Pebble Games and Complexity

Siu Man Chan

August 14, 2013

Pebble Games and Complexity

by

Siu Man Chan

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Spring 2013

Pebble Games and Complexity

Abstract

Pebble Games and Complexity

by

Siu Man Chan

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Elchanan Mossel, Co-chair

Professor Luca Trevisan, Co-chair

We study the connection between pebble games and complexity.

First, we derive complexity results using pebble games. It is shown that three pebble games used for studying computational complexity are equivalent: namely, the two-person pebble game of Dymond–Tompa, the two-person pebble game of Raz–McKenzie, and the one-person reversible pebble game of Bennett have the same pebble costs over any directed acyclic graph. The three pebble games have been used for studying parallel complexity and for proving lower bounds under restricted settings, and we show one more such lower bound on circuit-depth.

Second, the pebble costs are applied to proof complexity. Concerning a family of unsatisfiable CNFs called pebbling contradictions, the pebble cost in any of the pebble games controls the scaling of some parameters of resolution refutations. Namely, the pebble cost controls the minimum depth of resolution refutations and the minimum size of tree-like resolution refutations.

Finally, we study the space complexity of computing the pebble costs and of computing the minimum depth of resolution refutations. It is PSPACE-complete to compute the pebble cost in any of the three pebble games, and to compute the minimum depth of resolution refutations.

Contents

Acknowledgments

I thank my advisors Luca Trevisan and Elchanan Mossel for letting me pursue the research that excites me, and for giving me a push when I need help with my career. They provided me encouragement, guidance, and optimism when I faced some of the seemingly impossible tasks, allowing me to eventually settle them. More importantly, they allowed me to spend enough time at first to try out different approaches that do not work, which is necessary for getting the right idea to work at last.

I am grateful for UC Berkeley for offering me a warm atmosphere and a wonderful environment to carry out my research. Many people made my time in Berkeley a lot more fun, and steered my research and my life greatly. Special thanks to my office buddies Anand Bhaskar and Thomas Watson, my other academic siblings James Cook and Anindya De, TGIF organizer Urmila Mahadev, language technician Piyush Srivastava, and other fellow Theory students from Berkeley: they include, but are not limited to, Jake Abernethy, Nima Anari, Antonio Blanca, Brielin Brown, Paul Christiano, Jonah Brown-Cohen, Kevin Dick, Milos Drezgic, Omid Etesami, Rafael Frongillo, Sakshi Jain, Alexandra Kolla, Henry Lin, Lorenzo Orecchia, George Pierrakos, Anupam Prakash, Aviad Rubinstein, Grant Schoenebeck, Tselil Schramm, Jarrett Schwartz, Seung Woo Shin, Meromit Singer, Yaron Singer, Isabelle Stanton, Alexandre Stauer, Ning Tan, Madhur Tulsiani, Gregory Valiant, Thomas Vidick, Di Wang, and Chris Wilkens. All my fellow students, together with other visiting students Andrew Drucker, Rishi Gupta, and Mohammad Moharrami, remind me of the many awesome retreats and laughters. For other senior people with whom I interacted in Berkeley, I thank Ilias Diakonikolas, Mihai Pătraşcu, Martin Suchara, Paul Valiant, and Virginia Vassilevska Williams for their various inspiring discussions and advices. I also thank other Faculty of Berkeley for their support and encouragement, and for the attitudes and passions I learned from them on teaching, research, life and everything; thanks to Richard Karp, Christos Papadimitriou, Prasad Raghavendra, Satish Rao, Alistair Sinclair, Yun Song, and Umesh Vazirani. All of the people I met in Berkeley shaped my way for the many more years to come.

I thank other people I have worked with for their encouragement and for their work on related research projects leading to this thesis. They include Stephen Cook, Yuval Filmus, Pierre McKenzie, Toniann Pitassi, Aaron Potechin, Robert Robere, and Dustin Wehr. It is clear that their ideas and thoughts have influenced my research a lot, both directly or indirectly.

Finally, I thank my family for their encouragement, patience, support and understanding throughout my study. I also thank my twin brother Siu On who is only implicitly acknowledged throughout by default.

Chapter 1

Introduction

Memory space and parallel time are two important resources of deterministic computation. To study these two resources, researchers considered different approaches. In this thesis, we focus on the approach of analyzing pebble games and the approach of analyzing concrete combinatorial models of computation.

It turns out that there is an unobserved connection between the two approaches. Namely, many of the combinatorial approaches for studying L versus NL and NC versus P under different restricted settings implicitly proved a lower bound scaling in the same way as the pebbling algorithms. This motivates us to study the interaction between pebble games and complexity: namely, to derive results in computational complexity using pebble games, and to study the computational complexity of pebble games.

1.1 Computational Complexity

For the moment, we will focus on the following chain of complexity classes concerning parallel time and memory space:

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}.$$

Recall that the complexity class P is considered as efficiently solvable problems [31]: formally it refers to the problems computable by a deterministic Turing machine in polynomial time, i. e., in deterministic time $n^{O(1)}$ (also denoted $\mathsf{DTime}[n^{O(1)}]$) where n denotes the size of the input under a reasonable encoding.

The complexity class NC^i is considered as problems which can be solved extremely efficiently in parallel: formally it refers to the problems computable by a (sufficiently uniform) family of boolean circuits of polynomial size and of bounded-poly-logarithmic depth, i. e., of size $n^{O(1)}$ and of depth $O\big(\log^i(n)\big)$. This is because, when evaluating a circuit using a simple layer-by-layer strategy, the size of a circuit determines the total number of steps and its depth determines the parallel time required in a parallel computation.

The complexity class NC is considered as problems which can be solved very efficiently in parallel: formally it refers to the problems computable by a (sufficiently uniform) family of boolean circuits of polynomial size and of poly-logarithmic depth, i. e., of size $n^{O(1)}$ and of depth $\log^{O(1)}(n)$. As a result, the complexity class $\mathsf{NC} = \bigcup_{i \geq 0} \mathsf{NC}^i$ is the union of all the levels in the hierarchy of parallel complexity classes. The class NC is robust in the sense that it captures the notion of efficiency under different models of parallel computation (see, e. g., [45]).

Apart from parallel time, memory space is another important resource of computation, and two complexity classes related to memory space are shown in the above chain.

The complexity class L is considered as problems which are solvable with an efficient usage of memory space: formally it refers to the problems computable by a deterministic Turing machine in logarithmic space, i. e., in deterministic space $O(\log n)$ (also denoted $\mathsf{DSpace}[\log n]$).

The complexity class NL is considered as problems which are solvable with an efficient usage of memory space when allowing non-determinism: formally it refers to the problems computable by a non-deterministic Turing machine in logarithmic space, i. e., in non-deterministic space $O(\log n)$.

The Quest for Separations

Most researchers belief that all (at least, almost all) of the inclusions in the above chain of complexity classes for parallel time and memory space are strict. It means that the above complexity classes are believed to be separated.

In particular, for parallel time it is believed that $\mathsf{NC} \subset \mathsf{P}$, i. e., there are efficiently solvable problems which do not admit speedups on parallel machines. In other words, there are inherently sequential problems. Also, it is believed that $\mathsf{NC}^i \subset \mathsf{NC}^{i+1}$, i. e., you can solve more problems with more parallel time.

As a consequence, it is believed that $\mathsf{L} \subset \mathsf{P}$, i. e., not all efficiently solvable problems can be solved with an efficient usage of memory space. Also, some researchers believe that $\mathsf{L} \subset \mathsf{NL}$, i. e., non-determinism saves space, which is the space complexity analogue of the belief that non-determinism saves time, i. e., $\mathsf{P} \subset \mathsf{NP}$.

To study the inclusion or separation of the above complexity classes, one approach is to study certain combinatorial models (e. g., boolean circuits) that captures the usage of parallel time or memory space (see § 1.4 for a detailed discussion), because the inclusion or separation in complexity classes will be mirrored as a corresponding inclusion or separation on combinatorial models. However, an unrestricted separation of any of the above complexity classes have not been proven. Researchers therefore studied their separations under certain restrictions on the combinatorial models of computation, such as the monotone restriction. In some cases, tight lower bounds and a complete separation of almost all of the above complexity classes can be proven *under such restrictions.* For example, under the monotone restriction, the following separations are known on suitable combinatorial models:

$$\mathsf{m\text{-}NC}^1 \subseteq \mathsf{m\text{-}L} \subset \mathsf{m\text{-}NL} \subseteq \mathsf{m\text{-}NC}^2 \subset \mathsf{m\text{-}NC}^3 \subset \cdots \subset \mathsf{m\text{-}NC} \subset \mathsf{m\text{-}P}.$$

In the line of research that achieved the monotone separations [27, 56, 59, 83, 85], three different pebble games appeared in their proofs. Certain parameters of the three pebble games (the pebble costs) scale in the same way as the lower bounds, in a precise sense to be described below. First, the restricted separations are proved by studying complete problems of the complexity classes. Second, in those complete problems (to be discussed in the next subsection), there is an underlying graph structure. Third, by considering pebble games over the graph structure (see § 1.2), the pebble costs in the pebble games scale in the same way as the lower bounds. It turns out that, over any directed acyclic graph, the pebble costs in the three different pebble games agree (Theorem 1), and the pebble cost is known to be connected to other complexity results related to the study of parallel time or of reversibility in computation.

Before we describe this connection, we first introduce the complete problems that can be used for studying the inclusion or separation of complexity classes.

Complete Problems

Complete problems provide an alternative characterization of a complexity class, because they are the "hardest" problems in that class.[1] We mention three such complete problems below, informally.

The **Circuit Valuation Problem** is specified by the following:

Input A boolean circuit $\mathcal{C}$ and an instance x.

Output The boolean value of evaluating the circuit $\mathcal{C}$ on instance x.

The Circuit Valuation Problem is complete for the class P of efficiently solvable problems [65]. Sub-problems of the Circuit Valuation Problem with additional restrictions on the circuit $\mathcal{C}$ are complete for smaller complexity classes: if the circuit $\mathcal{C}$ is restricted to have poly-logarithmic depth $\log^{O(1)} n$, then the sub-problem is complete for the class NC of problems solvable very efficiently in parallel; if the circuit $\mathcal{C}$ is restricted to have bounded-poly-logarithmic depth $O(\log^i n)$, then the sub-problem is *perhaps* complete for the class NC^i of problems solvable extremely efficiently in parallel.

In the **Generation Problem**, there is a collection of m statements. Consider $n := m^3$ implications of the form $u \wedge v \to w$, where each of u, v, w is one of the statements. The implication $u \wedge v \to w$ means that if statement u is true and statement v is true, then statement w is true.

Input A subset of the implications, denoted I, encoded as n bits.

Output Using the implications in I, does statement m logically follows from statement 1?

[1]When working with very efficient complexity classes like NC^1 or L, it is necessary to consider more efficient notions of reductions than, say, logarithmic space reductions or polynomial time reductions. Formally, when defining complete problems for very efficient complexity classes like NC^1 or L, we need to specify a very efficient notion of reduction. We ignore such subtleties for the current discussion (i. e., assume that the circuits are sufficiently uniform), and refer the interested readers to the literature for further discussions on such uniformity [91].

The Generation Problem is a P-complete problem, and is a monotone variant of the first P-complete problem called Path Systems [32]. Sub-problems of the Generation Problem with additional restrictions on the structure of 'the generation graph' are complete for smaller complexity classes like non-deterministic logspace (NL) and the parallel complexity classes (NC, and *perhaps even* NC^i) [8, 57], where the *generation graph* refers to the structure of a logical proof that statement m follows from statement 1 (see e. g., [27, § 3.2]). For the special case where the generation graph is a line graph, the Generation Problem reduces to the Directed Connectivity Problem.

In the **Directed Connectivity Problem**, there is a collection of m vertices. Consider $n := m^2$ directed edges of the form $u \to v$, where each of u, v is one of the vertices. The directed edge $u \to v$ means that vertex u can reach vertex v.

Input A subset of directed edges, denoted E, encoded as n bits.

Output Using the directed edges in E, is vertex m reachable from vertex 1 via a directed path?

The Directed Connectivity Problem is complete for NL, i. e., non-deterministic logarithmic space.

Since complete problems are an alternative characterization of complexity classes, any inclusion or separation of complexity classes can be studied via their corresponding complete problems. We next describe how the complete problem of Circuit Evaluation is used to convert results in pebble games into results in computational complexity.

1.2 Pebble Games

Pebble games were introduced for studying programming languages and compiler construction. The dependency in data flow is modeled by a directed acyclic graph of bounded in-degree, and the pebble games emulate the register allocation and resource usage in the flow of data over the graph. As another closely related example in database systems, a directed acyclic graph models the referential structure of tables in a database,[2] and the pebble games emulate the data access pattern executed by a certain query.

In terms of computational resources, deterministic space is traditionally emulated by *the number of (black) pebbles* required in a one-player pebble game [81, 94], and parallel time is traditionally emulated by the *time* required in a two-player pebble game introduced by Dymond and Tompa [37]. More accurately, the two-player pebble game of Dymond–Tompa emulates *alternating time* as a measure of parallel time (when the number of processors is unbounded): alternating time measures the time spent on an alternating machine [29], which is a natural model of (deterministic) parallel computation.

To give a concrete example of how pebble games are related to computational complexity, we next review the black pebble game briefly and informally.

[2]In reality, the referential structure can have cycles and have large in-degree. We ignore such complications in this exposition.

