness of the pigeonhole principle PHP^{n+1}_n (cf. Section 6.4). Again, proving the lower bound amounts to suitably choosing functions c_0 and c_1 and defining a Delayer-strategy for the (c_0, c_1)-game played on (PHP^{n+1}_n, k).

Theorem 7.5.2 *Any tree-like Parameterized Resolution refutation of (PHP^{n+1}_n, k) has size $n^{\Omega(k)}$.*

Proof. Let α be a partial assignment to the variables $\{x_{i,j} \mid i \in [n+1], j \in [n]\}$. Let $z_i(\alpha) = |\{j \in [n] \mid \alpha(x_{i,j}) = 0\}|$, i. e., $z_i(\alpha)$ is the number of holes already excluded by α for pigeon i. We define

$$c_0(x_{i,j}, \alpha) = \frac{n - z_i(\alpha)}{n - z_i(\alpha) - 1} \quad \text{and} \quad c_1(x_{i,j}, \alpha) = n - z_i(\alpha)$$

which apparently satisfies (6.1). We now describe Delayer's strategy in a (c_0, c_1)-game played on (PHP^{n+1}_n, k). If Prover asks for a value of $x_{i,j}$, then Delayer decides as follows:

set $\alpha(x_{i,j}) = 0$	if there exists $i' \in [n+1] \setminus \{i\}$ such that $\alpha(x_{i',j}) = 1$ or
	if there exists $j' \in [n] \setminus \{j\}$ such that $\alpha(x_{i,j'}) = 1$

set $\alpha(x_{i,j}) = 1$	if there is no $j' \in [n]$ with $\alpha(x_{i,j'}) = 1$ and $z_i(\alpha) \geq n - k$
let Prover decide	otherwise.

Intuitively, Delayer leaves the choice to Prover as long as pigeon i does not already sit in a hole, but there are at least k holes free for pigeon i, and there is no other pigeon sitting already in hole j. If Delayer uses this strategy, then clauses from PHP^{n+1}_n will not be violated in the game, i. e., a contradiction will always be reached on some parameterized clause. To verify this claim, let α be a partial assignment constructed during the game with $w(\alpha) \leq k$. Then, for every pigeon which has not been assigned to a hole yet, there are at least k holes where it could go (and of these only $w(\alpha)$ holes are already occupied by other pigeons). Thus α can be extended to a one-one mapping of exactly k pigeons to holes.

Therefore, at the end of the game exactly $k + 1$ variables have been set to 1. Let us denote by p the number of variables set to 1 by Prover and let d be the number of 1's assigned by Delayer. As argued before $p + d = k + 1$. Let us check how many points Delayer earns in this game. If Delayer assigns 1 to a variable $x_{i,j}$, then pigeon i was not assigned to a hole yet and, moreover, there must be $n-k$ holes which are already excluded for pigeon i by α, i. e., for some $J \subseteq [n]$ with $|J| = n-k$ we have $\alpha(x_{i,j'}) = 0$ for all $j' \in J$. Most of these 0's have been assigned by Prover, as Delayer has only assigned a 0 to $x_{i,j'}$ when some other pigeon was already sitting in hole j', and there can be at most k such holes. Thus, before Delayer sets $\alpha(x_{i,j}) = 1$, she has already earned points for at least $n-2k$ variables $x_{i,j'}$, $j' \in J$, yielding at least

$$\sum_{z=0}^{n-2k-1} \log \frac{n-z}{n-z-1} \;=\; \log \prod_{z=0}^{n-2k-1} \frac{n-z}{n-z-1} \;=\; \log \frac{n}{2k} \;=\; \log n - \log 2k$$

points for the Delayer. Let us note that because Delayer never allows a pigeon to go into more than one hole, she will really get the number of points calculated above for *every* of the d variables which she set to 1.

If, conversely, Prover sets variable $x_{i,j}$ to 1, then Delayer gets $\log(n - z_i(\alpha))$ points for this, but she also received points for most of the $z_i(\alpha)$ variables set to 0 before. Thus, in this case Delayer earns on pigeon i at least

$$\begin{aligned} & \log(n - z_i(\alpha)) + \sum_{z=0}^{z_i(\alpha)-k-1} \log \frac{n-z}{n-z+k-1} \\ = \;& \log(n - z_i(\alpha)) + \log \frac{n}{n - z_i(\alpha) + k} \\ = \;& \log n - \log \frac{n - z_i(\alpha) + k}{n - z_i(\alpha)} \\ \geq \;& \log n - \log k \end{aligned}$$

points. In total, Delayer gets at least

$$d(\log n - \log 2k) + p(\log n - \log k) \geq k(\log n - \log 2k)$$

points in the game. Applying Theorem 7.5.1, we obtain $(\frac{n}{2k})^k$ as a lower bound to the size of each tree-like Parameterized Resolution refutation of (PHP_n^{n+1}, k). □

By inspection of the above Delayer strategy it becomes clear that the lower bound from Theorem 7.5.2 also holds for the *functional pigeonhole principle* where in addition to the clauses from PHP_n^{n+1} we also include $\neg x_{i,j_1} \vee \neg x_{i,j_2}$ for all pigeons $i \in [n+1]$ and distinct holes $j_1, j_2 \in [n]$.

7.6 Kernels and Small Refutations

The notion of *efficient kernelization* plays an important role in the theory of parameterized complexity. A kernelization for a parameterized language L is a polynomial-time procedure $A : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that for each (x, k)

1. $(x,k) \in L$ if and only if $A(x,k) \in L$ and

2. if $A(x,k) = (x',k')$, then $k' \leq k$ and $|x'| \leq f(k)$ for some computable function f independent of $|x|$.

It is clear that if a parameterized language admits a kernelization, then the language is fixed-parameter tractable, but also the converse is true (cf. [FG06]). For parameterized proof complexity we suggest a similar notion of kernel for parameterized contradictions:

Definition 7.6.1 *A set* $\Gamma \subseteq \mathrm{PCon}$ *of parameterized contradictions has a* kernel *if there exists a computable function* f *such that every* $(F,k) \in \Gamma$ *has a subset* $F' \subseteq F$ *of clauses satisfying the following conditions:*

1. F' *contains at most* $f(k)$ *variables and*

2. (F',k) *is a parameterized contradiction.*

Note that a sequence of parameterized contradictions with a kernel of size $f(k)$ admits fpt-bounded tree-like Resolution refutations of size at most $2^{f(k)}$. Nevertheless, there are CNF's without a kernel, but with fpt-bounded refutations, for example $(x_1 \vee x_2 \vee \dots \vee x_n) \wedge \neg x_1 \wedge \neg x_2 \wedge \dots \wedge \neg x_n$.

We now give some examples of CNF's with kernels:

Pebbling contradictions. Fix a constant l and an acyclic connected directed graph G of constant maximum in-degree with a single sink vertex z. For any vertex v in G, let $\mathit{Pred}(v)$ be the set of immediate predecessors of v. For any v we use the propositional variables $x_1^v, \dots, x_l^v$. The *pebbling contradiction* consists of the conjunction of the constraints

$$\left(\bigwedge_{u \in \mathit{Pred}(v)} (x_1^u \vee \ldots \vee x_l^u)\right) \longrightarrow x_1^v \vee \ldots \vee x_l^v$$

for any $v \in V(G)$ and constraints $\neg x_1^z, \neg x_2^z, \dots, \neg x_l^z$. This means that for any source vertex s (which has an empty set of predecessors) one of the variables x_i^s is true. By induction this holds for every vertex in G, in particular also for the sink z contradicting the last l clauses. For constant l and constant maximum in-degree, the pebbling formula can be encoded as a CNF of polynomial size in the number of vertices of G.

To see that the pebbling contradictions have a kernel, consider the first $k+1$ vertices in a topological ordering of $V(G)$. The corresponding propagation formulas form a parameterized contradiction, because these formulas enforce $k+1$ true variables. For graphs of constant maximum in-degree, these formulas have $O(1)$ variables each, so their CNF encoding has size $O(k)$ and constitutes a kernel.

This is remarkable, because forms of pebbling tautologies are hard for tree-like Resolution in the non-parameterized setting [BSIW04].

Colorability. Fix a constant c and a graph G which is not c-colorable. The *c-coloring contradiction* is defined on variables $p_{u,j}$ where u is a vertex of G and j is one of the c colors. The CNF claims that G is c-colorable: (1) for any vertex u we have the clause $\bigvee_{1\leq j\leq c} p_{u,j}$ claiming that the vertex u gets a color; (2) for any edge $\{u, v\}$ in G and any color j, the clause $\neg p_{u,j} \vee \neg p_{v,j}$ claims that no pair of adjacent vertices gets the same color.

It is easy to see that the clauses of type (1) corresponding to any $k+1$ vertices form a kernel and thus have a short refutation. This contrasts with the fact that for random G and $c \geq 3$ this formula is hard in Resolution [BCMM05].

Graph pigeonhole principle. Fix G to be a bounded-degree bipartite graph, with the set of vertices partitioned into two sets: U of size $n+1$ (the pigeons) and V of size n (the holes). $PHP(G)$ is a variant of the pigeonhole principle where a pigeon can go only into a small set of holes as specified by the edges of G. If G is an expander, then the principle is hard for Resolution [BSW01]. $PHP(G)$ consists of the following clauses: (1) for $u \in U$ we have $\bigvee_{v\in\Gamma(u)} p_{u,v}$; (2) for any $v \in V$ and any $u_1, u_2 \in \Gamma(v)$ we have the clause $\neg p_{u_1,v} \vee \neg p_{u_2,v}$. It is clear that $k+1$ clauses of type (1) constitute a kernel.

Ordering principles. In Section 7.7 we will give different formulations of the total ordering principle. We will see that the propositional translations of the first-order formulation given in [DMS07] have easy refutations (as observed in [DMS07]) because of the presence of a kernel. We emphasize that the same ordering principle requires exponential-size tree-like Resolution refutations in the non-parameterized setting [BG01].

Bounded-width CNF. The kernels in the previous examples are very explicit, but this is not always the case. Is it easy to find a kernel if it is known to exist? The answer to this question has consequences regarding automatizability of tree-like Parameterized Resolution. We see now a general strategy for finding kernels and fpt-bounded refutations for parameterized contradictions of bounded width. We first explain this technique for vertex cover and then generalize it for bounded width CNF's.

Gao [Gao09] suggested to use a standard DPLL algorithm to find refutations of certain random parameterized d-CNF's. Here we prove that bounded width CNF's have a kernel and hence are efficiently refutable in tree-like Parameterized Resolution. The core of our quite simple argument is the interpretation of a classical parameterized algorithm for vertex cover as a DPLL procedure.

A vertex cover for a graph G is a set $C \subseteq V(G)$ such that for any $\{u, v\} \in E(G)$ either $u \in C$ or $v \in C$. To determine whether G has a vertex cover of size at most k there is a well-known [DF99, Chapter 3] fixed parameter tractable algorithm (here the parameter is k). This algorithm is based on the following observation: if a vertex is not in C, then all its neighbors must be in C. The algorithm is a simple recursive procedure which focuses on an arbitrary vertex u, and on its neighbors $v_1, \ldots, v_l$: if neither $G \setminus \{u\}$ has a vertex cover of size $k-1$ nor $G \setminus \{u, v_1, \ldots, v_l\}$ has a vertex cover of size $k-l$, then G has no vertex cover of size k.