Black Pebble Game and Space Complexity

Definition 1.2.1 (Black Pebble Game)**.** Fix a DAG G. The black pebble game over G is a one-player game as follows. Each vertex of G can store at most one (black) pebble, and the game begins with no pebbles on G. In each move, *Pebbler* applies one of the following rules: (1) if all immediate predecessors of a are pebbled, *Pebbler* may place a pebble on a (to pebble a); or (2) *Pebbler* may remove a pebble from a (to unpebble a) at any time. The game is over when the sink vertex is pebbled, but all other vertices are unpebbled. A game takes h pebbles if *Pebbler* needs h pebbles to finish the game.

The pebble cost of the black pebble game over a graph G is the number of pebbles needed to pebble the sink vertex.

At a high level (perhaps somewhat incorrectly), the black pebble game is connected with deterministic space complexity as follows. Recall the P-complete Circuit Evaluation Problem: given a boolean circuit $\mathcal{C}$ and an instance x, compute the value $\mathcal{C}(x)$ at the output gate of $\mathcal{C}$. A boolean circuit is formally a directed acyclic graph, where each vertex (i. e., gate) is labeled either by a boolean function (logical-and $\wedge$ or logical-or $\vee$) or an input literal (positive or negated variable). Ignore the labeling on the vertices, and consider a circuit as an unlabeled directed acyclic graph for the discussion in this subsection. Then any black pebbling strategy to pebble the sink vertex (i. e., the output gate of the circuit $\mathcal{C}$) gives an algorithm to evaluate the circuit, if we interpret putting a pebble on a vertex as remembering the value of the corresponding gate in memory. Note that the value at a gate a can be computed if the values at all immediately preceding gates have be computed (Rule 1), and the value at a gate a can be forgotten at any time (Rule 2). Now any pebbling strategy using h pebbles to pebble the sink vertex gives an algorithm to evaluate the circuit in deterministic space $O(h)$.

We next continue the discussion of how pebble games are related computational complexity.

Pebble Games and Computational Complexity

Upper Bounds The study of these pebble games led to non-trivial algorithms, upper bounding resource requirements. For space, Hopcroft, Paul, and Valiant [53] showed that any graph of bounded in-degree on t vertices requires at most $O(t/\log t)$ pebbles in the one-player black pebble game, implying that a time t (deterministic) computation requires at most $O(t/\log t)$ space, i. e., $\mathsf{DTime}[t] \subseteq \mathsf{DSpace}[t/\log t]$.

For parallel time, Dymond and Tompa [37] showed that any graph of bounded in-degree on t vertices requires at most $O(t/\log t)$ time in the two-player game, strengthening the above result to imply that a time t (deterministic) computation requires at most $O(t/\log t)$ alternating time,[3] i. e., $\mathsf{DTime}[t] \subseteq \mathsf{ATime}[t/\log t]$.

[3]The result on alternating time is stronger, since $\mathsf{ATime}[t] \subseteq \mathsf{DSpace}[t]$ [29].

Pebble Games and Complexity Classes Also, certain relationships among different resources of computation can be recast as pebbling results. For example, (a slight variant of) the two-player pebble game of Dymond and Tompa [99] (1) exactly *characterizes* the parallelism of different complexity classes (e. g., NC^i, NC and P); and (2) can re-derive known complexity results, including the simulation of Savitch [92] showing that $\mathsf{NL} \subseteq \mathsf{DSpace}[\log^2 n]$.

Lower Bounds and Trade-Offs The study of pebble games also gave lower bounds on resource requirements or indicating trade-offs of different resources *in restricted models of computation*.

Paul, Tarjan, and Celoni [82] constructed a graph of bounded in-degree on t vertices which requires $\Omega(t/\log t)$ pebbles in the one-player game emulating space; and by a simulation argument in pebble games, this graph also requires $\Omega(t/\log t)$ time in the two-player game emulating alternating time [37]. These lower bounds are tight given the upper bounds on pebble games. To the best of our knowledge, we still don't know how to save more space or alternating time (a measure of parallel time) than the pebbling algorithms for a P-complete problem.[4]

In addition to the **black pebble game** and the **Dymond–Tompa pebble game**, two other pebble games were used in the combinatorial approach for proving restricted lower bounds. Their usage will be discussed in details later in § 1.4, and for the moment, we give an overview for each of them below.

Raz–McKenzie pebble game Raz and McKenzie [85] introduced a two-player pebble game over a directed acyclic graph, motivated by the depth complexity of decision trees solving search problems [74]. The pebble game was first used for proving lower bounds on monotone alternating time (see § 1.4). Later, it was applied to proof complexity, e. g., [18], and inspired the use of *pebbling contradictions* which form the basis of most time-space trade-offs and many separation results in proof complexity (to be discussed in § 1.6). Elias and McKenzie [39] made explicit the role of the pebble game in the monotone results, and initiated a study of the pebble cost over different directed acyclic graphs.

Reversible pebble game Bennett [15] initiated the study of reversible computation as a possibility to eliminate (or significantly reduce) energy dissipation in logical computation. Reversible computation is increasingly important (i) because computing chips are getting smaller and energy dissipation is becoming an issue; and (ii) because observation-free quantum computation is inherently reversible. Bennett studied the time and space complexity in reversible simulation of irreversible computation, and as an abstraction mentioned *reversible pebble game* [16], which is the reversible version of the black pebble game. This led to the study of the reversible pebble game over different directed acyclic graphs [64, 70] and its relation to time-space trade-offs in reversible simulation of irreversible computation [21, 66, 69, 102].[5] Later, in the combinatorial approach, Potechin [83] independently and

[4]For example, concerning circuit depth (to be introduced next), although there are some non-pebbling algorithms for trading circuit depth for (semi-unboundedness of) fan-in [72, 103], those algorithms do not give a saving in depth when simulated on circuits of *bounded fan-in*.

[5]A reversible simulation of an irreversible computation can be considered as a compiler for converting a source program, which may be irreversible, into a functionally equivalent object program, which is reversible.

implicitly used the reversible pebble game (made explicit in [27]) for proving lower bounds on monotone space complexity (see § 1.4).

The reader is referred to the literature for further discussions on the black pebble game [64, 80], the Dymond-Tompa pebble game [37, 99], the Raz–McKenzie pebble game [39, 85], and the reversible pebble game [16, 64].

1.3 Our Results in Pebble Games

Theorem 1 (Just a Pebble Game)**.** *The Dymond–Tompa pebble game, the Raz–McKenzie pebble game, and the reversible pebble game of Bennett have the same pebble cost. That is, for any directed acyclic graph having a unique sink vertex, the following are equivalent for pebbling the sink vertex:*

- *it takes h time in the Dymond–Tompa pebble game (§ 3.1);*
- *it takes h time in the Raz–McKenzie pebble game (§ 3.2);*[6] *and*
- *it takes h pebbles in the reversible pebble game of Bennett (§ 3.3).*

Corollary 1.3.1 (Upper Bounds on Pebble Costs)**.** *Any directed acyclic graph on n vertices with bounded in-degree has cost at most $O(n/\log n)$ in the Raz–McKenzie pebble game or the reversible pebble game.*

Corollary 1.3.2 (Raz–McKenzie versus Black Pebbling)**.** *The Raz–McKenzie pebble cost is at least the (irreversible) black pebble cost.*

Remark 1.3.3 (Connections in Pebble Games). Theorem 1 establishes a connection among different pebble games introduced for very different reasons.

1. It strengthens and explains the simulation result of Dymond–Tompa [37], which states that the Dymond–Tompa pebble cost of a graph is at least the black pebble cost of a graph, mirroring the inclusion $\mathsf{ATime}[t] \subseteq \mathsf{DSpace}[t]$. It is because the reversible pebble cost is at least the black pebble cost.

2. It explains some of the known results in computational complexity to be reviewed next (§ 1.4).

3. It explains some of the known results in proof complexity to be reviewed next. In particular, Corollary 1.3.2 gives a new connection between two pebble games studied in proof complexity (see § 1.6).

[6] This solves an open problem raised in Elias–McKenzie [39] for connecting their pebble game with other pebble games.

4. It connects the pebbling results in the Dymond–Tompa pebble game [37], the Raz–McKenzie pebble game [39, 85], and the reversible pebble game [27, 64, 70, 83] over different directed acyclic graphs, e. g., line graphs, pyramid graphs, butterfly graphs, or the hard-to-pebble graph in [82]. For example, this transfers the tight bound of $\Theta(n/\log n)$ pebbles for the graph in [82] over the Dymond–Tompa pebble game to the Raz–McKenzie pebble game and to the reversible pebble game, which was not known before.

To better understand the connection between pebble games and complexity, we next review the combinatorial approach for proving lower bounds.

1.4 Combinatorial Models of Computation

We briefly recall two combinatorial models of computation which characterize parallel time and memory space. We ignore the issues of uniformity for the moment, by assuming that the combinatorial models are sufficiently uniform. Alternatively, the reader may want to append $/\mathsf{poly}$ to every machine-based complexity class.

Circuits Parallel time is modeled by the *depth* of a circuit: $\mathsf{ATime}[t] = \mathsf{Depth}[t]$ [91]. Recall that $\mathsf{ATime}[\cdot]$ refers to alternating time, a measure of parallel time on alternating machines. This thesis considers *boolean* circuits *of bounded fan-in* unless otherwise noted.

Switching Networks Memory space of a deterministic computation is modeled by the *size* of a switching network: $\mathsf{DSpace}[s] = \mathsf{SNSize}[2^{\Theta(s)}]$. A switching network computes by reachability in a symmetric way, where the symmetry/reversibility mirrors determinism [66, 90]. The direction $\mathsf{DSpace}[s] \subseteq \mathsf{SNSize}[2^{\Theta(s)}]$ is folklore [68] (e. g., see [83, §2]), and $\mathsf{SNSize}[2^{\Theta(s)}] \subseteq \mathsf{DSpace}[s]$ is proved by Reingold [90].

Researchers commonly add restrictions to the combinatorial models to get lower bounds in restricted settings. We recall two such restrictions below.

Monotone Restriction A boolean function is monotone if flipping an input bit from False to True cannot flip the output bit from True to False. When computing monotone boolean functions, it is common to add a monotone restriction to the model, which is to disallow logical negation. Monotone restriction applies to circuits and switching networks naturally (as opposed to e. g., Turing machines).

Problem-Specific Restriction In addition to the syntactic restriction of monotonicity, researchers also studied different semantic restrictions. Sometimes, the semantic restriction is designed with the computational problem in mind. We give one example below.

Karchmer and Wigderson [59] characterized the depth complexity of circuits as the communication complexity of a two-party game. To explore the intuition given by the communication game, and in particular whether depth complexity scales with iterated composition of hard functions (i. e., direct-sum phenomenon), Karchmer, Raz, and Wigderson [58, §6] invented a communication game called *universal composition relation* to model iterated composition of hard functions, where the structure of iterated composition forms a tree (of branching factor and height about $\log n$). Roughly, the universal composition relation is similar to a standard communication game, except that the parties are required to output a bit on some leaf node of the tree (intuitively, the parties need to locate a branch of the tree leading to the leaf node, hence the communication complexity should scale with the height of the tree). Note that this restriction makes sense only for the problem of universal composition relation (unlike the monotone restriction, which applies to any monotone boolean function), and also only in the model of communication game.

Previous Results

For the depth complexity of semantically restricted circuits, Edmonds, Impagliazzo, Rudich, and Sgall [38] employed an information-theoretic counting argument to show that the universal composition relation of d levels of k-bit boolean function requires $dk - O\big(d^2(2\log k)^{1/2}\big)$ bits of communication when $d = \log n/\log\log n$ and $k = \log n$. Håstad and Wigderson [50] subsequently constructed a sub-additive measure to show that the universal composition relation of d levels of k-bit boolean function requires $\big(1 - o(1)\big)dk$ bits of communication when $d = o(\sqrt{k/\log k})$ and $k = \log n$. Both results suggest the *semantic analogue* of the separation $\mathsf{NC}^1 \subset \mathsf{NC}^2$, where NC^i are circuits of polynomial size and of $O\big(\log^i(n)\big)$ depth.

For the depth complexity of monotone circuits, Karchmer and Wigderson [59] introduced the communication game framework to prove that the NL-complete problem of Directed Connectivity requires $\Omega(\log^2 n)$ depth on monotone circuits,[7] implying $\mathsf{m\text{-}NC}^1 \subset \mathsf{m\text{-}NL} \subseteq \mathsf{m\text{-}NC}^2$. Raz and McKenzie [85] extended the information-theoretic argument of Edmonds–Impagliazzo–Rudich–Sgall to reprove the Directed Connectivity result of Karchmer–Wigderson,[7] and in addition showed the separation of $\mathsf{m\text{-}NC} \subset \mathsf{m\text{-}P}$ and of $\mathsf{m\text{-}NC}^i \subset \mathsf{m\text{-}NC}^{i+1}$, by studying the P-complete problem of Generation.[8]

The results on monotone circuits were subsequently strengthened to monotone switching networks, i. e., from (monotone) alternating time to (monotone) deterministic space.[9] Departing from the communication game of Karchmer–Wigderson which forms the basis of most results concerning circuit depth [19, 38, 44, 47, 56, 58, 59, 85, 86], Potechin [83] introduced a Fourier analytic framework, proving that Directed Connectivity requires monotone switching networks of size $n^{\Omega(\log n)}$, which can be interpreted as proving $\mathsf{m\text{-}L} \subset \mathsf{m\text{-}NL}$ on monotone switching networks.[10] The Fourier analytic framework is recently reinterpreted as describing an enumerative-combinatorial invariant [27], and the lower bound of Raz–McKenzie on

[7]Karchmer–Wigderson [59] and Raz–McKenzie [85] in fact proved the same lower bound of $\Omega(\log^2 n)$ for the depth of monotone circuits solving *Un*directed Connectivity, which is L-complete [90].