This is easily interpretable as a parameterized DPLL procedure on the 2-CNF $F_G = \bigwedge_{\{u,v\} \in E(G)} (x_u \vee x_v)$ where x_u indicates whether $u \in C$. The DPLL procedure fixes an arbitrary variable x_u and splits on it. When $x_u = 1$, then the DPLL algorithm proceeds with analyzing $F_G \restriction_{x_u=1}$ which is equal to $F_{G \setminus \{u\}}$. When $x_u = 0$, then $x_{v_1} = 1, \ldots, x_{v_l} = 1$ by unit propagation. Thus the DPLL proceeds on formula $F_G \restriction_{\{x_u=0, x_{v_1}=1, \ldots, x_{v_l}=1\}} = F_{G \setminus \{u, v_1, \ldots, v_l\}}$. If at any point the DPLL has more than k variables set to one, it stops and backtracks. In Theorem 7.6.2 we extend this idea to a DPLL algorithm for bounded width formulas.

Theorem 7.6.2 *If F is a CNF of width d and (F, k) is a parameterized contradiction, then (F, k) has a tree-like Parameterized Resolution refutation of size $O(d^{k+1})$. Moreover, there is an algorithm that for any (F, k) either finds such tree-like refutation or finds a satisfying assignment for F of weight $\leq k$. The algorithm runs in time $O(|F| \cdot k \cdot d^{k+1})$.*

Proof. Assume (F, k) is a parameterized contradiction. We want to find a refutation for F with parameter k (i.e., at most k variables can be set to true). We first consider a clause $C = x_1 \vee x_2 \vee \ldots \vee x_l$ where $l \leq d$ with all positive literals. Such clause exists because otherwise the full zero assignment would satisfy F.

By induction on k we will prove that (F, k) has a parameterized tree-like refutation of size at most $2 \cdot \sum_{i=0}^{k+1} d^i - 1$. For $k = 0$ the clauses $\{\neg x_i\}_{i=1}^{l}$ are parameterized axioms of the system, thus C is refutable in size at most $1 + 2l \leq 1 + 2d$.

Now consider $k > 0$. For any $1 \leq i \leq l$, let F_i be the restriction of F obtained by setting $x_i = 1$. Each $(F_i, k-1)$ is a parameterized contradiction, otherwise (F, k) would not be. By inductive hypothesis $(F_i, k-1)$ has a tree-like refutation of size at most $s = 2 \sum_{i=0}^{k} d^i - 1$. This refutation can be turned into a tree-like derivation of $\neg x_i$ from (F, k). Now we can derive all $\neg x_i$ for $1 \leq i \leq l$ and refute clause C. Such refutation has length $1 + l + ls \leq 1 + d + ds = 2 \cdot \sum_{i=0}^{k+1} d^i - 1$.

By inspection of the proof, it is clear that the refutation can be computed by a simple procedure which at each step looks for a clause C with only positive literals, and builds a refutation of (F, k) recursively by: building l refutations of $(F_i, k-1)$; turning them in l derivations $(F, k) \vdash \neg x_i$; and resolving against C. This procedure can be easily implemented in the claimed running time.

So far we considered (F, k) to be a parameterized contradiction. If that is not the case, then the algorithm fails. It can fail in two ways: (a) it does not find a clause with only positive literals; (b) one among $(F_i, k-1)$ is not a parameterized contradiction. The algorithm will output the full zero assignment in case (a) and

$\{x_i = 1\} \cup \alpha$ in case (b), where α is an assignment witnessing $(F_i, k-1) \notin$ PCon. By induction we can show that on input (F, k) this procedure returns a satisfying assignment of weight $\leq k$. □

We state two interesting consequences of this result.

Corollary 7.6.3 *For each $d \in \mathbb{N}$, the set of all parameterized contradictions in d-CNF has a kernel.*

Proof. The refutations constructed in Theorem 7.6.2 contain $O(d^k)$ initial clauses in $O(d^{k+1})$ variables. These clauses form a kernel. □

The following corollary expresses some restricted form of automatizability (cf. also the discussion in Section 7.9).

Corollary 7.6.4 *If $\Gamma \subseteq$ PCon has a kernel, then there exists an fpt-algorithm which on input $(F, k) \in \Gamma$ returns both a kernel and a refutation of (F, k).*

Proof. Let Γ have a kernel of size $f(k)$. Then the kernel only contains clauses of width $\leq f(k)$. On input (F, k) we run the algorithm of Theorem 7.6.2 on the CNF formula consisting of all clauses of F with width $\leq f(k)$. This yields a kernel together with its refutation. □

Tseitin tautologies. Fix a bipartite graph $G = (L, R, E)$ such that the degree of the vertices on the left side is constant and the degree of all vertices on the right side is even. Fix now an arbitrary Boolean function $f : L \to \{0, 1\}$ such that $\sum_{u \in L} f(u) \equiv 1 \pmod 2$. The Tseitin tautology for (G, f) claims that there is no way to define $g : R \to \{0, 1\}$ such that for any $u \in L$, $\sum_{v \in \Gamma(u)} g(v) \equiv f(u) \pmod 2$. This fact follows by a simple parity argument. The CNF formulation of this claim uses variables x_v for $v \in R$. The CNF is constituted by the encoding of the constraints $\sum_{v \in \Gamma(u)} x_v \equiv f(u) \pmod 2$ for every $u \in L$. Each linear constraint requires exponential size to be represented in CNF, but this is not an issue here because left-side vertices have constant degree. Hence Tseitin formulas have bounded width. By Theorem 7.6.2 they have a kernel, but in contrast to our previous examples this kernel is not very explicit.

Corollary 7.6.5 *There are formulas (e. g. Tseitin tautologies) which are hard for general Resolution but easy for tree-like Parameterized Resolution.*

7.7 Ordering Principles

In this section we discuss Parameterized Resolution refutations for various *ordering principles OP*, also called *least element principles.* The principle claims that any finite partially ordered set has a minimal element. There is a direct propositional translation of OP to a family OP_n of CNF's. Each CNF OP_n expresses that there exists a partially ordered set of size n such that any element has a predecessor. We are also interested in the *linear ordering principle* LOP in which the set is required to be *totally* ordered.

Dantchev, Martin, and Szeider [DMS07] show that a propositional formulation of LOP has small refutations in tree-like Parameterized Resolution. They also show that such efficient refutation does not exist for OP. We observe that their formulation of LOP has short proofs because it contains very simple kernels. We describe LOP^*, an alternative formulation of the linear ordering principle which does not contain a kernel but nevertheless has (less trivial) fpt-bounded tree-like refutations.

We now describe the three propositional formulations of the ordering principles. For a model with n elements, OP_n, LOP_n, and LOP^*_n are three CNF's over variables $x_{i,j}$ for $i \neq j$ and $i, j \in [n]$.

OP: the general ordering principle has the following clauses:

$$\begin{array}{lll} \neg x_{i,j} \vee \neg x_{j,i} & \text{for every } i, j & \text{(Antisymmetry)} \\ \neg x_{i,j} \vee \neg x_{j,k} \vee x_{i,k} & \text{for every } i, j, k & \text{(Transitivity)} \\ \bigvee_{j \in [n] \setminus \{i\}} x_{j,i} & \text{for every } i & \text{(Predecessor)} \end{array}$$

LOP: is the same as OP with the addition of totality constraints:

$$\begin{array}{lll} x_{i,j} \vee x_{j,i} & \text{for every } i, j & \text{(Totality)} \end{array}$$

LOP*: is a different encoding of LOP where we consider only variables $x_{i,j}$ for $i < j$. The intended meaning is that $x_{i,j}$ is true whenever j precedes i in the ordering, and false if i precedes j. The reader may think $x_{i,j}$ to indicate if i and j are an inversion in the permutation for the indexes described by the total order. In particular the full true assignment represents the linear order $(n, n-1, n-2, \ldots, 2, 1)$ while the full false assignment represents $(1, 2, \ldots, n-2, n-1, n)$. This representation will help in the proof of Theorem 7.7.1.

LOP^*_n is obtained by substituting in LOP_n any occurrence of $x_{j,i}$ for $j > i$ with $\neg x_{i,j}$. In this way all totality and antisymmetry clauses vanish, and transitivity translates according to relative ranks of the involved indexes.

$$\begin{array}{lll} \neg x_{i,j} \vee \neg x_{j,k} \vee x_{i,k} & \text{for all } i < j < k & \text{(Transitivity 1)} \\ x_{i,j} \vee x_{j,k} \vee \neg x_{i,k} & \text{for all } i < j < k & \text{(Transitivity 2)} \\ \bigvee_{j<i} \neg x_{j,i} \vee \bigvee_{i<j} x_{i,j} & \text{for all } i & \text{(Predecessor)} \end{array}$$

Both OP and LOP are the canonical propositional translations of the first-order formulations of the general and total ordering principle, respectively. In

[DMS07] the upper bound for LOP and the lower bound for OP are proved by the model-theoretic criteria from Theorem 7.3.2 on the first-order logic formulations.

We remark that in the non-parameterized setting, neither OP, LOP, nor LOP^* have short tree-like Resolution refutations [BG01], but all of them have general Resolution refutations of polynomial size [Stå96]. It is interesting that in the parameterized setting LOP and LOP^* become easy for tree-like, while OP remains hard. Thus, OP provides a separation between tree-like and dag-like Parameterized Resolution.

It is easy to see that LOP has short tree-like refutations in Parameterized Resolution: notice that the totality clauses for any $k+1$ pairs of indexes form a parameterized contradiction of $2k+2$ variables at most, and so they are a kernel. Unfortunately, LOP is easy to refute for uninteresting reasons: the kernel is very simple. The alternative formulation LOP^* does not have a kernel because all clauses of bounded width are satisfiable by the all zero assignment which represents a total order. Nevertheless LOP^* admits fpt-bounded tree-like refutations.

Theorem 7.7.1 *The formulas LOP_n^* have fpt-bounded tree-like refutations in Parameterized Resolution.*

Proof. Let (LOP_n^*, k) be the given instance and assume w. l. o. g. that $k \leq n$. We are going to derive LOP_{k+1}^* from LOP_n^* in polynomial length. This concludes the proof of the theorem because LOP_{k+1}^* has $O(k^2)$ variables and consequently has a tree-like refutation of length $2^{O(k^2)}$.

The idea of the refutation is that for any total order either the least element is among $1, \ldots, k+1$ or there is an element less than all of them. This means that there are at least $k+1$ inversions with respect to the canonical order (i. e., $k+1$ variables are set to 1). To obtain LOP_{k+1}^* we have to derive

$$\bigvee_{1 \leq j < i} \neg x_{j,i} \vee \bigvee_{i < j \leq k+1} x_{i,j}$$

for any $1 \leq i \leq k+1$. W. l. o. g. we discuss the case $i = 1$ which requires simpler notation, the other k cases are analogous.

Our goal then is to derive $\bigvee_{1<j\leq k+1} x_{1,j}$. For any $l > k+1$ consider the following clauses: the first is an axiom of Parameterized Resolution, the others are transitivity axioms.

$$\neg x_{1,l} \vee \neg x_{2,l} \vee \ldots \vee \neg x_{k+1,l} \tag{7.1}$$

$$x_{1,2} \vee x_{2,l} \vee \neg x_{1,l} \tag{7.2}$$

$$x_{1,3} \vee x_{3,l} \vee \neg x_{1,l} \tag{7.3}$$

$$\vdots$$

$$x_{1,k+1} \vee x_{k+1,l} \vee \neg x_{1,l} \tag{7.4}$$

By applying Resolution between clause (7.1) and the transitivity clauses we obtain

$$x_{1,2} \vee x_{1,3} \vee \ldots \vee x_{1,k+1} \vee \neg x_{1,l} \tag{7.5}$$

We just proved that if 1 is the least index among the first $k+1$, then no index above $k+1$ can be less than 1, otherwise there would be at least $k+1$ true variables. The predecessor constraint for 1 contains the literal $x_{1,l}$ for every l; thus applying Resolution between that and clause (7.5) for every $l > k+1$ yields $\bigvee_{1<j\leq k+1} x_{1,j}$.