[8]We focus on the depth complexity of efficient problems, i. e., inside P or $\mathsf{m\text{-}P}$ under suitable restrictions, and did not mention, e. g., the lower bounds of k-clique [6, 27, 44, 49, 85, 87] or matching [86] on monotone circuits.

[9]By a simulation argument mirroring $\mathsf{ATime}[t] \subseteq \mathsf{DSpace}[t]$ (see e. g., [27, §1]), a lower bound of $2^{\Omega(t)}$ on the size of (monotone) switching networks translates to a lower bound of $\Omega(t)$ on the depth of (monotone) circuits, hence the result on monotone switching network is stronger.

[10]It should be noted that there are at least two combinatorial models for (non-uniform) $\mathsf{m\text{-}L}$ in the literature: as monotone (boolean) circuits (of bounded fan-in) of logarithmic *width* and polynomial size [46,

Generation is strengthened to monotone switching networks. For further discussion on the switching network model or the Generation Problem, see the references in [27].

As mentioned in § 1.2, two pebble games were used in the monotone results. In general, the monotone circuit depth for Generation scales as $\Omega(h \log n)$ when $h \le n^{O(1)}$ is the Raz–McKenzie pebble cost of the generation graph [39]; and the monotone switching network size for the Generation Problem scales as $n^{\Omega(h)}$ when $h \le n^{O(1)}$ is the reversible pebble cost of the generation graph [27].[11]

1.5 Our Results in Computational Complexity

Theorem 1 has the following consequence by the discussion in § 1.4

Corollary 1.5.1 (Improved Bounds for Generation). *For any directed acyclic graph G, for the sub-problem of Generation where the generation graph is restricted to G, the lower bound on the size of monotone switching networks [27] implies the lower bound on the depth of monotone circuits [39] up to constant factors.*

In addition, at a high level, we combine the semantic separation of circuit depth [38, 50] with the framework of Dymond–Tompa game. Instead of considering the *iterated multiplexor* problem of universal composition relation with a tree structure [58, §6], we consider the *iterated indexing* problem over any directed acyclic graph. This minor twist completely changed the combinatorics of the problems. Our computational problem, called DAG evaluation (Definition 4.0.11), is a generalization of the tree evaluation problem considered by Cook, McKenzie, Wehr, Braverman, and Santhanam [33].[12] The DAG evaluation problem is a slight variant of the P-complete problem of circuit evaluation, and it captures the combinatorial essence of the Generation Problem discussed above [39, 75].

For this computational problem, we consider a problem-specific restriction called *output-relevant* circuits (§ 4.2). Roughly, in terms of the two-party communication game of Karchmer and Wigderson [59], a circuit is output-relevant if the two parties are required to output a relevant bit, which is a more natural restriction (than the output-leaf restriction in the universal composition relation) for studying depth complexity.[13] In particular, it is unclear how to turn the universal composition relation into a proper Karchmer–Wigderson game

47], or as monotone switching networks of polynomial size [83, 88]. It appears that the two models are not comparable. This work focuses on monotone switching networks of polynomial size as the combinatorial model for (non-uniform) m-L.

[11]The proof in the journal vesrion of [27] clearly works for any directed acyclic graph.

[12]The generalization to DAG is also considered by Wehr [101]. For comparison, Wehr studied the branching program model, and proved a lower bound (instead of a tight bound) in terms of the black pebble cost of the directed acyclic graph under a relatively restricted setting.

[13]Output-relevance is motivated by the efficiency of shallowly packing certificates for use by two competing provers, the alternation of which governs the combinatorial recurrence behind both the Dymond–Tompa game and the Karchmer–Wigderson game. For further justification, see Remark 4.2.5.

(so that it corresponds properly to circuit depth), which is *not* the case for output-relevant circuits.

Theorem 2 (Pebbling is Optimal)**.** *Consider a directed acyclic graph G whose Dymond–Tompa game takes h time. Any output-relevant circuit solving the DAG evaluation problem over G of bit-length k has depth $\Omega(hk)$ when $2^k \geq |V|^{\Theta(1)}$.*

Theorem 2 is complemented by a matching upper bound, that there is a circuit of depth $O(hk)$ implementing the pebbling algorithm of Dymond–Tompa. Unlike previous bounds on monotone circuits [39, 56, 85] which are tight up to $n^{\Theta(1)}$, our bounds on restricted circuits are tight up to multiplicative factors. The tight bound can be interpreted as the semantic separation of NC from P and of NC^i from NC^{i+1}, by considering the pyramid graph of height $\Theta(\log^i n)$. In terms of circuit depth,[14] Theorem 2 gives an exponential improvement on an incomparable (but more natural[13]) model over previous results [38, 50], which suggested only a semantic separation of NC^1 from NC^2.

Remark 1.5.2 (Circuit Depth and NC versus P)*.* Theorem 2 supports the attempt to separate NC from P (and NC^i from NC^{i+1}) by focusing on circuit *depth*. By connecting (non-monotone but restricted) circuit depth with the Dymond–Tompa game, Theorem 2 gives evidence to support the attempt to study circuit depth *alone* (as opposed to a combination of depth and size) for separating NC from P,[14] due to very similar recurrence in the minimization of the depth complexity in the Karchmer–Wigderson game and in the Dymond–Tompa game.[15] More importantly, now Theorems 1 and 2 together put many of the existing combinatorial lower bounds concerning circuit depth for separating NC from P [27, 38, 50, 56, 58, 83, 85] into the Dymond–Tompa game framework. This connection explains the same scaling in lower bounds given by apparently different pebble games: there is just one pebble game in disguise.[16] However, this raises an interesting follow-up question: why do the different analyses for different combinatorial arguments under different restricted settings converge to the same pebble game (which basically characterizes parallelism)? Also, the Dymond–Tompa game lower bounds the depth complexity on these restricted computational models, so how far (i. e., how general a model) does this correspondence hold?

[14]We will discuss some related approaches for separating complexity classes in Chapter 7, including approaches that consider both the size and depth of a circuit, e. g., by algebro-geometric invariants [76, 78, 79], multi-party communication complexity [20, 25], competing-prover protocols [62], and block-respecting simulations [73].

[15]Another evidence was the monotone separation of $\mathsf{m\text{-}NC}$ from $\mathsf{m\text{-}P}$ (and of $\mathsf{m\text{-}NC}^i$ from $\mathsf{m\text{-}NC}^{i+1}$) by Raz–McKenzie [85], where the lower bound on depth holds regardless of size (also implied by its strengthening [27]), although this monotone evidence is weak due to known exponential separations of monotone depth from non-monotone depth, e. g., for matching [86].

[16]For example, this may explain why in the Fourier analytic framework [83], it is sufficient to consider reversible pebbling configurations [27] instead of knowledge sets [83]. Also, Corollary 1.5.1 completes the picture of simulation results between circuits and switching networks, for the sub-problem of Generation whose generation graph is *any directed acyclic graph* (in addition to the line graphs or the pyramid graphs known previously).

We next briefly review the motivation for studying proof complexity, before we state our results on the depth of resolution refutations.

1.6 Proof Complexity

The study of proof complexity was initiated by Cook and Reckhow [34], who showed that $\mathsf{NP} = \mathsf{co\text{-}NP}$ iff there is an efficient (i. e., polynomially bounded) proof system. Since its introduction, proof complexity has been studied by many researchers. We mention below two such motivations relevant to this thesis.

Combinatorial methods for studying computational complexity One way to approach the distant goal of separating P from NP is to show that $\mathsf{NP} \neq \mathsf{co\text{-}NP}$ (since $\mathsf{P} = \mathsf{co\text{-}P}$), by proving super-polynomial lower bounds on successively stronger proof systems for propositional tautologies. Hence proving combinatorial lower bounds on proof systems can be seen as sharpening our combinatorial tools for eventually separating complexity classes, if possible.

Analysis of practical automated theorem-provers Lower bounds and trade-off results for seemingly weak and restricted proof systems already apply to the performance characteristics of most of the automated theorem-provers used in practice. For example, after failing to search for a satisfying assignment, the execution of the proof search algorithm in [35, 36] (known as DLL or DPLL) corresponds to a refinement (i. e., a restricted version) of resolution refutation whose structure forms a tree, hence called a *tree-like* resolution. Resolution refutation is a weak proof system in theory but widely used in practice. The study of trade-off results, or the comparison of different variants of proof systems (e. g., tree-like versus general), have consequences to the performance of different proof search strategies used in practice (see, e. g., [55]).

Resources: Size, Space, and Rank

Out of the many resources considered for studying proof complexity, we mention below three resources relevant to this thesis.

Size The *size* of a refutation is the number of clauses,[17] or equivalently (up to a factor of two) the number of derivation steps. Hence the size complexity lower bounds the running time of a certain class of proof search algorithms (even allowing non-determinism). The size complexity is widely regarded as the most important complexity measure.

Space Among others, the *space* of a refutation may count the number of clauses (*clause space*) or the number of variables (*variable space*[18]) in any configuration in a refutation.

[17] Some literature calls this measure *length*, reserving size as the total number of symbols in a refutation (see e. g., the survey by Nordström [80]). The two measures are polynomially related, and are used interchangeably in this thesis.

[18] The term *variable space* was used in the literature to mean two related but different concepts: the number of literals counted with repetitions, or the number of variables counted without repetitions. The

Hence the space complexity measures the memory requirement (which is often a limiting resource for clause learning) of a certain class of proof search algorithms. Space complexity (in the configuration-style) was introduced by Esteban and Torán [41] and extended by Alekhnovich, Ben-Sasson, Razborov, and Wigderson [3].

Rank The *rank* of a refutation measures the sequentiality of a certain class of proof search algorithms, e. g., for resolution-based proof systems, it is *depth*; and for semi-algebraic proof systems (i. e., polynomial threshold proof systems like Gomory–Chvátal cutting planes or Lasserre/Positivstellensatz), it is the number of *rounds*. At a high level, the rank of many proof systems may be related: the rank (depth) of the weak proof system of resolution is related to another measure called *width* [17, 98], which in certain cases can be used for proving a rank (round) lower bound on the very strong proof system of Lasserre/Positivstellensatz [28, 48, 93, 96].[19] The depth of resolution refutations was first systematically studied by Urquhart [98], and the number of rounds of different semi-algebraic proof systems have been routinely studied, e. g., in proof complexity [22, 23] or in hardness of approximation [28, 93, 96].

There are some known relationships among different resources, connecting the most important resource of size to other resources. This gives another justification for studying space and rank.

Space and Size Clause space upper bounds (with some loss and via another measure *width*) the logarithm of size for resolution [7]. As a partial converse, the logarithm of size upper bounds clause space for tree-like resolution [41]. As for variable space, a lower bound on variable space can be escalated to a lower bound on clause space via substitution [13], and this connection yielded one of the tightest size-space trade-offs currently known in proof complexity by studying pebbling contradictions [13].

Rank and Size Urquhart argued that rank is significant since "all proofs of resolution size lower bounds implicitly prove depth lower bounds" [98]. In practice, there are natural rank-based procedures for generating refutations in some proof systems, e. g., the Davis–Putnam procedure for resolution [36], (a variation of) the Gröbner basis algorithm for Polynomial Calculus [30], and the semi-definite programming of Lasserre/Positivstellensatz [48, 67]. In this sense, rank measures the time needed for *deterministically* generating refutations in many practical proof systems, and for them rank may be as important as size (e. g., in [28, 48, 67, 93, 96]).

Previous Results

The pebbling approach is routinely studied in proof complexity, in the form of *pebbling contradictions*, i. e., an unsatisfiable formula with one boolean variable per vertex, stating that (1) all source variables are true; (2) truth propagates through the graph; and (3) some sink

latter meaning, which is recently becoming the standard usage [80, Footnote 5], is used here.

[19] A paper even suggests that any rank lower bound on resolution can be directely translated (with some loss) into a rank lower bound on some strong proof systems including Lasserre [9], but an anonymous reviewer claims that this proof is broken.

variable is false. Often, certain pebbling properties (e. g., time and space) of the underlying graph is escalated to the formula via substitution [13] or lifting [54].

The study of pebbling contradictions gave many of the best known separations (of different proof systems) and trade-offs (of different resources). In particular, the **Raz–McKenzie pebble game** has been used for separating tree-like and general cutting plane refutations [18], and the (irreversible) **black pebble game** has been used for separating tree-like and general resolution size [12, 98], regular and general resolution size [4], DPLL (tree-like resolution) and a theoretical proof system based on clause learning algorithms [10], Nullstellensatz and Polynomial Calculus degree [22], and the hierarchy of tree-like k-DNF-resolution and general resolution size [40].[20]

1.7 Our Results in Proof Complexity

Let Σ_G denote the pebbling contradiction over G (Definition 5.1.2, see also [80, 98]). The substitution construction of Alekhnovich–Razborov [11] is denoted $\Sigma^{\oplus}$ below; for generalizations, see [13]. Denote $\textsc{Val}(G)$ as the value of the graph G, i. e., the pebble cost in the Dymond–Tompa game, or equivalently, in the Raz–McKenzie pebble game or in the reversible pebble game (Theorem 1).