We obtained the predecessor axiom for index 1 in LOP^*_{k+1} by a derivation of size $O(kn)$. With $k+1$ such deductions we obtain LOP^*_{k+1}. As the whole refutation of LOP^*_n has length $O(k^2n) + 2^{O(k^2)}$, it is fpt-bounded. □

The following theorem has been first proved in [DMS07]. Their proof is based on the model-theoretic criterion from Theorem 7.3.2. We give a combinatorial proof based on Prover-Delayer games.

Theorem 7.7.2 *Any tree-like Parameterized Resolution refutation of* (OP_n, k) *has size* $n^{\Omega(k)}$.

Proof. Let α be an assignment to the variables of OP. The Delayer will keep the following information:

- $G(\alpha) = (V(\alpha), E(\alpha))$ the graph obtained taking as edges the (i,j)'s such that $\alpha(x_{i,j}) = 1$;
- $G^*(\alpha)$ the transitive closure of $G(\alpha)$ and $G^T(\alpha)$ the transpose graph of $G(\alpha)$.

In particular, for any vertex j in $G(\alpha)$, the Delayer considers the following information

- $z_j(\alpha) = |\{i \in [n] \mid \alpha(x_{i,j}) \text{ is not assigned}\}|$,
- $Pred_j(\alpha) = \{i \in [n] \mid \alpha(x_{i,j}) = 1\}$, and
- $PPred_j(\alpha)$ the subset of $Pred_j(\alpha)$ of those edges set to 1 by the Prover .

Loosely speaking the Delayer, taking as few decisions as possible, wants to force: (1) the game to end on a parameterized clause, and (2) the Prover to decide only one predecessor for each node. To reach the former, in some cases she will be forced to decide a predecessor of a node j to avoid that after few more trivial queries the game ends on a predecessor clause. To get (2) she will be forced to say that some node can't be predecessor of some node j. In both cases we will prove that Delayer will keep her number of decisions bounded.

Let α be the assignment built so far in the game and let $x_{i,j}$ be the variable queried by Prover. Delayer acts as follows:

1. if $(i,j) \in E(\alpha)^*$, then answer 1;
2. if $(i,j) \in (E(\alpha)^*)^T$, then answer 0;
3. if $|Pred_j(\alpha)| = 0$ and $z_j(\alpha) \leq k+1$, then answer 1;
4. if $|PPred_j(\alpha)| \geq 1$, then answer 0;
5. otherwise, she leaves the decision to the Prover.

To simplify the argument we assume that in the game, after each decision by the Prover or after a decision by the Delayer according to Rule (3), the Prover asks all variables corresponding to edges that are in $G^*(\alpha)$ and $(G(\alpha)^*)^T$ but not in $G(\alpha)$. This will not change our result since on these nodes Delayer does not score any point.

Let $P^\epsilon(t)$ be the set of edges set to $\epsilon \in \{0,1\}$ by the Prover after stage t ends. Let $D^\epsilon(t)$ be the set of edges set to $\epsilon \in \{0,1\}$ by the Delayer. Finally, let $D^*(t) \subseteq D^1(t)$ be the set of edges set to 1 by the Delayer according to Rule (3) of her strategy. $P^\epsilon_j(t)$, $D^\epsilon_j(t)$, and $D^*_j(t)$ are the subsets of the respective sets formed by those edges having end-node j, i. e., edges of the form (i,j) for some i.

Let α_t be the assignment built after stage t and let α^*_t be the extensions of α_t obtained by assigning all edges from $G^*(\alpha_t)$ to 1 and all edges from $(G(\alpha_t)^*)^T$ to 0. We define $N_j(t) = \{\, (i,j) \mid i \in [n], (i,j) \in dom(\alpha^*_t) \setminus P^0(t) \,\}$.

Lemma 7.7.3 *At each stage t of the game, it holds:*

1. $|P^1(t)| + |D^*(t)| \geq \sqrt{|E(\alpha_t)|}$;
2. *if* $|P^1_j(t)| + |D^*_j(t)| = 0$, *then* $|N_j(t)| \leq k$;
3. *if* $w(\alpha_t) \leq k$, *then* α^*_t *does not falsify any predecessor clause;*
4. *for each* $j \in [n]$, $|D^*_j(t)| \leq 1$ *and* $|P^1_j(t)| \leq 1$.

Proof. Condition (1) follows since $|P^1(t)| + |D^1(t)| = |E(\alpha_t)|$, and $|E(\alpha_t)| \leq |E^*(\alpha_t)| \leq (|P^1(t)| + |D^*(t)|)^2$.

Condition (2): $|P^1_j(t)| + |D^*_j(t)| = 0$ implies that the vertex j has no predecessor. The only way to set a predecessor to a vertex which already has one is by Rule (1), but a vertex without predecessors cannot get one by transitive closure. Then an edge $x_{i,j}$ is in $dom(\alpha^*_t) \setminus P^0(t)$ if and only if i is a successor of j in $G^*(\alpha_t)$. Hence there must be a directed tree rooted in j and containing all such successors. As $G(\alpha_t)^T$ contains at most k edges, there are at most k successors of j. Hence $|N_j(t)| \leq k$.

Condition (3): consider a predecessor clause C_j which is not satisfied by α_t. Then there are at least $k+1$ variables $x_{i,j}$ unset, since otherwise, according to Rule (3) Delayer should have set one predecessor for j. If $|P^1_j(t)| \geq 1$, then C_j

would be satisfied. Then by $|P^1_j(t)| + |D^*_j(t)| = 0$ and by condition (2) at most k additional literals of C_j are set to 0 by α^*_t. The claim follows since there is at least one unset literal in C_j.

Condition (4): the first time that a predecessor of some node j is decided in the game is either by a decision of the Prover or by a decision of the Delayer according to Rule (3). Since Delayer applies Rule (3) only in the case no predecessor has been yet decided, it follows that $|D^*_j(t)| \leq 1$. Moreover, by Rule (4) Delayer prevents the Prover to set more than one predecessor for each node, hence $|P^1_j(t)| \leq 1$. □

Lemma 7.7.4 *After the last stage f of the game the following holds:*

- *a parameterized clause is falsified;*
- $|P^1(f)| + |D^*(f)| \geq \sqrt{k+1}$.

Proof. For the first condition, we notice that Rules (1) and (2) in the Delayer's strategy guarantee that neither antisymmetry nor transitivity axioms will be ever falsified during the game. Assuming that α_f has weight strictly less then $k+1$, then by Lemma 7.7.3 (part 3), no predecessor clause is falsified. Hence $w(\alpha_f) = k+1$ and a parameterized clause is falsified.

The second property follows by Lemma 7.7.3 (part 1) and by $|E(\alpha_f)| \geq w(\alpha_f)$ which is equal to $k+1$ because of the first part of this lemma. □

Set $c_1(x_{i,j}, \alpha) = z_j(\alpha)$ and $c_0(x_{i,j}, \alpha) = \frac{z_j(\alpha)}{z_j(\alpha)-1}$. For a given play of the game, let $t_{i,j}$ be the stage of the game when the variable $x_{i,j}$ is set. Let $sc_j(t)$ be the number of points scored by the Delayer up to stage t for answers of the Prover to the variables $x_{1,j}, x_{2,j}, \ldots, x_{n,j}$. Then the number of points scored by the Delayer at the end of the game is $\sum_{j=1}^{n} sc_j(f)$.

Lemma 7.7.5 *The following implications hold*

1. *If $|P^1_j(f)| = 1$, then $sc_j(f) \geq \log n - \log(k+1)$.*
2. *If $|D^*_j(f)| = 1$, then $sc_j(f) \geq \log n - \log(2k+1)$.*

Proof. For the first claim, let $(i,j) \in D^*_j(f)$ and let $t_{i,j}$ be the stage when $x_{i,j}$ was set. We claim that $|P^0_j(t_{i,j})| \geq n - (2k+1)$. W. l. o. g. we can assume that the variables $x_{i',j}$ set to 0 by the Prover are the first ones with end-node j to be set to 0, because $c_0(x_{i',j}, \alpha)$ is strictly decreasing with respect to $z_j(\alpha)$. Hence the Delayer gets at least

$$\sum_{l=n}^{2k+2} \log \frac{l}{l-1} = \log n - \log(2k+1)$$

points on variables $x_{1,j}, \ldots, x_{n,j}$.

It remains to prove the claim that $|P^0_j(t_{i,j})| \geq n - (2k+1)$. According to Rule (3) of the strategy, there are at least $n - (k+1)$ variables $x_{i',j}$ set to 0 in $\alpha_{t_{i,j}}$. Hence $|P^0_j(t_{i,j})| + |D^0_j(t_{i,j})| \geq n - (k+1)$. Since at this stage i is the first predecessor of j to be fixed, then the Delayer has not set variables $x_{i',j}$ to 0 according to Rule (4), but only by Rule (2). Moreover, for the same reason, if t' is the stage preceding $t_{i,j}$ we have that: $|D^0_j(t_{i,j})| = |D^0_j(t')| = |N_j(t')| \leq k$, where the last inequality holds by Lemma 7.7.3 (part 2). Then $|P^0_j(t_{i,j})| \geq n-(2k+1)$.

We now show the second claim of the lemma. Let $t_{i,j}$ be the stage in which Prover sets some $x_{i,j}$ to 1, and let α be the partial assignment corresponding to that stage. W. l. o. g. we assume that all variables in $P^0_j(t_{i,j})$ are set before any variable in $D^0_j(t_{i,j})$, because c_0 is monotone decreasing in the size of the second argument. Fix $p = |P^0_j(t_{i,j})|$. By Lemma 7.7.3 (part 2) we get $|N_j(t')| \leq k$ where t' is the stage preceding $t_{i,j}$. Hence we know that $z_j(\alpha) \geq n-k-p$. The amount of points got by Delayer on vertex j is at least

$$\sum_{l=n}^{n-p+1} \log \frac{l}{l-1} + \log(n-k-p) = \log n - \log \frac{n-p}{n-k-p} \geq \log n - \log(k+1) \ .$$

□

The Delayer scores $\sum_{j=1}^{n} sc_j(f)$. By Lemma 7.7.4 there are at least $\sqrt{k+1}$ vertices such that either $|D^*_j(f)| \geq 1$ or $|P^1_j(f)| \geq 1$. For each vertex such events are mutually exclusive by the definition of the rules. Then by Lemma 7.7.5 Delayer gets at least $\sqrt{k+1}(\log n - \log(2k+1))$ points. By Theorem 7.5.1 we get the lower bound. □

7.8 Hardness of the Pigeonhole Principle in Parameterized Resolution

In this section we answer a question raised by Dantchev, Martin, and Szeider [DMS07, Section 4]: What is the complexity of the pigeonhole principle (PHP) in dag-like Parameterized Resolution? Clearly this question becomes even more interesting in the light of our previous upper bounds. Using a certain encoding of the parameterized axioms—based on the same pigeonhole principle—one can get efficient proofs of PHP in Parameterized Resolution (Proposition 17 in [DMS07]). In contrast, we show that dag-like Parameterized Resolution is not fpt-bounded by proving that PHP requires proofs of size $n^{\Omega(k)}$ (Theorem 7.8.2). Again for this lower bound we use an interpretation of proofs in Parameterized Resolution as games. Here we employ a slight modification of Pudlák's game for Resolution lower bounds as devised in [Pud00], although, as we argue below, the analysis for the lower bound of PHP in Resolution does not suffice for proving lower bounds in Parameterized Resolution. Our result on dag-like Parameterized Resolution proofs for PHP represents the next step in the program proposed by the work of [DMS07] that approaches the separation of FPT from W[2] as in the Cook-Reckhow program of separating NP from coNP for classical proof complexity.