Theorem 3 (Depth of Pebbling Contradictions)**.** *Fix a directed acyclic graph $G = (V, E)$ with a unique sink τ. The depth complexity of resolution refutation for Σ_G is* exactly *the pebble cost in the Raz–McKenzie pebble game to pebble the sink vertex of $\hat{G}$, where $\hat{G} := (V \cup \{\hat{\tau}\}, E \cup \{(\tau, \hat{\tau})\})$ is G augmented with an extra vertex $\hat{\tau}$ as the new sink.*

It is easy to see that the variable space needed for resolution refutation of Σ_G is at most the (irreversible) black pebble cost of G by simulating a black pebbling strategy. Take G to be the line graph on n vertices, this gives a separation of variable space (at most 2) and depth (at least $\log n$), solving an open problem raised by Urquhart [98, Problem 7.2].

Theorem 4 (Tight Size Bounds for Tree-Like Resolution)**.** *The tree-like resolution refutation of $\Sigma_G^{\oplus}$ has size complexity $2^{\Theta(\textsc{Val}(G))}$.*

Remark 1.7.1 (Decision Tree and Reversible Pebble Game). Theorem 3 gives an exact characterization, improving on the lower bound of Urquhart [98]. Exact combinatorial characterization can be useful for translating results to different settings, e. g., Berkholz [17] recently connected the exact combinatorial characterization of resolution width [7] with the combinatorial game of Kasai–Adachi–Iwata [2, 60], proving an unconditional time lower bound. Theorem 4 can be seen as a result in this direction. Also, Theorem 3 allows us to settle the space complexity of the minimum depth of resolution refutations in § 1.8.

[20] We did not mention the use of black-white pebbling for time-space trade-offs [11, 13], see e. g., [80].

Moreover, this shows that the lower bounds in previous works [10, 12, 40], in particular those concerning the depth of resolution refutations [98], the degree of Polynomial Calculus [22][21], and the size of tree-like cutting plane refutations [18], morally follow from the pebble cost of a single pebble game, wearing different costumes listed in Theorem 1. Since the Dymond–Tompa game and the Raz–McKenzie pebble game were introduced for studying depth complexity, this may explain the use of the (reversible) black pebble game in Theorem 3 and in previous works.

Recall that k-DNF-resolution (Definition 5.2.1) extends the usual resolution.

Theorem 5. *Any k-DNF-resolution refutation of Σ_G has depth at least $1 + (\textsc{Val}(G) - 1)/k$.*

It is not hard to see that the lower bound should worsen with k, the arity of the DNF resolution. For constant k (which roughly corresponds to the case of boolean circuits of bounded fan-in), this lower bound is tight up to constant factors.

1.8 Our Results in Space Complexity of Pebble Costs and Depth of Resolution Refutations

Recall that it is of interest to compute the pebble costs due to their connections to different complexity results, e. g., we want to compute the scaling in (i) the most efficient algorithms known [37, 42, 99]; (ii) lower bounds under restrictions [39, 83, 85]; and (iii) resource usage of certain proof search algorithms [18, 22, 98]. We show that computing the pebble cost in any of three pebble games is PSPACE-hard under logspace reduction, giving a new link between pebble games and computational complexity.

Theorem 6 (Log-Space Reduction)**.** *There is a logspace algorithm that, given a quantified boolean formula φ with m clauses over n variables, outputs a graph $\hat{G}_\varphi$, such that φ is satisfiable iff the pebble cost of $\hat{G}_F$ is at most $\gamma + 1$, where $\gamma := 7 + m + 3n + \alpha_n$ and α_n is the number of universal quantifiers in φ. Moreover, after deleting the sink node of G_φ, the resulting graph also has a unique sink node.*

Theorem 6 follows from Construction 6.1.7 and is summarized as Theorem 14 in § 6.1. The pebble costs, as well as the minimum depth of resolution refutation, are known to be computable in PSPACE (see, e. g., [98]). Due to Theorems 1 and 3, we have the following corollary.

Theorem 7 (PSPACE-Completeness)**.** *It is PSPACE-complete to compute (1) the pebble cost in the Bennett–Dymond–Tompa–Raz–McKenzie pebble game; or (2) the minimum depth of resolution refutation.*

[21]Buresh-Oppenheim, Clegg, Impagliazzo, and Pitassi only claimed the result in terms of the (irreversible) black pebble game, but it appears that their proof [22, Lemma 4.10] works also in terms of the reversible pebble game, due to its combinatorial recurrence (Proposition 3.5.2 and Corollary 3.5.10).

This answers an open problem raised by Urquhart [98, Problem 7.1].

1.9 Techniques

The equivalence of the pebble games is proved by simulation arguments, on observing their similarities in combinatorial recurrence. The results on restricted models fall into three categories: (1) circuits under semantic restriction (thrifty circuits versus output-relevant circuits); (2) computational models under monotone restriction (monotone circuits versus monotone switching networks); and (3) weak proof systems (resolution refutations versus k-DNF-resolution refutations). Note that in all cases, the second model simulates (thus is stronger than) the first model.

All the upper bounds proved in this work are given by a pebbling strategy (of one of the pebble games listed in Theorem 1),[22] implemented in the weaker models. As for the lower bounds in slightly stronger models, although the computation appears not to follow a pebbling strategy, morally we can always decode an underlying strategy (or a family of strategies). In other words, the hardness of pebbling is escalated to the hardness in the respective, slightly stronger models.

As for the actual execution of the lower bound arguments, we consider the specifics of the models: (1) for circuits under semantic restriction, our lower bound is based on the extension by Raz–McKenzie [85] of the information-theoretic adversary argument by Edmonds–Impagliazzo–Rudich–Sgall [38]; (2) for computational models under monotone restriction, the lower bounds are based on the extension by Chan–Potechin [27] of the framework of invariants by Potechin [83] or the extension by Raz–McKenzie [85] mentioned above (first proved in [39], see [26, Appendix B]); and (3) for weak proof systems, our lower bound is an adversary argument based on the recurrence of the Raz–McKenzie pebble game.

Finally, the results on the space complexity of the pebble games, and of the minimum depth of resolution refutations, are proved via a gadget reduction from the PSPACE-complete problem of True Quantified Boolean Formulas.

1.10 Organization

Preliminary definitions and conventions are collected in Chapter 2.

The three pebble games are introduced, and proved equivalent, in Chapter 3. The equivalence of the three pebble games (Theorem 1) follows from Theorems 8 and 9.

The DAG evaluation problem is treated in Chapter 4, which proves the lower bound of Theorem 2 as Theorem 12, based on the information theoretic counting arguments in

[22]Note that the upper bound for the problem of Generation on monotone models is not given by an optimal pebbling strategy, unlike other problems considered here, e. g., graph reachability and the DAG evaluation problem.

Appendix A. The lower bound is complemented by a matching upper bound, proved as Theorem 11. Proposition 4.2.6 shows that output-relevant circuits simulate thrifty circuits.

The proof complexity of resolution refutations is studied in Chapter 5, which proves Theorems 3 to 5.

The space complexity of the pebble costs, and of the minimum depth of resolution refutations, is studied in Chapter 6. This is proved via a gadget reduction, using the gadgets collected in Appendix B.

Other approaches for separating complexity classes around P are discussed in Chapter 7, and future directions are listed in Chapter 8.

Most of the materials (i. e., except the results on the space complexity of pebble games in Chapter 6) have appeared in another work by the author [26].

Chapter 2

Preliminaries

Denote $[n] := \{0, 1, \ldots, n-1\}$. A subset S of a set A is identified with its indicator function $\chi_S \in 2^A \cong \{0,1\}^A$, where $\chi_S(i) = 1$ iff $i \in S$.

Notation 2.0.1 (Restriction)**.** The notation $\restriction$ will be overloaded for different (non-conflicting) definitions. In general, for a tuple x in a product space $X := A^B$ where A and B are sets, $x{\restriction}_b := x_b \in A$ denotes the entry of x indexed by $b \in B$. However, there is an exception for instances to the evaluation problem BDEP^k_G (Notation 4.2.1). In any case, for a subset $C \subseteq B$, $x{\restriction}_C$ denotes the tuple $\langle x{\restriction}_c \rangle_{c \in C} \in A^C$; for a subset $Y \subseteq X$, $Y{\restriction}_b := \{y{\restriction}_b\}_{y \in Y}$ for $b \in B$ and $Y{\restriction}_C := \{y{\restriction}_C\}_{y \in Y}$ for $C \subseteq B$.

This work focuses on boolean circuits of fan-in two having a single output gate, and the main concern is their depth complexity, measured by the number of edges on the longest path from an input gate to the output gate (which may be zero), where negation costs no increase in depth.

We fix our notation for directed acyclic graphs below. For brevity, immediate predecessors are called in-neighbors here, and immediate successors are called out-neighbors.

Notation 2.0.2 (Directed Acyclic Graph)**.** Consider a directed acyclic graph (DAG) $G = (V, E)$. For every vertex $a \in V$, denote its in-neighbors as $\delta^{\text{in}}(a) := \{b \in V : (b, a) \in E\}$ and out-neighbors as $\delta^{\text{out}}(a) := \{b \in V : (a, b) \in E\}$, and in-degree as $\deg^{\text{in}}(a) := |\delta^{\text{in}}(a)|$. For the DAG G, its source vertices are $U := U(G) := \{a \in V : \delta^{\text{in}}(a) = \emptyset\}$ and sink vertices are $W := W(G) := \{a \in V : \delta^{\text{out}}(a) = \emptyset\}$.

Chapter 3

Equivalence of Pebble Games

We first informally review the three pebble games (§§ 3.1 to 3.3), and then show their equivalence (§§ 3.4 and 3.5).

To avoid confusion with the two-*party* communication games of Karchmer–Wigderson (see § 4.1) or of Raz–McKenzie (called Dart game), this thesis refers to *Pebbler* and *Challenger* (or *Colorer*) as the two *players* in a Dymond–Tompa game (or Raz–McKenzie pebble game).

3.1 Dymond–Tompa Game

The following version of the Dymond–Tompa game is needed, where *Pebbler* only pebble one vertex in each round, similar to the variant used in [24]. Concerning the number of pebbles, this one-pebble-per-round version is clearly equivalent to the original multiple-pebble-per-round version by Dymond and Tompa (by a simulation argument). The informal Definition 3.1.1 is is formalized as Definition 3.4.1 in § 3.4. Its pebble cost is the time needed.

Definition 3.1.1 (Dymond–Tompa Game [37])**.** Fix a DAG G. The Dymond–Tompa game (DT_G) over G is a two-player (competitive) game as follows. The two players, *Pebbler* and *Challenger*, alternate to move. The *Pebbler* begins by pebbling a sink vertex of G, which is then challenged by *Challenger*. In all subsequent rounds, *Pebbler* places a pebble on a vertex of G, then *Challenger* either (1) rechallenges the currently challenged vertex; or (2) challenges the vertex pebbled by *Pebbler*. The game is over when *Challenger* challenges $a \in V$, but all in-neighbors of a are pebbled. A game takes h time if *Pebbler* needs h pebble moves to win, against an optimal *Challenger* play.

3.2 Raz–McKenzie Pebble Game

Raz–McKenzie [85] employed the following pebble game in their adversarial strategy for proving lower bounds on the depth of monotone circuits. Elias–McKenzie [39] initiated the study of the pebble game over different directed acyclic graphs. The informal Definition 3.2.1 is formalized as Definition 3.4.12 in § 3.4. Its pebble cost is the time needed.

Definition 3.2.1 (Raz–McKenzie Pebble Game)**.** Fix a DAG G. The Raz–McKenzie pebble game (RM_G) over G is a two-player (competitive) game as follows. The two players, *Pebbler* and *Colorer*, alternate to move. The *Pebbler* begins by pebbling a sink vertex of G, which is then colored red by *Colorer*. In all subsequent rounds, *Pebbler* places a pebble on a vertex of G, then *Colorer* colors this vertex either (1) as red; or (2) as blue. The game is over when some vertex $a \in V$ is colored red, but all in-neighbors of a are colored blue. A game takes h time if *Pebbler* needs h pebble moves to win, against an optimal *Colorer* play.

3.3 Reversible Pebble Game

Bennett [16] mentioned reversible pebble game as an abstraction for a reversible simulation of irreversible computation. The informal Definition 3.3.1 is formalized as Definition 3.5.4. Its pebble cost is the number of pebbles needed.

Definition 3.3.1 (Reversible Pebble Game)**.** Fix a DAG G. The reversible pebble game over G is a one-player game as follows. Each vertex of G can store at most one pebble, and the game begins with no pebbles on G. In each move, *Pebbler* applies one of the following rules: (1) if all in-neighbors of a are pebbled, *Pebbler* may place a pebble on a (to pebble a); or (2) if all in-neighbors of a are pebbled, *Pebbler* may remove a pebble from a (to unpebble a). The game is over when the sink vertex is pebbled, but all other vertices are unpebbled. A game takes h pebbles if *Pebbler* needs h pebbles to finish the game.

3.4 When Dymond–Tompa meet Raz–McKenzie

This section formalizes the Dymond–Tompa game (§ 3.4) and the Raz–McKenzie pebble game (§ 3.4), and proves their equivalence (§ 3.4).

Dymond–Tompa Game

Definitions 3.4.1 and 3.4.2 formalize the intuitive Definition 3.1.1 for the Dymond–Tompa Game.

Definition 3.4.1 (Dymond–Tompa Game Tree)**.** Fix a DAG $G = (V, E)$. A *configuration* of the Dymond–Tompa game (DT_G) over G is a tuple $\langle\!\langle P, r, c\rangle\!\rangle$, where $P \subseteq V$ are the pebbled vertices, $r \in P \cup \{\perp\}$ is the vertex just pebbled, and $c \in P$ is the vertex under challenge. The player taking the turn in $\langle\!\langle P, r, c\rangle\!\rangle$ is *Pebbler* if $r = \perp$, and is *Challenger* if $r \in P$.

The *initial configuration* for G is $C_G := \langle\!\langle \{\tau\}, \perp, \tau\rangle\!\rangle$,[1] and the game is *over* in a configuration $\langle\!\langle P, r, c\rangle\!\rangle$ if $r = \perp$ and $\delta^{\text{in}}(c) \subseteq P$. A configuration $C := \langle\!\langle P, r, c\rangle\!\rangle$ *moves* to a

[1]Recall that G is assumed to have a unique sink vertex τ in Theorem 1.

configuration $C' := \langle\!\langle P', r', c' \rangle\!\rangle$ (denoted as $C \vdash C'$), if (1) $r = \bot$ and $r' \in V \setminus P$ (*Pebbler* moves in C and then *Challenger* moves in C'),[2] and the game is not over in C and $P' = P \cup \{r'\}$ and $c' = c$; or (2) $r \in P$ and $r' = \bot$ (*Challenger* moves in C and then *Pebbler* moves in C'), and $c' \in \{c, r\}$ and $P' = P$.

In the Dymond–Tompa game tree ($\textsc{GameTree}_G$) for DT_G, every node is labeled with a configuration. First construct the root node of $\textsc{GameTree}_G$, labeled with the initial configuration C_G. And for any node x labeled with C, for every C' such that $C \vdash C'$, construct a child node x' of x labeled with C'. The game tree is finite since *Pebbler* is required to pebble an unpebbled vertex.[2]

Definition 3.4.2 (Value of a (Sub)-Game)**.** For a node x on $\textsc{GameTree}_G$, define its value

$$\textsc{Val}(x) := \begin{cases} 1 & \text{if } x \text{ is a leaf node,} \\ \min_{x'\colon \text{ child of } x} \textsc{Val}(x') & \text{if } \textit{Pebbler} \text{ moves at internal node } x, \\ 1 + \max_{x'\colon \text{ child of } x} \textsc{Val}(x') & \text{if } \textit{Challenger} \text{ moves at internal node } x. \end{cases}$$

Then DT_G takes h time if $\textsc{Val}(\text{root of } \textsc{GameTree}_G) = h$.

Intuitively, an optimal game play should focus only on the effective predecessors $V_c(P)$ of the currently challenged vertex c (Definition 3.4.3). This is formalized as Lemma 3.4.5, by an induction on Lemma 3.4.6.

Definition 3.4.3 (Effective Predecessors)**.** Relative to any $S \subseteq V$, for vertices a and b in V, define the transitive relation $a \rightsquigarrow_S b$ if there is a directed path (possibly of zero length) from a to b avoiding S, i. e., there exists $\{v_0, v_1, \dots, v_\ell\} \subseteq V \setminus S$ such that $v_0 = a$ and $v_\ell = b$ and $v_i \in \delta^{\text{in}}(v_{i+1})$ for $0 \le i < \ell$. When c is under challenge and P are the pebbled vertices, define the (not necessarily proper) *effective predecessors* of c avoiding P as $V_c(P) := \{a \in V : a \rightsquigarrow_{(P \setminus \{c\})} c\}$.

Proposition 3.4.4 (Effective Predecessors)**.** *We have the following:*

1. $V_c(P) \cap P = \{c\}$ *when* $c \in P$;
2. *If* $a \in V_c(Q)$, *then* $V_a(Q) \subseteq V_c(Q)$;
3. *If* $c \in Q \subseteq R$, *then* $V_c(Q) \supseteq V_c(R)$; *and*
4. *If* $c \in Q \subseteq R$ *and* $(R \setminus Q) \cap V_c(Q) = \emptyset$, *then* $V_c(Q) = V_c(R)$.

[2]Note that $r' \in V \setminus P$ in item (1) in the definition of $\vdash$, i. e., *Pebbler* is required to pebble an unpebbled vertex. The game is effectively the same with or without this requirement, since *Challenger* can always rechallenge the last challenged vertex if *Pebbler* repebbles a pebbled vertex, hence an optimal *Pebbler* strategy should obey this requirement. This requirement is added here to avoid working with an infinite game tree, so as to simplify subsequent definitions while not affecting the values of subgames.

Lemma 3.4.5 (Predecessors Determine a Subgame)**.** *The value of a subgame depends only on the effective predecessors of the challenged vertex, i. e., if $V_c(Q) = V_c(R)$, then* $\textsc{Val}(\langle\!\langle Q, \bot, c\rangle\!\rangle) = \textsc{Val}(\langle\!\langle R, \bot, c\rangle\!\rangle)$.[3]

Proof. Let $P = Q \cup \big(V \setminus V_c(Q)\big) = R \cup \big(V \setminus V_c(R)\big)$, then $Q \subseteq P$ and $R \subseteq P$, now do an induction using Lemma 3.4.6 to show $\textsc{Val}(\langle\!\langle Q, \bot, c\rangle\!\rangle) = \textsc{Val}(\langle\!\langle P, \bot, c\rangle\!\rangle) = \textsc{Val}(\langle\!\langle R, \bot, c\rangle\!\rangle)$. More precisely, recall that for a subset $S \subseteq V$, a sink vertex s of S satisfies $s \in S$ and $\delta^{\text{out}}(s) \cap S = \emptyset$. Enumerate $P \setminus Q =: \{s_1, s_2, \ldots, s_\ell\} \subseteq V \setminus \big(Q \cup V_c(Q)\big)$ so that s_i is a sink of S_i where $S_\ell := P$ and $S_i := S_{i+1} \setminus \{s_{i+1}\}$ for $0 \leq i < \ell$, and apply Lemma 3.4.6 to get $\textsc{Val}(\langle\!\langle S_i, \bot, c\rangle\!\rangle) = \textsc{Val}(\langle\!\langle S_{i+1}, \bot, c\rangle\!\rangle)$. □

Lemma 3.4.6 (Predecessors Determine Adjacent Subgames)**.** *If $R = Q \cup \{q\}$ for some sink q of $V \setminus \big(Q \cup V_c(Q)\big)$, then* $\textsc{Val}(\langle\!\langle Q, \bot, c\rangle\!\rangle) = \textsc{Val}(\langle\!\langle R, \bot, c\rangle\!\rangle)$.[3]

Proof. Say two *Pebbler* configurations $C_1 := \langle\!\langle P_1, \bot, c_1\rangle\!\rangle$ and $C_2 := \langle\!\langle P_2, \bot, c_2\rangle\!\rangle$ form an *adjacent pair* (denoted $\langle C_1, C_2\rangle$) if $c_1 = c = c_2$ for some $c \in V$ and $P_2 = P_1 \cup \{q\}$ for some sink q of $V \setminus \big(P_1 \cup V_c(P_1)\big)$. In this case $V_c(P_1) = V_c(P_2)$ by Proposition 3.4.4. For two configurations C and C', say C is a *descendant* of C' (denoted $C \preceq C'$) if there are configurations $\{C_1, \ldots, C_\ell\}$, such that $C_{i+1} \vdash C_i$ for $1 \leq i < \ell$ and $C_1 = C$ and $C_\ell = C'$.[4] This partial order on configurations induces a partial order on adjacent pairs by $\langle C_1, C_2\rangle \preceq \langle C_1', C_2'\rangle$ if $C_1 \preceq C_1'$ and $C_2 \preceq C_2'$.

Do an induction following the $\preceq$ order on adjacent pairs $\langle Q, R\rangle$ to show that $\textsc{Val}(Q) = \textsc{Val}(R)$. When $V_c(Q) = V_c(R)$, note that $\delta^{\text{in}}(c) \subseteq Q$ iff $V_c(Q) = \{c\}$ iff $V_c(R) = \{c\}$ iff $\delta^{\text{in}}(c) \subseteq R$, i. e., the game is over in $\langle\!\langle Q, \bot, c\rangle\!\rangle$ iff it is over in $\langle\!\langle R, \bot, c\rangle\!\rangle$. If the game is over, then $\textsc{Val}(\langle\!\langle Q, \bot, c\rangle\!\rangle) = 1 = \textsc{Val}(\langle\!\langle R, \bot, c\rangle\!\rangle)$, establishing the base case. Otherwise, the game is not over. Expand and compare

$$\textsc{Val}(\langle\!\langle Q, \bot, c\rangle\!\rangle) = \min_{r \notin Q} \textsc{Val}(\langle\!\langle Q \cup \{r\}, r, c\rangle\!\rangle) \quad \text{and} \quad \textsc{Val}(\langle\!\langle R, \bot, c\rangle\!\rangle) = \min_{r \notin R} \textsc{Val}(\langle\!\langle R \cup \{r\}, r, c\rangle\!\rangle) \ .$$

For an $r \in V \setminus Q$, there are two cases.

- $r \notin R$: Note that $\langle \langle\!\langle Q \cup \{r\}, \bot, c\rangle\!\rangle, \langle\!\langle R \cup \{r\}, \bot, c\rangle\!\rangle \rangle \prec \langle \langle\!\langle Q, \bot, c\rangle\!\rangle, \langle\!\langle R, \bot, c\rangle\!\rangle \rangle$, and since q is a sink of $V \setminus \big(Q \cup V_c(Q)\big)$ and $q \notin V_r(Q \cup \{r\})$, we have $\langle \langle\!\langle Q \cup \{r\}, \bot, r\rangle\!\rangle, \langle\!\langle R \cup \{r\}, \bot, r\rangle\!\rangle \rangle \prec \langle \langle\!\langle Q, \bot, c\rangle\!\rangle, \langle\!\langle R, \bot, c\rangle\!\rangle \rangle$, hence induction hypothesis gives

$$\begin{aligned}\textsc{Val}(\langle\!\langle Q \cup \{r\}, r, c\rangle\!\rangle) &= 1 + \max\big\{\ \textsc{Val}(\langle\!\langle Q \cup \{r\}, \bot, r\rangle\!\rangle)\ ,\ \textsc{Val}(\langle\!\langle Q \cup \{r\}, \bot, c\rangle\!\rangle)\ \big\} \\ &= 1 + \max\big\{\ \textsc{Val}(\langle\!\langle R \cup \{r\}, \bot, r\rangle\!\rangle)\ ,\ \textsc{Val}(\langle\!\langle R \cup \{r\}, \bot, c\rangle\!\rangle)\ \big\} \\ &= \textsc{Val}(\langle\!\langle R \cup \{r\}, r, c\rangle\!\rangle)\ ;\end{aligned}$$

[3]Clearly the subtree rooted at (and hence the value of) a node x on $\textsc{GameTree}_G$ depends only on the configuration labeled at x, thus it makes sense to talk about the value of a configuration, although in general there can be multiple nodes on $\textsc{GameTree}_G$ labeled with the same configuration.

[4]Hence $C \preceq C'$ if some node labeled with C is a (not necessarily proper) descendant of some node labeled with C' on $\textsc{GameTree}_G$.

- $r = q \in R \setminus Q$: Then

$$\begin{aligned}\textsc{Val}(\langle\!\langle Q \cup \{q\}, q, c\rangle\!\rangle) &= \textsc{Val}(\langle\!\langle R, q, c\rangle\!\rangle)\\ &= 1 + \max\{\ \textsc{Val}(\langle\!\langle R, \bot, q\rangle\!\rangle)\ ,\ \ \textsc{Val}(\langle\!\langle R, \bot, c\rangle\!\rangle)\ \}\\ &> \textsc{Val}(\langle\!\langle R, \bot, c\rangle\!\rangle)\ .\end{aligned}$$

Now

$$\begin{aligned}\textsc{Val}(\langle\!\langle Q, \bot, c\rangle\!\rangle) &= \min_{r \notin Q} \textsc{Val}(\langle\!\langle Q \cup \{r\}, r, c\rangle\!\rangle)\\ &= \min\{\ \min_{r \notin R} \textsc{Val}(\langle\!\langle Q \cup \{r\}, r, c\rangle\!\rangle)\ ,\ \ \textsc{Val}(\langle\!\langle Q \cup \{q\}, q, c\rangle\!\rangle)\ \}\\ &= \textsc{Val}(\langle\!\langle R, \bot, c\rangle\!\rangle)\ . \qquad \square\end{aligned}$$

Since the game should only focus on the effective predecessors $V_c(P)$ of the currently challenged vertex c, an optimal game play should go from the sink to the sources of G (Claims 3.4.8 and 3.4.10, see Definition 3.4.7).

Definition 3.4.7 (Upstream Strategies)**.** Say a strategy for *Pebbler* is *upstream* if *Pebbler* only pebbles an effective predecessor of the currently challenged vertex, and a strategy for *Challenger* is *upstream* if *Challenger* only challenges an effective predecessor of the previously challenged vertex. More precisely, for configurations $C := \langle\!\langle P, r, c\rangle\!\rangle$ and $C' := \langle\!\langle P', r', c'\rangle\!\rangle$, say *$C$ moves upstream to C'* (denoted as $C \vdash_{\leftsquigarrow} C'$) iff $C \vdash C'$ and if (1) $r = \bot$ (*Pebbler* moves in C) then $r' \in V_c(P)$; or (2) $r \in P$ (*Challenger* moves in C) then $c' \in V_c(P)$. Then a *Pebbler* (resp. *Challenger*) strategy is upstream if every *Pebbler* (resp. *Challenger*) move from C to C' satisfies $C \vdash_{\leftsquigarrow} C'$.