In studying lower bounds for Parameterized Resolution one is immediately faced with the following observations that exclude some techniques used for proving lower bounds in classical Resolution. First, Weighted CNF Sat is not resilient to restrictions in the sense that if we apply a restriction to a parameterized contradiction (F, k) we obtain (F', k') with $k' \leq k$. Then, either we use restrictions with only 0's which does not make sense, or we restrict k too much. The *width* lower bound method for Resolution [BP96, BSW01], which is based on restrictions, can therefore be excluded from the bag of techniques to prove lower bounds in Parameterized Resolution. Also one has to take into account that constant width CNF's have efficient tree-like Parameterized Resolution refutations. This implies that any technique employed to prove lower bounds in Parameterized Resolution for formulas of high initial width must not be preserved under the use of *extension variables* which reduce the width of a formula to a constant.

We decide to study the complexity of the pigeonhole principle in Parameterized Resolution employing a refined analysis of the original proof by Haken [Hak85], but using the interpretation of his method as a game given by Pudlák in his work [Pud00].

7.8.1 Parameterized Proofs as Games

While the game of Pudlák [Pud00] is also played between Prover and Delayer, it is very different from our asymmetric Prover-Delayer game of Section 6.3. In the asymmetric game, Delayer is using *one* strategy which tries to force Prover on a long path, thereby showing that the proof tree is large. In Pudlák's game, the Delayer uses a *family* of strategies (called "superstrategy" by Pudlák) with the aim to force Prover through every part of the proof.

We will now explain how Pudlák's game from [Pud00] can be adapted to show lower bounds in Parameterized Resolution. Consider any Parameterized Resolution refutation of a parameterized contradiction (F, k). An arbitrary assignment falsifies the empty clause at the end of the refutation, and because of soundness of the inference rule, one of its predecessor clauses must be falsified as well. The argument can be repeated, finding a falsified predecessor for any falsified clause in the refutation. This defines a walk in the refutation which starts at the empty clause and goes back to one of the initial clauses or a parameterized axiom, touching just falsified clauses. In the case of (Parameterized) Resolution it is easy to determine with a single variable query what predecessor is falsified: if $A \vee B$ is inferred from $A \vee x$ and $B \vee \neg x$, knowing the value of x is sufficient to walk back one step.

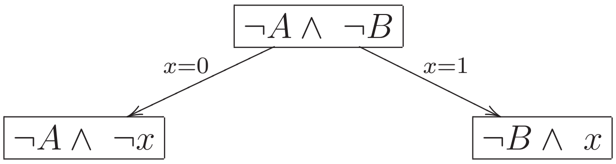
This allows to interpret a refutation as the following interactive process: a *Delayer* claims to know an assignment with at most k variables set to 1 and which satisfies F. The *Prover* queries the Delayer for variable values, can save such values in memory and can forget some of them at will. The Prover wants to expose Delayer lie by exhibiting either an initial or a parameterized clause which is falsified by the assignment. Following Pudlák, we call the memory configuration of the Prover at any step a *record*.

A *Prover strategy* is a finite directed acyclic graph with a single source marked by the empty record. The strategy must (a) define which variable to query in the next step; (b) define what values queried so far are kept in the record; (c) expose the Delayer at any leaf record. The Prover complexity is the size of the strategy, which is the number of distinct records that appear in all possible games (i. e., we ignore the behavior of Prover on records which are not reachable from the initial empty record).

There is a 1-1 correspondence between Prover strategies and parameterized refutations of (F, k). Namely, a parameterized refutation of (F, k) can be used by Prover to traverse it backwards from the empty clause up to the initial or parameterized clauses, with the invariant that at each step he keeps in his record the set of values which falsify the clause in the proof. Any inference of the refutation translates to a step in such exploration as

$$\frac{A \lor x \qquad B \lor \neg x}{A \lor B}$$

corresponds to



where Prover saves new information in his record (the value of x), but also deletes some previous information (corresponding to $\neg A$ or $\neg B$). It is easy to see that also the reverse translation from strategies into proofs is possible: the negation of the records in a strategy translate into clauses, deletions into weakening, and queries into inference steps.

7.8.2 Delayer Strategies as Refutation Lower Bounds

Because of this correspondence between refutations and Prover strategies, we want to show that Prover always needs a high ($n^{\Omega(k)}$) number of records to expose the Delayer. Notice that a Prover strategy which never forgets information reaches a falsified initial clause in less steps, but then a record occurs in less games. Instead, a record appearing in different games typically contains little information, thus a strategy of small size may require Prover to forget a lot of information. Indeed a Prover strategy which never forgets information corresponds to a tree-like Parameterized Resolution refutation, which is a weaker proof system than general Parameterized Resolution.

So far we argued that Prover strategies represent refutations. We now show that a *randomized Delayer strategy* is the key to obtain lower bounds on the size of such refutations. We may think of such a Delayer strategy as a collection of single strategies (Pudlák [Pud00] calls this a "superstrategy") which force Prover to generate many records. Without loss of generality we also allow Delayer to *give up*, which lets Prover win even before exposing Delayer. The key to lower bounds is then given by the following fact:

Fact 7.8.1 *Fix a Prover strategy $\mathcal{P}$ which has a set of records R, and a randomized Delayer strategy $\mathcal{D}$. If for each record $r \in R$*

$$\Pr_{\mathcal{D}}[\textit{ Prover wins on record } r\,] \le \frac{1}{S}$$

then $|R| \ge S$.

Proof. The probability that strategy P wins against the Delayer is at most $\frac{|R|}{S}$ by union bound, and is at least 1 by definition. This proves the claim. □

To show exponential lower bounds for general Resolution, Pudlák [Pud99] analyses the probability that a given Prover record is *compatible* with a chosen Delayer strategy, i. e., he bounds the probability that a record can appear during some game. However, this analysis does not suffice to prove lower bounds in Parameterized Resolution. Instead of *compatibility*, we therefore focus on the stronger notion that a given record is *winning* in a game and we show that each Prover record can only be winning against a small fraction of Delayer strategies.

7.8.3 The Lower Bound for the Pigeonhole Principle

Theorem 7.8.2 *Any refutation of (PHP^{n+1}_n, k) in Parameterized Resolution requires size $n^{\Theta(k)}$.*

Proof. For the upper bound we can easily build a decision tree which explores all $n^{O(k)}$ assignments with weight at most k.

The lower bound is based on the game interpretation of a Resolution refutation. Any line in the refutation is a game position between Prover and Delayer. We show that for any refutation (i. e., any Prover) there exists a randomized Delayer strategy such that the probability of a game position to be a winning position for the Prover is very small. The Prover wins with probability 1, so there must be a lot of winning positions (i. e., lines in the refutation).

During the game the Prover record is represented as a matrix P with $n + 1$ rows and n columns, with values in $\{0, 1, *\}$. An entry $P(i, j) = 1$ means that Prover has recorded the information of pigeon i going into hole j, and likewise

$P(i,j) = 0$ means that pigeon i is not in hole j. For $P(i,j) = *$, Prover does not have any information. Thus the matrix P corresponds to a partial assignment to the variables $x_{i,j}$. The Prover wins whenever this assignment falsifies either an initial clause from PHP_n^{n+1} or a parameterized clause (i. e., there are at least k ones on the record), or if the Delayer surrenders.

We specify the following sets of "forbidden holes" for a pigeon i,

$$F(i) = \left\{ j \in [n] \;\middle|\; \begin{array}{lll} \text{either} & P(i,j) = 0 & \\ \text{or} & P(i',j) = 1 & \text{for some } i' \neq i \\ \text{or} & P(i,j') = 1 & \text{for some } j' \neq j \end{array} \right\}$$

and we say that pigeon i has $|F(i)|$ forbidden holes on record P (or just "on the record" if P is clear from the context).

The Delayer strategy. The Delayer chooses uniformly at random a partial matching α of $n - 2k$ pigeons to $n - 2k$ holes. In the game, if Prover asks about a pigeon from $dom(\alpha)$, then Delayer will always answer according to the matching α. We denote $\overline{dom}(\alpha) = [n+1] \setminus dom(\alpha)$ and $\overline{rng}(\alpha) = [n] \setminus rng(\alpha)$. These are the unassigned pigeons and holes for which Delayer has not made a decision before the start of the game. During the game Delayer only cares about the cells in P indexed by $\overline{dom}(\alpha) \times \overline{rng}(\alpha)$, because only here Prover can force her into a contradiction with an initial clause. We call P' the minor of P of size $(2k+1) \times (2k)$ on rows from $\overline{dom}(\alpha)$ and columns from $\overline{rng}(\alpha)$. The Delayer keeps also another private matrix D not known to the Prover, also of size $(2k+1) \times (2k)$. This matrix essentially encodes the "next answer" for any Prover question in $\overline{dom}(\alpha) \times \overline{rng}(\alpha)$.

The Delayer strategy is completely specified by how the Delayer answers to the questions and how she computes D at each step. The Delayer answers to the Prover question $x_{i,j}$ according to the following scheme:

1. For pigeons $i \in dom(\alpha)$, Delayer answers 1 if $\alpha(i) = j$ and 0 if $\alpha(i) \neq j$.

2. For pigeons $i \notin dom(\alpha)$ and holes $j \in rng(\alpha)$, Delayer answers 0.

3. For pigeons $i \notin dom(\alpha)$ and holes $j \notin rng(\alpha)$, Delayer answers $D(i,j)$.

The answers are both based on α and D. The matrix D is recomputed at each step, according to the values of P'. Let $G = \{\, i \in \overline{dom}(\alpha) : |F(i) \cap \overline{rng}(\alpha)| \geq k \,\}$, i. e., G contains all pigeons not assigned by α for which Prover has already excluded more than k of the remaining $2k$ holes left free by α. Because Delayer does not want to allow Prover to easily reach a contradiction on the pigeons from G (for instance, Prover could just continue to query holes for some pigeon $i \in G$), Delayer secretly assigns a hole to each pigeon in G. To achieve this, Delayer defines D as an arbitrary completion of P' (i. e., $*$'s are replaced by 0/1), such that D represents a partial matching which assigns exactly all pigeons from G to some set of holes from $\overline{rng}(\alpha)$. More precisely, D is *any* matrix in $\{0,1\}^{(2k+1)\times(2k)}$ which satisfies:

1. D is a completion of P', i. e., $D(i,j) = P'(i,j)$ for $P'(i,j) \neq *$.
2. Exactly the pigeons in G are matched, i. e.,
   (a) For any $i \in G$ there exists $j \in \overline{rng}(\alpha)$ such that $D(i,j) = 1$;
   (b) For any $i \in \overline{dom}(\alpha) \setminus G$ and any $j \in \overline{rng}(\alpha)$, $D(i,j) = 0$.
3. D is a matching, i. e.,
   (a) For any $i, i' \in \overline{dom}(\alpha)$, $i \neq i'$, and $j \in \overline{rng}(\alpha)$, either $D(i,j) = 0$ or $D(i',j) = 0$;
   (b) For any $i \in \overline{dom}(\alpha)$ and $j, j' \in \overline{rng}(\alpha)$, $j \neq j'$, either $D(i,j) = 0$ or $D(i,j') = 0$.

It is possible that such D does not exist: in this case *the Delayer surrenders.* Given the choice of α, the strategy of the Delayer is completely determined. Thus we will also call α the strategy of the Delayer.

Losing positions for the Delayer. The idea of the Delayer strategy is to play honestly on a large fraction of the variables, while trying to cheat only on the cells of P'. The Delayer always answers consistently with a partial matching, so the only way for Prover to expose her is to either find k assigned ones, or to force her to give up. We characterize the particular kinds of records where the game may end.

Claim 7.8.3 *If the Prover wins, at that time in the game either P contains at least k ones, or G contains at least k pigeons.*

Proof. We first see that Prover never wins by falsifying a clause from PHP^{n+1}_{n}. By definition of her strategy, Delayer always answers according to a partial matching which matches all pigeons from $dom(\alpha) \cup G$. Thus no conflict between pigeons arises. Also P can never falsify any of the big initial clauses $\bigvee_{j \in [n]} x_{i,j}$. This is clear for pigeons $i \in dom(\alpha)$ because α assigns some hole to i. But it also holds for pigeons $i \in \overline{dom}(\alpha)$ because after there are k forbidden holes for i from $\overline{rng}(\alpha)$, Delayer secretly reserves a hole for i in D. Thus, during the game there can never be more than $n - k$ forbidden holes for some pigeon without Delayer having already reserved some hole for it.

It remains to show that if $|G| \leq k$, then Delayer can always find a D as required by the strategy, thus she does not surrender. Consider the private matrix D_{old} computed by the Delayer at the previous step and denote G_{old} to be set of pigeons matched in D_{old}. If in the previous step of the game Prover only deleted some information from the record, then $G \subseteq G_{old}$ and thus Delayer can compute D. Assume now that Prover increased the information on the record and therefore $G \supseteq G_{old}$. The elements in $G \setminus G_{old}$ have exactly k forbidden holes because in each step of the game the number of forbidden holes per pigeon can increase at most by one, and they where below the threshold k at the previous step. Consider the matching induced by D_{old}. Each element of $G \setminus G_{old}$ still has k available holes, thus even excluding the holes occupied by pigeons from G_{old}, they still have at least $k - |G_{old}|$ available holes each. A simple greedy matching strategy will be sufficient to match all pigeons from $G \setminus G_{old}$ of which there are $\leq k - |G_{old}|$ many.

We conclude that the game ends either because P contains at least k ones, or because the Delayer surrenders, which means $|G| > k$. □

Intuition for the remaining part of the proof. The proof is a probabilistic argument. For a random α, the probability that a clause in the refutation corresponds to the final position in the interactive game is $\frac{1}{n^{\Omega(k)}}$. Any game ends somewhere, so there must be $n^{\Omega(k)}$ clauses.

The probability estimation is based on two observations: (a) if a game interaction produces a record, this record is compatible with the α chosen by the Delayer; (b) if Delayer gives up, it means that P' contains too much information for the Delayer to continue cheating.

The record analysis is divided in three cases: in cases (I) and (II) we argue that if a record contains a large amount of information, then there is little chance for a random α to be compatible with said record. In particular, case (I) corresponds to the Prover exposing k ones in the Delayer answers, and in case (II) Prover's record contains a huge amount of zeros. In both cases, Delayer succeeded in forcing Prover to write down a good part from the matching α.

For case (III) we argue about a record with less than k ones and small amount of information. Such record must force the Delayer to give up. This means that the small amount of information must be concentrated on P'. For random α and random relative position of P' with respect to P, this happens with small probability. Intuitively, in case (III) Prover is forced to "guess" at least $\overline{dom}(\alpha)$ or $\overline{rng}(\alpha)$ which he can achieve only with small probability.

Prover probability of winning. We now show that in each game position (i. e., in each line of the refutation), Prover has winning probability $\frac{1}{n^{\Omega(k)}}$ against Delayer's randomized strategy. This proves that any refutation contains $n^{\Omega(k)}$ lines.

Observe that as a necessary condition for a record to be a winning position for Prover, the record has to be compatible with the choice of α. This is because Delayer always answers according to α on pigeons from $dom(\alpha)$, and thus there is no way that information which is inconsistent with α can be written in Prover's record. We will use this fact in the following steps.

We will consider three cases for a record: (I) the record contains exactly k ones, (II) the record contains at least $n^{2/3} + 2k + 1$ pigeons which each have at least $kn^{1/3} \log n$ forbidden holes, (III) the record has $< k$ ones and contains at most $n^{2/3} + 2k$ pigeons which have more than $kn^{1/3} \log n$ forbidden holes each. Those three cases cover all possible records in the game. The result follows by proving that in each case, the probability that a record is a winning position for Prover is $\frac{1}{n^{\Omega(k)}}$.

Case I. Prover has exactly k ones on the record. On such records, the Prover always wins, but only for those game plays that actually lead to Prover's record. A necessary condition for this is that P is compatible with α. Let S be the set of pigeons with a 1 on the record. Then

$$\Pr_{\alpha}[\, P \text{ is winning against } \alpha \,] \;\leq\; \Pr_{\alpha}[\, P \text{ is compatible with } \alpha \,] \;\leq$$

$$\Pr_{\alpha}\left[|S \cap dom(\alpha)| < \frac{k}{2} \right] + \Pr_{\alpha}\left[P \text{ is compatible with } \alpha \;\middle|\; |S \cap dom(\alpha)| \geq \frac{k}{2} \right] \;.$$

The probability that $|S \cap dom(\alpha)| < \frac{k}{2}$ is equal to the probability that the uniform random set $\overline{dom}(\alpha) \subset [n+1]$ of size $2k+1$ intersects the set $S \subset [n+1]$ of size k in at least $k/2$ positions. Let $\overline{dom}(\alpha) = \{i_1, \ldots, i_{2k+1}\}$. Then

$$\Pr_{\alpha}\left[|S \cap \overline{dom}(\alpha)| \geq \frac{k}{2} \right] =$$

$$= \Pr_{i_1,\ldots,i_{2k+1}}\Big[\exists\, 1 \leq l_1 < \ldots < l_{k/2} \leq 2k+1 \text{ s.t. } \{i_{l_1}, \ldots, i_{l_{k/2}}\} \subseteq S \Big] \leq$$

$$\leq \sum_{1 \leq l_1 < \ldots < l_{k/2} \leq 2k+1} \Pr_{i_1,\ldots,i_{k/2}}\Big[\{i_{l_1}, \ldots, i_{l_{k/2}}\} \subseteq S \Big] \;\leq\; \binom{2k+1}{k/2} \cdot \frac{\binom{k}{k/2}}{\binom{n+1}{k/2}} \;\leq\; \frac{1}{n^{\Omega(k)}} \;.$$

The probability of α to be compatible with P, given that $|S \cap dom(\alpha)| \geq \frac{k}{2}$, is bounded by the probability of guessing the matching in P for the pigeons in $S \cap dom(\alpha)$, that is

$$\frac{1}{n} \cdot \frac{1}{n-1} \cdot \ldots \cdot \frac{1}{n - |S \cap dom(\alpha)| + 1} \;\leq\; \frac{1}{n^{\Omega(k)}} \;.$$

This completes Case I.

Case II. The Prover has a set S of at least $n^{2/3} + 2k + 1$ pigeons with $kn^{1/3} \log n$ forbidden holes each on the record P. At least $n^{2/3}$ of them are in $dom(\alpha)$. As in Case I the probability of strategy α to lose in position P is bounded by the probability of α being compatible with P. For compatibility it is necessary that a random α does not match any of the (at least) $n^{2/3}$ pigeons in $dom(\alpha)$ with any of their corresponding forbidden holes. Let us consider the matching holes for such pigeons as a randomly chosen sequence $h_1 \ldots h_{n^{2/3}}$. Assuming that $h_1 \ldots h_{l-1}$ are compatible choices there are at most $n - kn^{1/3} \log n - l + 1$ good choices for h_l over $n - l + 1$ possible choices. Thus h_l is compatible with probability at most

$$\frac{n - kn^{1/3} \log n - l + 1}{n - l + 1} \;=\; 1 - \frac{kn^{1/3} \log n}{n - l + 1} \;\leq\; 1 - \frac{k \log n}{n^{2/3}} \;.$$

The probability of the whole sequence to be compatible is then bounded by

$$\left(1 - \frac{k \log n}{n^{2/3}}\right)^{n^{2/3}} \leq \frac{1}{e^{\Omega(k \log n)}} \leq \frac{1}{n^{\Omega(k)}} \enspace .$$

Case III. Record P does not contain k ones, and the set S of pigeons with more than $kn^{1/3} \log n$ forbidden holes has size at most $n^{2/3} + 2k$. If Prover wins with record P against strategy α, then Claim 7.8.3 implies $|G| > k$. Thus the probability of strategy α to lose on record P is at most the probability that in $\overline{dom}(\alpha)$ there are k pigeons with at least k forbidden holes contained in $\overline{rng}(\alpha)$ each. We split the analysis into two further sub-cases.

Case III. (a) Assume first that $|S \cap \overline{dom}(\alpha)| \geq k$. Intuitively, in this case Prover has managed to place a good number of pigeons ($\geq k$) with many forbidden holes into $\overline{dom}(\alpha)$. For these pigeons from $S \cap \overline{dom}(\alpha)$, it will be easy for Prover to exclude $\geq k$ holes of $\overline{rng}(\alpha)$, and thus force these pigeons into G. But in total, S only contains few pigeons ($\leq n^{2/3} + 2k$), and this means that Prover has to "guess" $\overline{dom}(\alpha)$ which is hard for him.

For the formal analysis, let $i_1, \ldots, i_{2k+1}$ denote the elements of $\overline{dom}(\alpha)$. They form a uniformly chosen set of size $2k + 1$ in $[n + 1]$. The probability that they intersect S in k positions is at most

$$\Pr_{i_1,\ldots,i_{2k+1}} [\, |S \cap \{i_1, \ldots, i_{2k+1}\}| \geq k \,] \leq$$

$$\sum_{1 \leq l_1 < l_2 < \ldots < l_k \leq 2k+1} \Pr_{i_{l_1},\ldots,i_{l_k}} [\, \{i_{l_1}, \ldots, i_{l_k}\} \subseteq S \,] \leq \binom{2k+1}{k} \frac{\binom{n^{2/3}+2k}{k}}{\binom{n}{k}} \leq \frac{1}{n^{\Omega(k)}} \enspace .$$

Case III. (b) The other possibility is that $|S \cap \overline{dom}(\alpha)| < k$. But then for P to be winning against α, there must exist at least one pigeon $i \in \overline{dom}(\alpha) \setminus S$ such that $|F(i) \cap \overline{rng}(\alpha)| \geq k$. By union bound this is at most $2k + 1$ times the probability of such event for a fixed $i \in \overline{dom}(\alpha) \setminus S$. By $j_1, \ldots, j_{2k}$ we denote the elements of $\overline{rng}(\alpha)$, they form a uniformly chosen set of size $2k$ in $[n]$. Then

$$\Pr_{j_1,\ldots,j_{2k}} [\, |F(i) \cap \{j_1, \ldots, j_{2k}\}| \geq k \,] \leq$$

$$\sum_{1 \leq l_1 < l_2 < \ldots < l_k \leq 2k} \Pr_{j_{l_1},\ldots,j_{l_k}} [\, \{j_{l_1}, \ldots, j_{l_k}\} \subseteq F(i) \,] \leq \binom{2k}{k} \frac{\binom{kn^{1/3} \log n}{k}}{\binom{n}{k}} \leq \frac{1}{n^{\Omega(k)}} \enspace .$$

This completes the proof. □

7.8.4 An Alternative Probabilistic Proof

We now provide an alternative proof of the lower bound for the pigeonhole principle in Parameterized Resolution (Theorem 7.8.2). While this second proof is simpler than our first game-theoretic argument, it is less direct: it is essentially a reduction from the lower bound for Parameterized Resolution to the lower bound for the pigeonhole principle in classical Resolution, shown by Haken [Hak85].