Claim 3.4.8 (Optimal Upstream *Pebbler*)**.** *Any subgame-optimal Pebbler strategy is upstream, i. e., if configurations $C = \langle\!\langle P, \bot, c\rangle\!\rangle$ and C' satisfy $C \vdash C'$ and* $\textsc{Val}(C) = \textsc{Val}(C')$*,*[3] *then $C \vdash_{\leftsquigarrow} C'$.*

Proof. Consider the *Pebbler* move from $C =: \langle\!\langle P, \bot, c\rangle\!\rangle$ to $C' =: \langle\!\langle P \cup \{r\}, r, c\rangle\!\rangle$ where $r \notin P$ and $r \notin V_c(P)$ (hence $C \vdash C'$ but $C \not\vdash_{\leftsquigarrow} C'$), either (1) challenging r is no worse for *Challenger*, i. e., $C_r := \langle\!\langle P \cup \{r\}, \bot, r\rangle\!\rangle$ has $\textsc{Val}(C_r) \geq \textsc{Val}(C)$, then a subgame-optimal strategy of *Pebbler* would avoid the move from C to C' (since $\textsc{Val}(C') \geq 1 + \textsc{Val}(C_r) > \textsc{Val}(C)$); or (2) C_r is worse for *Challenger*, i. e., $\textsc{Val}(C_r) < \textsc{Val}(C)$, then *Challenger* may choose to rechallenge c by moving to $C_c := \langle\!\langle P \cup \{r\}, \bot, c\rangle\!\rangle$ so that $\textsc{Val}(C_c) = \textsc{Val}(C)$ (by Proposition 3.4.4 and Lemma 3.4.5), hence a subgame-optimal strategy of *Pebbler* would avoid the move from C to C' (since $\textsc{Val}(C') \geq 1 + \textsc{Val}(C_c) > \textsc{Val}(C)$). $\square$

Corollary 3.4.9 (Optimal Upstream *Pebbler*)**.** *If the game is not over in a Pebbler configuration $\langle\!\langle P, \bot, c\rangle\!\rangle$, then* $\textsc{Val}(\langle\!\langle P, \bot, c\rangle\!\rangle) = \min_{r \in V_c(P) \setminus P} \textsc{Val}(\langle\!\langle P \cup \{r\}, r, c\rangle\!\rangle)$.

Claim 3.4.10 (Optimal Upstream *Challenger*)**.** *There exists an optimal Challenger strategy that is upstream, i. e., if configurations $C = \langle\!\langle P, \bot, c\rangle\!\rangle$ and $C' = \langle\!\langle P \cup \{r\}, r, c\rangle\!\rangle$ satisfy $C \vdash C'$, then there is a Challenger move from C' to C'' with $C' \vdash_{\leftsquigarrow} C''$ and $\textsc{Val}(C'') \geq \textsc{Val}(C) - 1$.*

Proof. If $r \in V_c(P)$, then $C' \vdash C''$ implies $C' \vdash_{\leftsquigarrow} C''$. Now Definition 3.4.2 gives a C'' with $C' \vdash_{\leftsquigarrow} C''$ and $\textsc{Val}(C'') \geq \textsc{Val}(C') - 1 \geq \textsc{Val}(C) - 1$. Otherwise $r \notin V_c(P)$, then consider $C'' := \langle\!\langle P \cup \{r\}, \bot, c\rangle\!\rangle$. Proposition 3.4.4 and Lemma 3.4.5 give $\textsc{Val}(C'') = \textsc{Val}(C)$, and clearly $C' \vdash_{\leftsquigarrow} C''$. □

Proposition 3.4.11 (Upstream is Monotone)**.** *Consider $C_1 \vdash C_2 \vdash C_3$ where $C_1 =: \langle\!\langle P_1, \bot, c_1\rangle\!\rangle$ and $C_3 =: \langle\!\langle P_3, \bot, c_3\rangle\!\rangle$. If $C_1 \vdash_{\leftsquigarrow} C_2$ or $C_2 \vdash_{\leftsquigarrow} C_3$, then $c_3 \in V_{c_1}(P_1)$ and $V_{c_1}(P_1) \supseteq V_{c_3}(P_3)$.*

Raz–McKenzie Pebble Game

Definitions 3.4.12 and 3.4.13 formalize the intuitive Definition 3.2.1 for the Raz–McKenzie pebble game.

Definition 3.4.12 (Raz–McKenzie Game Tree)**.** Fix a DAG $G = (V, E)$. A *configuration* of the Raz–McKenzie game (RM_G) over G is a tuple $\langle\!\langle P, r, B\rangle\!\rangle$, where $P \subseteq V$ are the pebbled vertices, $r \in P \cup \{\bot\}$ is the vertex just pebbled, and $B \subset P$ are the blue vertices (and $P \setminus (B \cup \{r\})$ are the red vertices). The player taking the turn in $\langle\!\langle P, r, B\rangle\!\rangle$ is *Pebbler* if $r = \bot$, and is *Colorer* if $r \in P$.

The *initial configuration* for G is $C^{\mathsf{RM}}_G := \langle\!\langle \{\tau\}, \bot, \emptyset\rangle\!\rangle$,[1] and the game is *over* in a configuration $\langle\!\langle P, r, B\rangle\!\rangle$ if $r = \bot$ and some $d \in P \setminus B$ has $\delta^{\text{in}}(d) \subseteq B$. A configuration $C := \langle\!\langle P, r, B\rangle\!\rangle$ *moves* to a configuration $C' := \langle\!\langle P', r', B'\rangle\!\rangle$ (denoted as $C \vdash C'$), if (1) $r = \bot$ and $r' \in V \setminus P$ (*Pebbler* moves in C and then *Colorer* moves in C'),[5] and the game is not over in C and $P' = P \cup \{r'\}$ and $B' = B$; or (2) $r \in P$ and $r' = \bot$ (*Colorer* moves in C and then *Pebbler* moves in C'), and $B \subseteq B' \subseteq B \cup \{r\}$ and $P' = P$.

In the Raz–McKenzie game tree ($\textsc{GameTree}^{\mathsf{RM}}_G$) for RM_G, every node is labeled with a configuration. First construct the root node of $\textsc{GameTree}^{\mathsf{RM}}_G$, labeled with the initial configuration C^{RM}_G. And for any node x labeled with C, for every C' such that $C \vdash C'$, construct a child node x' of x labeled with C'. The game tree is finite since *Pebbler* is required to pebble an unpebbled vertex.[5]

[5]Note that $r' \in V \setminus P$ in item (1) in the definition of $\vdash$, i. e., *Pebbler* is required to pebble an unpebbled vertex. The game is effectively the same with or without this requirement, since *Colorer* can always recolor a vertex with its existing color if *Pebbler* repebbles a pebbled vertex, hence an optimal *Pebbler* strategy should obey this requirement. This requirement is added here to avoid working with an infinite game tree, so as to simplify subsequent definitions while not affecting the values of subgames.

Definition 3.4.13 (Value of a (Sub)-Game)**.** For a node x on $\textsc{GameTree}_G^{\mathsf{RM}}$, define its value

$$\textsc{Val}(x) := \begin{cases} 1 & \text{if } x \text{ is a leaf node,} \\ \min_{x' \text{: child of } x} \textsc{Val}(x') & \text{if } \textit{Pebbler} \text{ moves at internal node } x, \\ 1 + \max_{x' \text{: child of } x} \textsc{Val}(x') & \text{if } \textit{Colorer} \text{ moves at internal node } x. \end{cases}$$

Then RM_G takes h time if $\textsc{Val}(\text{root of } \textsc{GameTree}_G^{\mathsf{RM}}) = h$.

Dymond–Tompa equals Raz–McKenzie

Theorem 8 (Dymond–Tompa equals Raz–McKenzie)**.** *For any DAG G, DT_G takes h time iff RM_G takes h time.*

Proof. For the $\Leftarrow$ direction, given an optimal *Colorer* strategy for RM_G, we should construct a *Challenger* strategy for DT_G to make at least h moves. In each move, when c is under challenge, after *Pebbler* pebbles $r \in V \setminus P$ to a configuration $\langle\!\langle P \cup \{r\}, r, c \rangle\!\rangle$ in DT_G, *Challenger* (1) challenges r if $r \in V_c(P)ni$ and *Colorer* colors r red in response to *Pebbler*; and (2) rechallenges c otherwise. *Challenger* strategy maintains the invariant that c is the only red vertex among its effective predecessors, i. e., $V_c(P) \cap (P \setminus B) = \{c\}$ (by induction on *Challenger* moves). If the game DT_G is over in a configuration $\langle\!\langle P, \perp, c \rangle\!\rangle$, then $\delta^{\text{in}}(c) \subseteq P$. It follows that c is red but all of $\delta^{\text{in}}(c)$ are blue; for otherwise, some $r \in \delta^{\text{in}}(c)$ is red, but then r is colored red by *Colorer* in a round when some d is challenged, and both c and r are effective predecessors of d in that round (recall Proposition 3.4.11), contradicting the *Challenger* strategy. So the game RM_G is also over.

For the $\Rightarrow$ direction, given an optimal *Challenger* strategy for DT_G, we should construct a *Colorer* strategy for RM_G to make at least h moves. By Claim 3.4.10, assume that *Challenger* strategy is upstream. In each move, when c is under challenge, after *Pebbler* pebbles $r \in V \setminus P$ to configurations $\langle\!\langle P', r, c \rangle\!\rangle$ in DT_G and $\langle\!\langle P', r, B \rangle\!\rangle$ in RM_G with $P' = P \cup \{r\}$, if (1) *Challenger* responses to *Pebbler* by challenging $r \neq c$ (hence $r \in V_c(P)$), then *Colorer* colors r red; or (2) *Challenger* responses by rechallenging c, then *Colorer* (i) colors r blue unless there are red vertices $d, d' \in P \setminus B$ blocked by making r blue, i. e., $d \notin V_{d'}(B \cup \{r\})$ but $d \in V_{d'}(B)$; in which case (ii) colors r red. *Colorer* strategy maintains the invariant that c is the only red vertex among its effective predecessors, i. e., $V_c(P) \cap (P \setminus B) = \{c\}$; and there is a blue-avoiding path covering all red vertices, i. e., for any $d, d' \in P \setminus B$, $d \in V_{d'}(B)$ or $d' \in V_d(B)$ (by induction on *Colorer* moves). As a result, c is the first red vertex in this blue-avoiding path, i. e., $c \in V_d(B)$ for any $d \in P \setminus B$. If the game RM_G is over in a configuration $\langle\!\langle P, \perp, B \rangle\!\rangle$, then $\delta^{\text{in}}(d) \subseteq B$ for some $d \in P \setminus B$, thus $d = c$ is the vertex under challenge and $\delta^{\text{in}}(c) \subseteq P$, so the game DT_G is also over. $\square$

3.5 When Raz–McKenzie meet Bennett

The Raz–McKenzie pebble game (§ 3.5) is connected with the reversible pebble game of Bennett (§ 3.5) by a simulation argument in § 3.5.

Reformulating Raz–McKenzie Pebble Game

Focusing on the *Pebbler* side of the Raz–McKenzie pebble game and interpreting it as a one-player game, Definition 3.5.1 and Proposition 3.5.2 bring the Raz–McKenzie (two-player) pebble game to a form closer to the (one-player) reversible pebble game.

Definition 3.5.1 (Reduced Configuration)**.** Fix a DAG $G = (V, E)$. A *reduced configuration* of the Raz–McKenzie pebble game (RM_G) over G is a pair $(\!|B, R|\!)$ of blue B and red R vertices ($B, R \subseteq V$) which are disjoint $B \cap R = \{\}$. Any reduced configuration $(\!|B, R|\!)$ corresponds to the *Pebbler* configuration $\langle\!\langle R \cup B, \bot, B \rangle\!\rangle$.

Proposition 3.5.2 (Value of Reduced Configuration)**.**

$$\textsc{Val}((\!|B, R|\!)) = \begin{cases} 1 & \textit{if } \exists r \in R \textit{ s.t. } \delta^{in}(r) \subseteq B, \\ 1 + \min\limits_{v \in V \setminus (R \cup B)} \max \begin{Bmatrix} \textsc{Val}((\!|B, R \cup \{v\}|\!)), \\ \textsc{Val}((\!|B \cup \{v\}, R|\!)) \end{Bmatrix} & \textit{otherwise.} \end{cases}$$

Proposition 3.5.3 (Monotonicity)**.** *If $B_1 \subseteq B_2$ and $R_1 \subseteq R_2$, then* $\textsc{Val}((\!|B_1, R_1|\!)) \geq \textsc{Val}((\!|B_2, R_2|\!))$.

Reversible Pebble Game

Following its usage in proof complexity [4, 12, 51, 98], the (reversible) black pebble game is parameterized below with two extra sets of vertices S (extending the sources) and T (extending the sinks) in Definition 3.5.4. By changing S and T as the induction step goes, and by focusing on the progress in pebbling outside of S and T, this parameterization sets up the right recurrence in its translation to and from the Raz–McKenzie pebble game (Lemmas 3.5.6 and 3.5.9).

Definition 3.5.4 (Reversible Pebble Game)**.** Fix a DAG $G = (V, E)$ and two vertex subsets $S, T \subseteq V$ which are disjoint $S \cap T = \{\}$. A *configuration* P in the reversible pebble game ($\mathsf{RP}_{G,S,T}$) is a subset of pebbled vertices $P \subseteq V$. Two configurations P_1 and P_2 are *adjacent* in $\mathsf{RP}_{G,S,T}$ if P_1 and P_2 differ by at most one vertex $v \in V$, all of whose in-neighbors are pebbled or in S, i. e., $P_1 \Delta P_2 \subseteq \{v\}$ where $\delta^{\text{in}}(v) \subseteq P_1 \cup S$ for some $v \in V$ (in this case, equivalently $\delta^{\text{in}}(v) \subseteq P_2 \cup S$). Note that all of S are virtually pebbled, hence referred to as *assuming* S. Say a configuration P *precisely pebbles* a vertex in $T \subseteq V$ *assuming* $S \subseteq V$ if $P \setminus S = \{t\}$ for some vertex $t \in T$. For two configurations P_s and P_t, a *reversible (pebbling) strategy* $\mathcal{P} = \langle P_1, P_2, \ldots, P_\ell \rangle$ *from P_s to P_t in $\mathsf{RP}_{G,S,T}$* is a sequence of adjacent configurations, i. e.,

P_{j-1} is adjacent to P_j assuming S for $1 < j \leq \ell$, such that $P_1 = P_s$ and $P_\ell = P_t$. A *reversible (pebbling) strategy for $\mathsf{RP}_{G,S,T}$* is a reversible strategy from $\{\}$ to some P_t in $\mathsf{RP}_{G,S,T}$ where P_t precisely pebbles a vertex in T assuming S.

Definition 3.5.5 (Value of a Configuration)**.** The value of a configuration P is $\textsc{Val}(P) := |P|$ the number of pebbles in P. The value of a reversible strategy $\mathcal{P} := \langle P_1, P_2, \ldots, P_\ell \rangle$ is $\textsc{Val}(\mathcal{P}) := \max_{1 \leq j \leq \ell} \textsc{Val}(P_j)$. The value of the reversible pebble game $\mathsf{RP}_{G,S,T}$ is $\textsc{Val}(\mathsf{RP}_{G,S,T}) := \min_{\mathcal{P}} \textsc{Val}(\mathcal{P})$, where the minimum is over all reversible strategy $\mathcal{P}$ for $\mathsf{RP}_{G,S,T}$ (from $\{\}$ to precisely pebble some vertex in T assuming S in $\mathsf{RP}_{G,S,T}$).

Raz–McKenzie equals Bennett

Lemma 3.5.6 (Reversible Strategy from Raz–McKenzie Strategy, Induction)**.** *There is a reversible strategy $\mathcal{P}$ for $\mathsf{RP}_{G,B,R}$ of value* $\textsc{Val}(\mathcal{P}) \leq \textsc{Val}((\!|B, R|\!)) =: h$.

Proof. If $h = 1$, then some vertex $r \in R$ has all its in-neighbors $\delta^{\text{in}}(r) \subseteq B$. Now pebble $r \in R$ assuming B, i. e., $\mathcal{P} := \langle \{\}, \{r\} \rangle$, establishing the base case.

If $h > 1$, fix $v \in V \setminus (R \cup B)$ such that

$$\textsc{Val}((\!|B, R|\!)) = 1 + \max\big\{\textsc{Val}((\!|B, R \cup \{v\}|\!)), \textsc{Val}((\!|B \cup \{v\}, R|\!))\big\} \ .$$

Since $\textsc{Val}((\!|B, R \cup \{v\}|\!)) < h$, there is a reversible strategy $\mathcal{P}_1 =: \langle P_1, P_2, \ldots, P_\ell \rangle$ for $\mathsf{RP}_{G,B,R\cup\{v\}}$ (i. e., assuming B to precisely pebble a vertex in $R \cup \{v\}$) of value $\textsc{Val}(\mathcal{P}_1) < h$. If a vertex in R is precisely pebbled assuming B ($P_\ell \setminus B = \{r\}$ for some $r \in R$), then we are done $\mathcal{P} := \mathcal{P}_1$. Otherwise, v is precisely pebbled assuming B ($P_\ell \setminus B = \{v\}$). Since $\textsc{Val}((\!|B \cup \{v\}, R|\!)) < h$, there is a reversible strategy $\mathcal{P}_2$ for $\mathsf{RP}_{G,B\cup\{v\},R}$ (i. e., assuming $B \cup \{v\}$ to precisely pebble a vertex $r \in R$) of value $\textsc{Val}(\mathcal{P}_2) < h$. Hence run $\mathcal{P}_1$, then run $\mathcal{P}_2$, and finally run $\mathcal{P}_1$ in reverse to forget v. That is, let $\mathcal{P} :=$ the concatenation of $\mathcal{P}_1$, $\mathcal{P}_2 \cup \{v\}$, and $\mathcal{P}_{\overleftarrow{1}} \cup \{r\}$; where $\mathcal{P}_{\overleftarrow{1}} := \langle P_\ell, P_{\ell-1}, \ldots, P_1 \rangle$ reverses $\mathcal{P}_1$, and $\mathcal{P}_1 \cup \{r\} := \langle P_1 \cup \{r\}, P_2 \cup \{r\}, \ldots, P_\ell \cup \{r\} \rangle$ denotes the configuration-wise union. Note that $\mathcal{P}$ is a strategy from $\{\}$ to precisely pebble $r \in R$ assuming B. □

Lemma 3.5.7 (Raz–McKenzie Strategy from Reversible Strategy)**.** *Any reversible strategy $\mathcal{P} =: \langle P_1, P_2, \ldots, P_\ell \rangle$ for $\mathsf{RP}_{G,B,R}$ has value* $\textsc{Val}(\mathcal{P}) \geq \textsc{Val}((\!|B, R|\!))$.

Proof. Let $r \in R$ be precisely pebbled assuming B, i. e., $P_\ell \setminus B =: \{r\}$. Without loss of generality $\delta^{\text{in}}(r) \cap R = \emptyset$, for otherwise replace every configuration P_j containing r with $P_j \setminus \{r\} \cup \{r'\}$ for some predecessor r' of r such that $\delta^{\text{in}}(r') \cap R = \emptyset$. Let m be the first time (i. e., least integer) such that r is pebbled since P_m, i. e., $P_b \ni r$ for $m \leq b \leq \ell$. Since $P_1 = \{\}$, $m > 1$. So P_{m-1} differs from P_m by a reversible pebble move to pebble $r \in R \cap P_m$ assuming B. Thus $\delta^{\text{in}}(r) \subseteq P_m \cup B$. Let $\mathcal{P}_1 := \langle P_m, P_{m+1}, \ldots, P_\ell \rangle$ be the strategy since P_m, and $\mathcal{P}_{\overleftarrow{1}} := \langle P_\ell, P_{\ell-1}, \ldots, P_m \rangle$ be its reverse. Note that $\delta^{\text{in}}(r) \subseteq \big(P_m \cap V_R(B)\big) \cup B$ (see Definition 3.5.8). Apply Lemma 3.5.9 on $\mathcal{P}_{\overleftarrow{1}} \cap V_R(B)$, where $\mathcal{P}_{\overleftarrow{1}} \cap V_R(B) := \langle P_\ell \cap$

$V_R(B), P_{\ell-1} \cap V_R(B), \ldots, P_m \cap V_R(B)\rangle$ is its configuration-wise intersection with $V_R(B)$, to get P_b with $m \leq b \leq \ell$ satisfying the second inequality in

$$|P_b| \geq \big|\big(P_b \cap V_R(B)\big) \setminus B\big| + 1 \geq \textsc{Val}(\llparenthesis B, R \rrparenthesis),$$

since $P_\ell \cap V_R(B) = \emptyset$ (thus $\tilde{P} = B$ in Lemma 3.5.9), and $r \in P_b \setminus V_R(B)$ gives the first inequality. $\square$

The following is a pebbling argument by induction, with a twist in using the right 'potential function' and the correct order for induction. First, since we are interested in pebbling R, it suffices to restrict attention to predecessors of R in the pebbling strategy. Moreover, since B is assumed, further restrict attention to those pebbling moves outside of B. The region of interest is denoted $V_R(B)$ below. The induction step is applied to the pebbling move P_m where the first vertex (denoted v below) is remembered till the end (i. e., P_ℓ) in the region $V_R(B)$ of interest. By further restricting attention to $V \setminus \{v\}$ (in the Pebbling Case below) or to $V_v(v)$ (in the Unpebbling Case below), it ensures the technical condition that B and R are disjoint when applying the induction hypothesis (as witnessed by the support of a strategy).

Definition 3.5.8 (Predecessors, Support)**.** Fix a DAG $G = (V, E)$. Say $u \in V$ is a (not necessarily proper) *predecessor* of $v \in V$ if there is a directed path (possibly of zero length) from u to v.[6] Denote the predecessors by $V_v := \{u\colon u \text{ is a predecessor of } v\}$ for $v \in V$, and $V_R := \big(\bigcup_{r\in R} V_r\big)$ for $R \subseteq V$. Define the *predecessors of R relative to B* as $V_R(B) := V_R \setminus (R \cup B)$. As a shorthand, denote $V_v(v) := V_{\{v\}}(\{v\})$ as the *proper* predecessors of v.

For $U \subseteq V$, say a configuration P is *U-supported* if $P \subseteq U$, and say a strategy $\mathcal{P} := \langle P_1, P_2, \ldots, P_\ell\rangle$ is *U-supported* if each P_j is U-supported for $1 \leq j \leq \ell$.

Lemma 3.5.9 (Raz–McKenzie Strategy from Reversible Strategy, Induction)**.** *Any $V_R(B)$-supported reversible strategy $\mathcal{P} =: \langle P_1, P_2, \ldots, P_\ell\rangle$ in $\mathsf{RP}_{G,B,R}$ where $\delta^{in}(r) \subseteq P_\ell \cup B$ for some $r \in R$, has a configuration P_b for some $1 \leq b \leq \ell$, so that $\big|P_b \setminus \tilde{P}\big| + 1 \geq \textsc{Val}(\llparenthesis \tilde{P}, R \rrparenthesis)$, where $\tilde{P} := \big(\bigcap_{1\leq j\leq \ell} P_j\big) \cup B$.*

Proof. Decrease ℓ if necessary, let ℓ be the first time on $\mathcal{P}$ (i. e., least integer) so that $\delta^{\text{in}}(r) \subseteq P_\ell \cup B$ for some $r \in R$. If $\delta^{\text{in}}(r) \subseteq \tilde{P}$, then $\textsc{Val}(\llparenthesis \tilde{P}, R \rrparenthesis) = 1$, so any configuration on $\mathcal{P}$ would do. Otherwise, $\delta^{\text{in}}(r) \not\subseteq \tilde{P}$. Let $\tilde{P}_i := \big(\bigcap_{i\leq j\leq \ell} P_j\big) \cup B$ be the set of vertices remembered since configuration P_i assuming B. Now $\delta^{\text{in}}(r) \not\subseteq \tilde{P} = \tilde{P}_1$ and $\delta^{\text{in}}(r) \subseteq \tilde{P}_\ell$, and $\tilde{P}_{j-1} \subseteq \tilde{P}_j$ for $1 < j \leq \ell$. Let $m := \operatorname{argmin}\{1 < j \leq \ell\colon \tilde{P}_1 \neq \tilde{P}_j\}$ indexes the earliest configuration so that $\tilde{P}_{m-1} \subset \tilde{P}_m$, and let $v \in \tilde{P}_m \setminus \tilde{P}_{m-1} = \tilde{P}_m \setminus \tilde{P}_1$ be the first vertex remembered till the end. Let $\mathcal{P}_1 := \langle P_m, P_{m+1}, \ldots, P_\ell\rangle$ be the strategy since P_m, which is shorter than $\mathcal{P}$.[7] Now for

[6]Hence the relation of predecessor is the reflexive transitive closure of the relation of immediate predecessor (in-neighbor).

[7]Formally, the double induction argument does an outer induction on the length of $\mathcal{P}$, then an inner induction on $\textsc{Val}(\llparenthesis \tilde{P}, R \rrparenthesis)$.

any P_b on $\mathcal{P}_1$ (i. e., $m \le b \le \ell$),

$$|P_b \setminus \tilde{P}_1| \ge |P_b \setminus \tilde{P}_m| + 1 \ , \tag{3.1}$$

since $v \in \big(P_b \setminus \tilde{P}_1\big) \setminus \big(P_b \setminus \tilde{P}_m\big)$. Note that $\tilde{P}_m = \tilde{P}_1 \cup \{v\} = \tilde{P} \cup \{v\}$.

Clearly $v \notin \tilde{P}$. Also, v is pebbled by a reversible pebble move at P_m, hence $\delta^{\text{in}}(v) \subseteq P_m \cup B$. It follows that $v \notin R$; for otherwise $v \in R$, either it contradicts the minimality of ℓ, or some vertex before v is pebbled till v is pebbled, contradicting the minimality of m. By the recurrence of $\textsc{Val}(\langle\!|\tilde{P}, R|\!\rangle)$ (Proposition 3.5.2), at least one of the following is true.