Theorem 7.8.2 *Any refutation of (PHP_n^{n+1}, k) in Parameterized Resolution requires size $n^{\Theta(k)}$.*

Proof. We will only prove the lower bound. Consider the number of parameterized axioms explicitly used in the refutation. Without loss of generality we may assume that this number is $o(n^{k/5})$, otherwise the claim follows immediately.

Now choose uniformly at random a set of $n - \sqrt{n}$ pigeons and match them with a set of $n - \sqrt{n}$ uniformly chosen holes. Such partial matching f induces the following natural partial assignment of the variables of PHP_n^{n+1}:

$$\begin{aligned}
x_{i,j} &= 1 && \text{whenever } i \in dom(f) \text{ and } f(i) = j\\
x_{i,j} &= 0 && \text{whenever } i \in dom(f) \text{ and } f(i) \neq j\\
x_{i,j} &= 0 && \text{whenever } j \in rng(f) \text{ and there exist } i' \neq i \text{ such that } f(i') = j\\
x_{i,j} &= \star && \text{otherwise.}
\end{aligned}$$

We claim that with non-zero probability such partial assignment satisfies all parameterized axioms used in the refutation. Notice that we do not care if such assignment falsifies unused parameterized axioms. Before proving this claim, we show how the result follows from it.

The restricted refutation will not contain any parameterized axiom. Thus it is a classical Resolution refutation for the restricted formula, which in turn is equivalent (up to a re-indexing of pigeons and holes) to $PHP_{\sqrt{n}}^{\sqrt{n}+1}$. Such refutation must be of size at least 2^{n^c} [BP96] for some $c > 0$, thus asymptotically bigger than $n^{k/5}$. This concludes the proof.

The missing part is to show that the probabilistic choice of the partial matching realizes the desired properties with positive probability. Consider a parameterized axiom $\neg x_{i_1,j_1} \vee \ldots \vee \neg x_{i_{k+1},j_{k+1}}$. If there are two equal indexes j_a and j_b for $a \neq b$, then such axiom is just a weakening of a standard clause of the pigeonhole principle, so either the random matching assigns such hole j_a and the axiom is satisfied or no pigeon is matched with j_a. In the latter case the restricted axiom is deducible from the pigeonhole principle axiom $\neg x_{i_a,j_a} \vee \neg x_{i_b,j_a}$.

We can now focus on a parameterized axiom in which exactly $k+1$ holes are represented: the probability that such axiom fails to be satisfied is the probability that all x_{i_l,j_l} are either true or unassigned for $1 \leq l \leq k+1$. The probability that a pigeon i_l is unassigned is at most

$$\frac{\sqrt{n}}{n - \sqrt{n} - k - 1} \leq (1 - o(1))\frac{1}{n^{1/2}}$$

whatever is the outcome for the other pigeons in the axiom. Thus the probability that more than $k/2$ pigeon indexes in an axiom correspond to unassigned pigeons is at most the probability that there exists a sets of $\frac{k}{2}$ pigeon indexes among the $k+1$ in the axiom which are collectively unassigned. This is at most $(1-o(1))\frac{\binom{k}{k/2}}{n^{k/4}}$.

For axioms with more than $k/2$ assigned indexes the probability of being not satisfied is zero if two assigned indexes correspond to the same pigeon, thus we may focus on $k/2$ different pigeon indexes: whatever happens to the other pigeon indexes, any index i_l has probability at most $\frac{1}{n-\sqrt{n}-k/2} \leq (1-o(1))\frac{1}{n}$ to be assigned to j_l. Thus the failure of being falsified is at most $(1-o(1))\frac{1}{n^{k/2}}$.

An axiom is not satisfied with probability at most $(1-o(1))\frac{\binom{k}{k/2}}{n^{k/4}} = o(n^{k/5})$, thus by counting there exists a partial matching which satisfies all the parameterized axioms used in the refutation. This concludes the proof. □

7.9 On the Automatizability of (Parameterized) Resolution

Practitioners are not only interested in the size of a proof, but face the more complicated problem to actually construct a proof for a given instance. Of course, in the presence of super-polynomial lower bounds to the proof size this cannot be done in polynomial time. Thus, in proof search the best one can hope for is the following notion of automatizability:

Definition 7.9.1 (Bonet, Pitassi, Raz [BPR00]) *A proof system P for a language L is* automatizable *if there exists a deterministic procedure that takes as input a string x and outputs a P-proof of x in time polynomial in the size of the shortest P-proof of x if $x \in L$. If $x \notin L$, then the behavior of the algorithm is unspecified.*

For practical purposes automatizable systems would be very desirable. Searching for a proof we may not find the shortest one, but we are guaranteed to find one that is only polynomially longer. Unfortunately, most proof systems appear to be not automatizable. For strong proof systems as Frege systems this holds under cryptographic assumptions [KP98, BPR00]. But even weak systems such as Resolution and Polynomial Calculus are not automatizable under plausible assumptions from parameterized complexity [AR08, GL10].

These results yield the first connection between classical proof complexity and parameterized complexity. Alekhnovich and Razborov [AR08] proved that if (classical) Resolution was automatizable, then $\mathsf{W[P]}$ coincides with FPR, the randomized version of FPT. Since separating FPT from $\mathsf{W[2]}$ is a similar but probably harder problem than proving $\mathsf{W[P]} \neq \mathsf{FPR}$, proving lower bounds to successively stronger parameterized proof systems on one side, and strengthening the result of Alekhnovich and Razborov on the other seems, in our view, a promising approach towards *unconditional non-automatizability* of (classical) Resolution and other proof systems, which is an important problem in proof complexity.

The concept of automatizability can be easily extended to parameterized proof systems and it appears to be an interesting problem to investigate. However, there are some differences with respect to (classical) Resolution. Namely, a quasi-polynomial approximation of the shortest proof is meaningless in the context of Parameterized Resolution, because every $(F, k) \in \mathrm{PCon}$ with $|F| = n$ has a refutation of size $c \cdot \binom{n}{k+1}$ for some constant c. If $k \leq \log n$ this is smaller than $n^{\log n}$; otherwise $\binom{n}{k+1} \leq 2^{(k+1)^2}$ which is fpt with respect to k. Hence for any $(F, k) \in \mathrm{PCon}$ there exists a refutation of size $f(k)q(n)$ where f is some computable function and q is some quasi-polynomial function. We are in the situation discussed in [AR08], where we want to discriminate between polynomial and quasi-polynomial efficiency. Another difficulty is that known non-automatizability results in [BKPS02, ABMP01, AR08] all use formulas which are interesting and meaningful on assignments of big weight.

Chapter 8

The Broader Picture

> Schon jetzt erklären die Meister der Naturwissenschaften die Notwendigkeit monographischer Behandlung und also das Interesse an Einzelheiten. Dies ist aber nicht denkbar ohne eine Methode, die das Interesse an der Gesamtheit offenbart. Hat man das erlangt, so braucht man freilich nicht in Millionen Einzelheiten umherzutasten.
>
> Johann Wolfgang Goethe

In this thesis we focused on two non-classical models for proof systems: computation with non-uniform information and parameterized proof systems. However, modern computational complexity employs and successfully studies other computational models such as randomization, quantum computing, oracle access, or space complexity. In proof complexity these considerations were started recently by several researchers.[1]

Proof systems with oracle access and their relation to proof systems with advice were already briefly mentioned in Section 4.3.2. To place our results into a broader perspective, we give below a small (and incomplete) account on recent research on proof systems with randomization and quantum computation, and on space complexity for proof systems. We conclude in Section 8.2 by outlining some further directions for research on non-classical proof complexity.

[1] This increased interest in non-classical aspects in proof complexity was very visible in a special session on proof complexity organized by Jan Krajíček at the conference TAMC 2010 in Prague where all the five speakers of the session talked on non-classical aspects in proof complexity: Stefan Dantchev on a new model for randomized proof systems, Edward Hirsch on heuristic acceptors and optimal proof systems [Hir10], Iddo Tzameret on algebraic proof systems [Tza10], Sebastian Müller on proof systems with advice [BM10a], and myself on proof complexity of non-classical logics [Bey10].

8.1 Other Models for Proof Systems

8.1.1 Probabilistic Proof Systems

Usually, the term "probabilistic proofs" is associated with interactive proof systems like IP or Babai's Arthur-Merlin classes MA and AM. Besides from randomization, the power of these proof systems stems from using interaction between a powerful prover and a polynomial-time verifier.

A non-interactive model of randomized proofs was very recently introduced by Hirsch and Itsykson [HI10]. They define two concepts: heuristic acceptors and heuristic proof systems. Acceptors are not really proof systems, but algorithms which accept all elements from the language and do not stop on other inputs. There is, however, a close relationship between acceptors and proof systems (cf. [KP89]). There is a nice recent survey on optimal acceptors and optimal proof systems by Hirsch [Hir10].

For the randomized approach, we have to consider a probability distribution. A distribution D is concentrated on some set A, if $\mu_D(A) = 1$.

Definition 8.1.1 (Hirsch, Itsykson [HI10]) *A pair* (D, L) *is a* distributional proving problem *if* D *is a family of probability distributions* D_n *concentrated on* $\overline{L} \cap \{0,1\}^n$.

Hirsch and Itsykson define a *heuristic acceptor* for a distributional proving problem (D, L) as a randomized algorithm which always accepts inputs from L and accepts inputs from $\bar{L}$ only with small probability (see [Hir10] for the exact definition). For this model they show an optimality result:

Theorem 8.1.2 (Hirsch, Itsykson [HI10]) *Let* L *be recursively enumerable and* D *be a polynomial-time samplable distribution. Then there exists an optimal automatizer for* (D, L).

The authors also consider heuristic proof systems and show interesting results on these systems with respect to automatizability, i. e., the problem to construct proofs for given formulas (see [Hir10]).

8.1.2 Quantum Proof Systems

Quantum computations are a new computational paradigm which links computer science and physics in a historically unparalleled way. The physicist Richard Feynman [Fey82] discovered in 1982 that classical computers cannot efficiently simulate quantum-mechanic systems. Building on this observation, he was the first to consider the construction of computers based on quantum-mechanic principles. This motivation is further strengthened by the belief that ongoing miniaturization of electronic circuits will inevitably bring us into the realm of quantum effects as already foreseen by Keyes in 1988 [Key88].

In 1985 Deutsch [Deu85] introduced a theoretical model for quantum computers and demonstrated that there are problems which can be solved more efficiently on quantum machines than on classical computers. Undoubtedly the most important result in the area of quantum computing was established by Shor [Sho97] who designed an efficient quantum algorithm for factoring natural numbers. This result is very important as the security of almost all modern public-key cryptosystems as RSA is based on the intractability of factoring.

A further remarkable result was shown by Grover [Gro96] who constructed a quantum algorithm solving the search problem in an unstructured database with n elements in $\sqrt{n}$ steps. Classical procedures need at least n steps to search the database, and even in the randomized model the expected time is still $\frac{n}{2}$. Such search problems are typical for NP complete problems such as propositional satisfiability.

This speedup in search raises hopes that we can reduce lengths of proofs when proving theorems. This was the motivation for Pudlák's recent introduction of *quantum proof systems* [Pud09]. Pudlák first introduces a general model of quantum proof systems and then focuses on quantum Frege systems. Let us start with the general concept.

Definition 8.1.3 (Pudlák [Pud09]) *A* quantum proof system *consists of a set $A \subseteq \Sigma^*$ (the set of valid proofs) and a family of circuits C_n (the proof system) such that*

1. *A is decidable in polynomial time and C_n is* P*-uniform* (Efficiency)*;*
2. *for any proof $\pi \in A$, $C_{|\pi|}(\pi)$ produces a superposition of strings of tautologies* (Correctness)*;*
3. *for every tautology φ there exists $\pi \in A$ such that φ occurs in the superposition of $C_{|\pi|}(\pi)$* (Completeness)*.*

Regarding the completeness condition, it is also important that by measuring $C_{|\pi|}(\pi)$ we can obtain φ with a probability which is not too small. Hence quantum proof systems also have probabilistic aspects.

The next concept which Pudlák introduces are *quantum rules* which are based on unitary transformations. Using a finite set of quantum rules, Pudlák arrives at the notion of *quantum Frege systems.* Comparing quantum Frege with classical Frege systems, Pudlák obtains the surprising result that quantum Frege systems do not have shorter proofs, i. e., every quantum Frege system is simulated by a classical Frege system. On the other hand, it does not seem possible to extract classical proofs from quantum Frege proofs, i. e., under cryptographic assumptions quantum Frege systems are not p-simulated by classical Frege systems.

8.1.3 Space in Proof Complexity

Besides running time—which corresponds to lengths of proofs—one further important measure is the space consumption of algorithms. Space complexity for proof systems was intensively investigated in the context of Resolution [ET01, ABSRW02, BSN08]. Here the minimal space to refute a set of clauses is of particular interest as it corresponds to the memory consumption of modern SAT solvers which often combine DPLL algorithms with clause learning. Therefore, both lower bounds for Resolution space [ABSRW02, BSG03, EGM04, ET03] as well as optimal trade-offs between space and length, i. e., between memory and run-time consumption, have been intensively studied [Nor06, NH08, BSN08, BSN09].

8.2 Future Perspectives

Analogously to what has been done in complexity theory, we believe that research on alternative resources in proof complexity will also help to understand limitations in theorem proving in the classical model. In particular, in understanding how these new computational models might advance theorem proving and what are their limits, the following main general questions seem interesting. Their aim is to generalize to proof complexity the approach of non-classical models already well studied in complexity theory:

1. Are theorems significantly easier to prove when proofs are verified by alternative resources?

2. How are these computational models related with respect to their strength in proving theorems?

3. What is the tradeoff between the computational strength of the resources and the efficiency in proving theorems?

4. Can theorem proving be automated and under which resources?

In general, we have to admit that our understanding of non-classical resources in proof complexity is still at a very initial stage. We believe, however, that further research into non-classical measures of proofs will both strengthen the connections between computational and proof complexity and lead to new insights for classical proof systems.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[ABMP01] Michael Alekhnovich, Samuel R. Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is NP-hard to linearly approximate. *The Journal of Symbolic Logic*, 66(1):171–191, 2001.

[ABSRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002.

[ABSRW04] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM Journal on Computing*, 34(1):67–88, 2004.

[AD08] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, 2008.

[Ajt94] Miklós Ajtai. The complexity of the pigeonhole-principle. *Combinatorica*, 14(4):417–433, 1994.

[AKMT00] V. Arvind, Johannes Köbler, Martin Mundhenk, and Jacobo Torán. Nondeterministic instance complexity and hard-to-prove tautologies. In *Proc. 17th Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 314–323. Springer-Verlag, Berlin Heidelberg, 2000.

[AR08] Michael Alekhnovich and Alexander A. Razborov. Resolution is not automatizable unless W[P] is tractable. *SIAM Journal on Computing*, 38(4):1347–1363, 2008.

[BBP95] Maria Luisa Bonet, Samuel R. Buss, and Toniann Pitassi. Are there hard examples for Frege systems? In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 30–56. Birkhäuser, 1995.

[BCF03] Harry Buhrman, Richard Chang, and Lance Fortnow. One bit of advice. In *Proc. 20th Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 547–558. Springer-Verlag, Berlin Heidelberg, 2003.

[BCMM05] Paul Beame, Joseph C. Culberson, David G. Mitchell, and Cristopher Moore. The resolution complexity of random graph k-colorability. *Discrete Applied Mathematics*, 153(1-3):25–47, 2005.

[BDG88] José L. Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I.* Springer-Verlag, Berlin Heidelberg, 1988.

[Bei91] Richard Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Computer Science*, 84:199–223, 1991.

[Bey07] Olaf Beyersdorff. Classes of representable disjoint NP-pairs. *Theoretical Computer Science*, 377(1–3):93–109, 2007.

[Bey09] Olaf Beyersdorff. On the correspondence between arithmetic theories and propositional proof systems – a survey. *Mathematical Logic Quarterly*, 55(2):116–137, 2009.

[Bey10] Olaf Beyersdorff. Proof complexity of non-classical logics. In *Proc. 7th Conference on Theory and Applications of Models of Computation*, volume 6108 of *Lecture Notes in Computer Science*, pages 15–27. Springer-Verlag, Berlin Heidelberg, 2010.

[BFL01] Harry Buhrman, Lance Fortnow, and Sophie Laplante. Resource-bounded Kolmogorov complexity revisited. *SIAM Journal on Computing*, 31(3):887–905, 2001.

[BG01] Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, 2001.

[BGL10] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A lower bound for the pigeonhole principle in tree-like resolution by asymmetric prover-delayer games. *Information Processing Letters*, 110(23):1074–1077, 2010.

[BGL11] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. Parameterized complexity of DPLL search procedures. In *Proc. 14th International Conference on Theory and Applications of Satisfiability Testing*, volume 6695 of *Lecture Notes in Computer Science*, pages 5–18. Springer-Verlag, Berlin Heidelberg, 2011.

[BGLR11] Olaf Beyersdorff, Nicola Galesi, Massimo Lauria, and Alexander Razborov. Parameterized bounded-depth Frege is not optimal. In *Proc. 38th International Colloquium on Automata, Languages, and Programming*, volume 6755 of *Lecture Notes in Computer Science*, pages 630–641. Springer-Verlag, Berlin Heidelberg, 2011.

[BH08] Harry Buhrman and John M. Hitchcock. NP-hard sets are exponentially dense unless coNP $\subseteq$ NP/poly. In *Proc. 23rd Annual IEEE Conference on Computational Complexity*, pages 1–7, 2008.

[BIK$^+$92] Paul W. Beame, Russel Impagliazzo, Jan Krajíček, Toniann Pitassi, Pavel Pudlák, and Alan Woods. Exponential lower bounds for the pigeonhole principle. In *Proc. 24th ACM Symposium on Theory of Computing*, pages 200–220, 1992.

[BIK$^+$96] Paul W. Beame, Russel Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. *Proc. London Mathematical Society*, 73(3):1–26, 1996.

[BKM09] Olaf Beyersdorff, Johannes Köbler, and Jochen Messner. Nondeterministic functions and the existence of optimal proof systems. *Theoretical Computer Science*, 410(38–40):3839–3855, 2009.

[BKM11] Olaf Beyersdorff, Johannes Köbler, and Sebastian Müller. Proof systems that take advice. *Information and Computation*, 209(3):320–332, 2011.

[BKPS02] Paul Beame, Richard M. Karp, Toniann Pitassi, and Michael E. Saks. The efficiency of resolution and Davis–Putnam procedures. *SIAM Journal on Computing*, 31(4):1048–1075, 2002.

[BM10a] Olaf Beyersdorff and Sebastian Müller. Different approaches to proof systems. In *Proc. 7th Conference on Theory and Applications of Models of Computation*, volume 6108 of *Lecture Notes in Computer Science*, pages 50–59. Springer-Verlag, Berlin Heidelberg, 2010.

[BM10b] Olaf Beyersdorff and Sebastian Müller. A tight Karp-Lipton collapse result in bounded arithmetic. *ACM Transactions on Computational Logic*, 11(4), 2010.

[Boo74] Ronald V. Book. Tally languages and complexity classes. *Information and Control*, 26:186–193, 1974.

[BP96] Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *Proc. 37th IEEE Symposium on the Foundations of Computer Science*, pages 274–282, 1996.

[BPI93] Paul Beame, Toniann Pitassi, and Russel Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3(2):97–140, 1993.

[BPR97] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. Lower bounds for cutting planes proofs with small coefficients. *The Journal of Symbolic Logic*, 62(3):708–728, 1997.

[BPR00] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for Frege systems. *SIAM Journal on Computing*, 29(6):1939–1967, 2000.

[BS92] José L. Balcázar and Uwe Schöning. Logarithmic advice classes. *Theoretical Computer Science*, 99:279–290, 1992.

[BS11] Olaf Beyersdorff and Zenon Sadowski. Do there exist complete sets for promise classes? *Mathematical Logic Quarterly*, 57(6):535–550, 2011.

[BSG03] Eli Ben-Sasson and Nicola Galesi. Space complexity of random formulae in resolution. *Random Structures and Algorithms*, 23(1):92–109, 2003.

[BSIW04] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004.

[BSN08] Eli Ben-Sasson and Jakob Nordström. Short proofs may be spacious: An optimal separation of space and length in resolution. In *Proc. 49th IEEE Symposium on the Foundations of Computer Science*, pages 709–718, 2008.

[BSN09] Eli Ben-Sasson and Jakob Nordström. Understanding space in resolution: Optimal lower bounds and exponential trade-offs. Technical Report TR09-034, Electronic Colloquium on Computational Complexity, 2009.

[BSW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *Journal of the ACM*, 48(2):149–169, 2001.

[Bus86] Samuel R. Buss. *Bounded Arithmetic.* Bibliopolis, Napoli, 1986.

[Bus98] Samuel R. Buss. An introduction to proof theory. In Samuel R. Buss, editor, *Handbook of Proof Theory*, pages 1–78. Elsevier, Amsterdam, 1998.

[Cai07] Jin-Yi Cai. $S_2^p \subseteq ZPP^{NP}$. *Journal of Computer and System Sciences*, 73(1):25–35, 2007.

[CCHO05] Jin-Yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Information and Computation*, 198(1):1–23, 2005.

[CEI96] Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proc. 28th ACM Symposium on Theory of Computing*, pages 174–183, 1996.

[CF08] Yijia Chen and Jörg Flum. The parameterized complexity of maximality and minimality problems. *Annals of Pure and Applied Logic*, 151(1):22–61, 2008.

[CF10] Yijia Chen and Jörg Flum. On optimal proof systems and logics for PTIME. In *Proc. 37th International Colloquium on Automata, Languages, and Programming*, 2010.

[CK96] Richard Chang and Jim Kadin. The Boolean hierarchy and the polynomial hierarchy: A closer connection. *SIAM Journal on Computing*, 25(2):340–354, 1996.

[CK07] Stephen A. Cook and Jan Krajíček. Consequences of the provability of NP $\subseteq$ P/poly. *The Journal of Symbolic Logic*, 72(4):1353–1371, 2007.

[CN10] Stephen A. Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity.* Cambridge University Press, 2010.

[Coo75] Stephen A. Cook. Feasibly constructive proofs and the propositional calculus. In *Proc. 7th Annual ACM Symposium on Theory of Computing*, pages 83–97, 1975.

[Coo05] Stephen A. Cook. Theories for complexity classes and their propositional translations. In Jan Krajíček, editor, *Complexity of Computations and Proofs*, pages 175–227. Quaderni di Matematica, 2005.

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

[Deu85] David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. of the Royal Society*, 400:97–117, 1985.

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Springer-Verlag, Berlin Heidelberg, 1999.

[DMS07] Stefan S. Dantchev, Barnaby Martin, and Stefan Szeider. Parameterized proof complexity. In *Proc. 48th IEEE Symposium on the Foundations of Computer Science*, pages 150–160, 2007.

[DR01] Stefan S. Dantchev and Søren Riis. Tree resolution proofs of the weak pigeon-hole principle. In *Proc. 16th Annual IEEE Conference on Computational Complexity*, pages 69–75, 2001.

[EGM04] Juan Luis Esteban, Nicola Galesi, and Jochen Messner. On the complexity of resolution with bounded conjunctions. *Theoretical Computer Science*, 321(2–3):347–370, 2004.

[ET01] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001.

[ET03] Juan Luis Esteban and Jacobo Torán. A combinatorial characterization of treelike resolution space. *Information Processing Letters*, 87(6):295–300, 2003.

[Fey82] Richard Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[FG03] Jörg Flum and Martin Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Springer-Verlag, Berlin Heidelberg, 2006.

[FK05] Lance Fortnow and Adam R. Klivans. NP with small advice. In *Proc. 20th Annual IEEE Conference on Computational Complexity*, pages 228–234, 2005.

[Gao09] Yong Gao. Data reductions, fixed parameter tractability, and random weighted d-CNF satisfiability. *Artificial Intelligence*, 173(14):1343–1366, 2009.

[GL10] Nicola Galesi and Massimo Lauria. On the automatizability of polynomial calculus. *Theory of Computing Systems*, 47(2):491–506, 2010.

[Göd93] Kurt Gödel. Ein Brief an Johann von Neumann, 20. März, 1956. In P. Clote and J. Krajíček, editors, *Arithmetic, Proof Theory, and Computational Complexity*, pages 7–9. Oxford University Press, 1993.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28th ACM Symposium on Theory of Computing*, pages 212–219, 1996.

[GSS05] Christian Glaßer, Alan L. Selman, and Samik Sengupta. Reductions between disjoint NP-pairs. *Information and Computation*, 200(2):247–267, 2005.

[GSSZ04] Christian Glaßer, Alan L. Selman, Samik Sengupta, and Liyu Zhang. Disjoint NP-pairs. *SIAM Journal on Computing*, 33(6):1369–1416, 2004.

[GSZ06] Christian Glaßer, Alan L. Selman, and Liyu Zhang. Survey of disjoint NP-pairs and relations to propositional proof systems. In Oded Goldreich, Arnold L. Rosenberg, and Alan L. Selman, editors, *Essays in Theoretical Computer Science in Memory of Shimon Even*, pages 241–253. Springer-Verlag, Berlin Heidelberg, 2006.

[GSZ07] Christian Glaßer, Alan L. Selman, and Liyu Zhang. Canonical disjoint NP-pairs of propositional proof systems. *Theoretical Computer Science*, 370(1–3):60–73, 2007.

[Hak85] Amin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[HH88] Juris Hartmanis and Lane A. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58:129–142, 1988.

[HI10] Edward A. Hirsch and Dmitry Itsykson. On optimal heuristic randomized semidecision procedures, with application to proof complexity. In *Proc. 27th Symposium on Theoretical Aspects of Computer Science*, pages 453–464, 2010.

[Hir10] Edward A. Hirsch. Optimal acceptors and optimal proof systems. In *Proc. 7th Conference on Theory and Applications of Models of Computation*. Springer-Verlag, Berlin Heidelberg, 2010.

[IM99] Kazuo Iwama and Shuichi Miyazaki. Tree-like resolution is superpolynomially slower than dag-like resolution for the pigeonhole principle. In *Proc. 10th International Symposium on Algorithms and Computation*, volume 1741 of *Lecture Notes in Computer Science*, pages 133–142. Springer-Verlag, Berlin Heidelberg, 1999.

[Its09] Dmitry Itsykson. Structural complexity of AvgBPP. In *Proc. 4th International Computer Science Symposium in Russia*, volume 5675 of

Lecture Notes in Computer Science, pages 155–166. Springer-Verlag, Berlin Heidelberg, 2009.

[Jeř09] Emil Jeřábek. Approximate counting by hashing in bounded arithmetic. *Journal of Symbolic Logic*, 74(3):829–860, 2009.

[Kad88] Jim Kadin. The polynomial time hierarchy collapses if the Boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, 1988.

[Kad89] Jim Kadin. $\mathrm{P}^{\mathrm{NP}[\log n]}$ and sparse Turing-complete sets for NP. *Journal of Computer and System Sciences*, 39:282–298, 1989.

[Key88] R. W. Keyes. Miniaturization of electronics and its limits. *IBM Journal of Research and Development*, 32:24–28, 1988.

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. 12th ACM Symposium on Theory of Computing*, pages 302–309. ACM Press, 1980.

[KMT03] Johannes Köbler, Jochen Messner, and Jacobo Torán. Optimal proof systems imply complete sets for promise classes. *Information and Computation*, 184(1):71–92, 2003.

[Kow84] Wojciech Kowalczyk. Some connections between representability of complexity classes and the power of formal systems of reasoning. In *Proc. 11th Symposium on Mathematical Foundations of Computer Science*, volume 176 of *Lecture Notes in Computer Science*, pages 364–369. Springer-Verlag, Berlin Heidelberg, 1984.

[KP89] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54(3):1063–1079, 1989.

[KP90] Jan Krajíček and Pavel Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 36:29–46, 1990.

[KP98] Jan Krajíček and Pavel Pudlák. Some consequences of cryptographical conjectures for S^1_2 and EF. *Information and Computation*, 140(1):82–94, 1998.

[KPT91] Jan Krajíček, Pavel Pudlák, and Gaisi Takeuti. Bounded arithmetic and the polynomial hierarchy. *Annals of Pure and Applied Logic*, 52:143–153, 1991.

[KPW95] Jan Krajíček, Pavel Pudlák, and Alan 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.

[Kra95] Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 1995.

[Kra97] Jan Krajíček. Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997.

[Kra01] Jan Krajíček. Tautologies from pseudo-random generators. *Bulletin of Symbolic Logic*, 7(2):197–212, 2001.

[Kra04a] Jan Krajíček. Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds. *The Journal of Symbolic Logic*, 69(1):265–286, 2004.

[Kra04b] Jan Krajíček. Implicit proofs. *The Journal of Symbolic Logic*, 69(2):387–397, 2004.

[Kra07] Jan Krajíček. A proof complexity generator. In *Proc. 13th International Congress of Logic, Methodology and Philosophy of Science*, Studies in Logic and the Foundations of Mathematics. King's College Publications, London, 2007.

[KS85] Ker-I Ko and Uwe Schöning. On circuit-size complexity and the low hierarchy in NP. *SIAM Journal on Computing*, 14:41–51, 1985.

[KST93] Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1993.

[KW98] Johannes Köbler and Osamu Watanabe. New collapse consequences of NP having small circuits. *SIAM Journal on Computing*, 28(1):311–324, 1998.

[NH08] Jakob Nordström and Johan Håstad. Towards an optimal separation of space and length in resolution. In *Proc. 40th ACM Symposium on Theory of Computing*, pages 701–710, 2008.

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[Nor06] Jakob Nordström. Narrow proofs may be spacious: separating space and width in resolution. In *Proc. 38th ACM Symposium on Theory of Computing*, pages 507–516, 2006.

[OKSW94] Pekka Orponen, Ker-I Ko, Uwe Schöning, and Osamu Watanabe. Instance complexity. *Journal of the ACM*, 41(1):96–121, 1994.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[PB94] Pavel Pudlák and Samuel R. Buss. How to lie without being (easily) convicted and the length of proofs in propositional calculus. In *Proc. 8th Workshop on Computer Science Logic*, volume 933 of *Lecture Notes in Computer Science*, pages 151–162. Springer-Verlag, Berlin Heidelberg, 1994.

[PI00] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for SAT. In *Proc. 11th Symposium on Discrete Algorithms*, pages 128–136, 2000.

[Pip79] Nicholas Pippenger. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, 1979.

[PS10] Toniann Pitassi and Rahul Santhanam. Effectively polynomial simulations. In *Proc. 1st Innovations in Computer Science*, 2010.

[Pud91] Pavel Pudlák. Ramsey's theorem in bounded arithmetic. In E. Börger et al., editors, *Computer Science Logic '90*, pages 308–312. Springer, Berlin, 1991.

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997.

[Pud98] Pavel Pudlák. The lengths of proofs. In Samuel R. Buss, editor, *Handbook of Proof Theory*, pages 547–637. Elsevier, Amsterdam, 1998.

[Pud99] Pavel Pudlák. On the complexity of propositional calculus. In *Sets and Proofs, Invited papers from Logic Colloquium'97*, pages 197–218. Cambridge University Press, 1999.

[Pud00] Pavel Pudlák. Proofs as games. *American Math. Monthly*, pages 541–550, 2000.

[Pud09] Pavel Pudlák. Quantum deduction rules. *Annals of Pure and Applied Logic*, 157(1):16–29, 2009.

[PW85] Jeff Paris and Alec J. Wilkie. Counting problems in bounded arithmetic. In *Methods in Mathematical Logic, Proc. 6th Latin American Symposium*, pages 317–340, 1985.

[Raz98] Alexander A. Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, 1998.

[Rii01] Søren Riis. A complexity gap for tree resolution. *Computational Complexity*, 10(3):179–209, 2001.

[Sch83] Uwe Schöning. A low and a high hierarchy within NP. *Journal of Computer and System Sciences*, 27:14–28, 1983.

[Seg07] Nathan Segerlind. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 13(4):417–481, 2007.

[Sel94] Alan L. Selman. A taxonomy of complexity classes of functions. *Journal of Computer and System Sciences*, 48(2):357–381, 1994.

[Sho97] Peter Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[Stå96] Gunnar Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33(3):277–280, 1996.

[Tse68] G. C. Tseitin. On the complexity of derivations in propositional calculus. In A. O. Slisenko, editor, *Studies in Mathematics and Mathematical Logic, Part II*, pages 115–125. 1968.

[Tur36] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.*, 42:230–265, 1936.

[Tza10] Iddo Tzameret. Algebraic proofs over noncommutative formulas. In *Proc. 7th Conference on Theory and Applications of Models of Computation*, volume 6108 of *Lecture Notes in Computer Science*, pages 60–71. Springer-Verlag, Berlin Heidelberg, 2010.

[Zam96] Domenico Zambella. Notes on polynomially bounded arithmetic. *The Journal of Symbolic Logic*, 61(3):942–966, 1996.

Index