- (Pebbling Case)

$$\textsc{Val}(\langle\!|\tilde{P} \cup \{v\}, R|\!\rangle) + 1 \ge \textsc{Val}(\langle\!|\tilde{P}, R|\!\rangle) \ . \tag{3.2}$$

  Note that $V_R(B \cup \{v\}) = V_R(B) \setminus \{v\}$. Hence $\mathcal{P}_2 := \mathcal{P}_1 \setminus \{v\} := \langle P_m \setminus \{v\}, P_{m+1} \setminus \{v\}, \dots, P_\ell \setminus \{v\}\rangle$ is $V_R(B \cup \{v\})$-supported. Now $\tilde{P}_m = \big(\bigcap_{m \le j \le \ell} P_j \setminus \{v\}\big) \cup B \cup \{v\}$. The induction hypothesis on $\mathcal{P}_2$ in $\mathsf{RP}_{G,B\cup\{v\},R}$ gives a P_b (on $\mathcal{P}_1$) satisfying the inequality in

$$|P_b \setminus \tilde{P}_m| + 1 = \big|\big(P_b \setminus \{v\}\big) \setminus \tilde{P}_m\big| + 1 \ge \textsc{Val}(\langle\!|\tilde{P}_m, R|\!\rangle) = \textsc{Val}(\langle\!|\tilde{P} \cup \{v\}, R|\!\rangle) \ . \tag{3.3}$$

  Finally $|P_b \setminus \tilde{P}| + 1 \ge \textsc{Val}(\langle\!|\tilde{P}, R|\!\rangle)$ by Inequalities 3.1 to 3.3.

- (Unpebbling Case)

$$\textsc{Val}(\langle\!|\tilde{P}, R \cup \{v\}|\!\rangle) + 1 \ge \textsc{Val}(\langle\!|\tilde{P}, R|\!\rangle) \ . \tag{3.4}$$

  In fact, $\delta^{\text{in}}(v) \subseteq (P_m \cap V_v(v)) \cup B$. Let $\mathcal{P}_{\bar{1}} := \langle P_\ell, P_{\ell-1}, \dots, P_m\rangle$ be the reverse of $\mathcal{P}_1$, and $\mathcal{P}_2 := \mathcal{P}_{\bar{1}} \cap V_v(v) := \langle P_\ell \cap V_v(v), P_{\ell-1} \cap V_v(v), \dots, P_m \cap V_v(v)\rangle$ be its configuration-wise intersection. Then $\mathcal{P}_2$ is $V_{R\cup\{v\}}(B)$-supported, and also $V_v(v)$-supported. Let $\tilde{P}' := \big(\bigcap_{m \le j \le \ell} P_j \cap V_v(v)\big) \cup B \subseteq \tilde{P}_m \setminus \{v\} = \tilde{P}$. The induction hypothesis on $\mathcal{P}_2$ in $\mathsf{RP}_{G,B,R\cup\{v\}}$ gives a P_b (on $\mathcal{P}_1$) satisfying the first inequality in

$$|P_b \cap V_v(v) \setminus \tilde{P}'| + 1 \ge \textsc{Val}(\langle\!|\tilde{P}', R \cup \{v\}|\!\rangle) \ge \textsc{Val}(\langle\!|\tilde{P}, R \cup \{v\}|\!\rangle) \ , \tag{3.5}$$

  where the last inequality follows from monotonicity (Proposition 3.5.3). Note that $P_b \cap V_v(v) \setminus \tilde{P}' \subseteq P_b \setminus \tilde{P}_m$, since $\tilde{P}_m \cap V_v(v) \subseteq \tilde{P}'$. Finally $|P_b \setminus \tilde{P}| + 1 \ge \textsc{Val}(\langle\!|\tilde{P}, R|\!\rangle)$ by Inequalities 3.1, 3.4 and 3.5. □

Corollary 3.5.10 (Raz–McKenzie equals Bennett). *For any DAG G, subsets $R, B \subseteq V$ which are disjoint $R \cap B = \{\}$, we have* $\textsc{Val}(\langle\!|B, R|\!\rangle) = \textsc{Val}(\mathsf{RP}_{G,B,R})$.

Proof. By Lemmas 3.5.6 and 3.5.7. □

Theorem 9 (Raz–McKenzie equals Bennett). *For any DAG G with a unique sink τ, we have* $\textsc{Val}(\langle\!|\{\}, \{\tau\}|\!\rangle) = \textsc{Val}(\mathsf{RP}_{G,\{\},\{\tau\}})$.

Chapter 4

DAG Evaluation Problem

This chapter studies the DAG evaluation problem. We define below the computational problem BDEP^k_G, the boolean version of the DAG evaluation problem of bit-length k over G. § 4.1 recalls the two-party communication game of Karchmer–Wigderson, § 4.2 introduces two classes of circuits with restricted computational semantics for BDEP^k_G, § 4.3 proves an upper bound as Theorem 11, § 4.4 connects the two-player pebble game of Raz–McKenzie with the two-party communication game of Karchmer–Wigderson, and § 4.5 proves a lower bound as Theorem 12.

The following computational problem naturally generalizes the Tree Evaluation Problem [33] to any directed acyclic graph G. This problem can be seen as a parameterized version of the P-complete circuit evaluation problem. By studying a slice of the problem (for a fixed graph G and constant k), we can focus on the combinatorics of the 'flow of values' over the graph.

Definition 4.0.11 (DAG Evaluation Problem over G)**.** Consider a DAG G and a bit-length parameter $k \in \mathbb{N}$. Denote the set of k-bit strings as $\{0,1\}^k \cong [K]$, where $K := 2^k$. The DAG Evaluation Problem over G (DEP^k_G) is specified by the following.

Input For every vertex $a \in V$, there is a function $t_a\colon [K]^{\delta^{\text{in}}(a)} \to [K]$.[1] The input to DEP^k_G enumerates the n bits of $\langle t_a \rangle_{a \in V}$ as n boolean variables where $n := k \sum_{a \in V} K^{\deg^{\text{in}}(a)}$.

'Computation' Define inductively the values $\langle v_a \rangle_{a \in V} \in [K]^V$ by $v_a := t_a\big(v\restriction_{\delta^{\text{in}}(a)}\big) \in [K]$ for $a \in V$. That is, the value v_a is the function t_a applied to the values at the in-neighbors of a.[1]

Output The output of DEP^k_G is the tuple of values $\langle v_w \rangle_{w \in W} \in [K]^W$.

Using terminologies of database systems, at every vertex $a \in V$, there is a table t_a whose dimension is the number of in-neighbors of a. The values at in-neighbors of a indexes the relevant entry in t_a, and we are interested in computing the values at the sinks.

[1]Note that for a source vertex $a \in U$, its function t_a degenerates to have a domain of $[K]^{\emptyset}$, hence the function $t_a \in [K]$ can be treated as a k-bit string. Thus its value v_a is just its function $t_a \in 2^{[K]}$ treated as a k bit-string.

Henceforth, without loss of generality, focus on DAGs with exactly one sink vertex τ. The interest is in the boolean circuit depth complexity of computing a decision version of DEP_G^k (as opposed to a $[K]$-valued function).

Definition 4.0.12 (Boolean DAG Evaluation Problem)**.** Fix a non-constant boolean function σ on k-bit strings $\sigma\colon \{0,1\}^k \to \{0,1\}$, say the zeroth bit $\sigma(s) := s\restriction_0$ for $s \in \{0,1\}^k$.[2] The Boolean DAG Evaluation Problem (BDEP_G^k) seeks to compute $\sigma(v_\tau)$.

4.1 Karchmer–Wigderson Game

Boolean circuit depth complexity is studied here via the (co-operative) communication game of Karchmer and Wigderson [59]. Recall that given a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ with promises $Y \subseteq f^{-1}(1)$ and $N \subseteq f^{-1}(0)$, the Karchmer–Wigderson game ($\mathsf{KW}_{Y,N}$) is a communication game between two parties defined as follows: Party 1 (the Yes party) is given a promised Yes instance $x \in Y$, Party 0 (the No party) is given a promised No instance $y \in N$, and they communicate to locate a bit position $i \in [n]$ where the inputs differ (i. e., $x_i \neq y_i$). (So the communication protocols are computing relations rather than functions.) And the Karchmer–Wigderson game for a boolean function f is $\mathsf{KW}_f := \mathsf{KW}_{f^{-1}(1),f^{-1}(0)}$. Karchmer and Wigderson observed that the communication complexity captures exactly the circuit depth.[3]

Theorem 10 (Circuit is a Protocol)**.** *The depth complexity of f on boolean circuits is exactly the communication complexity of KW_f.*

Notation 4.1.1 (Admissible Inputs)**.** Consider the protocol Π (as a rooted binary tree) and a node $g \in \Pi$. Denote $f_g^{-1}(1)$ as the set of inputs that can be given to Party 1 (the Yes party) at g and $f_g^{-1}(0)$ as the set of inputs that can be given to Party 0 (the No party). That is, the combinatorial rectangle associated with the node g is $f_g^{-1}(1) \times f_g^{-1}(0)$.

Notation 4.1.2 (Output Node)**.** Given an instance $(x, y) \in f^{-1}(1) \times f^{-1}(0)$ and a protocol Π, denote $\Pi(x, y)$ as the output node (rather than just the value) after running the protocol Π on the instance.

4.2 Thrifty and Output-Relevant Circuits

This subsection introduces two families of circuits with restricted computational semantics for BDEP_G^k: thrifty circuits and output-relevant circuits.

When concerning depth complexity, a circuit can be assumed to be a formula without loss of generality. Then a boolean formula $\mathcal{C}$ is isomorphic to a corresponding communication

[2] All non-constant boolean functions are equivalent with respect to the (restricted) lower bounds in this work. The zeroth bit is chosen here since its computation is trivial, i. e., takes no extra depth.

[3] Also observed independently by Yannakakis and was implicit in [61], see [59].

protocol Π (denoted $\mathcal{C} \equiv \Pi$) not only graph-theoretically (as a rooted binary tree), but also *computationally* (subsets of Yes and No instances match the combinatorial rectangles, i. e., for every $g \in \mathcal{C} \equiv \Pi$ under the graph isomorphism, any input $x \in f_g^{-1}(1)$ evaluates to 1 at gate $g \in \mathcal{C}$ and any input $y \in f_g^{-1}(0)$ evaluates to 0 at gate g). Therefore certain computational notions for a formula (or a circuit) $\mathcal{C}$ can equivalently be defined over the communication protocol Π under the Karchmer–Wigderson correspondence $\equiv$, as is done below for the notions of thrifty circuits and output-relevant circuits.

Intuitively, a circuit for BDEP_G^k is thrifty if its computation depends only on the values v_a, but not on other irrelevant bits (variables) of the functions t_a (Definition 4.2.3), as an analogue of thrifty branching programs [33]; and a circuit for BDEP_G^k is output-relevant if it only outputs relevant bits (variables), similar to the players who only output leaves of the universal composition relation [58, §6]. Note that after taking away the output-relevant restriction, a communication game for BDEP_G^k is a proper Karchmer–Wigderson game (so that it corresponds properly to circuit depth). This is *not* the case for the universal composition relation.

Notation 4.2.1 (Values)**.** For an input $x \in \{0,1\}^n$ to BDEP_G^k, denote $x\restriction_{v_a}$ as the v_a value of x (see Definition 4.0.11). As a shorthand, write $x\restriction_a$ for $x\restriction_{v_a}$, and $x\restriction_S$ for $\langle x\restriction_a\rangle_{a\in S}$ when $S \subseteq V$.

Definition 4.2.2 (Thrifty Protocols and Circuits)**.** A protocol Π for $\mathsf{KW}_{Y,N}$ is *thrifty* where $Y \subseteq f^{-1}(1)$ and $N \subseteq f^{-1}(0)$ for $f := \mathsf{BDEP}_G^k$, if for any pair of promised Yes instances $x, x' \in Y$, and any pair of promised No instances $y, y' \in N$, such that $x\restriction_V = x'\restriction_V$ and $y\restriction_V = y'\restriction_V$, we have $\Pi(x,y) = \Pi(x',y')$. A circuit $\mathcal{C}$ for f is *thrifty*, if there is a thrifty protocol Π for KW_f isomorphic to (the formula equivalent to) $\mathcal{C}$ (i. e., $\mathcal{C} \equiv \Pi$).

Definition 4.2.3 (Relevant Bits)**.** For an input $x \in \{0,1\}^n$ to BDEP_G^k, an input bit (variable) is *relevant* to x if (1) it is a variable specifying the $\langle v_{a'}\rangle_{a'\in\delta^{\text{in}}(a)}$ entry of t_a for a vertex $a \in V$; or equivalently (2) $x'\restriction_V \neq x\restriction_V$ where x' and x differ only on that bit.

Definition 4.2.4 (Output-Relevant Protocols and Circuits)**.** A protocol Π for $\mathsf{KW}_{Y,N}$ is *output-relevant* where $Y \subseteq f^{-1}(1)$ and $N \subseteq f^{-1}(0)$ for $f := \mathsf{BDEP}_G^k$, if for any $(x,y) \in Y \times N$, the node $\Pi(x,y)$ outputs a bit (position) relevant to x and relevant to y. A circuit $\mathcal{C}$ for f is *output-relevant*, if there is an output-relevant protocol Π for KW_f isomorphic to (the formula equivalent to) $\mathcal{C}$ (i. e., $\mathcal{C} \equiv \Pi$).

Remark 4.2.5 (Relevant Outputs as Certificates). Recall that the depth of a decision tree depends on the certificate complexity, where a *certificate* for a particular input $x \in \{0,1\}^n$ is a subset of bits of x sufficient to witness the membership/non-membership of x in a language. For both the Dymond–Tompa game (in particular the interpreted variant [99]) and the Karchmer–Wigderson game [59], it is of interest to efficiently pack certificates (for different Yes/No-instances) into (the leaves of) a shallow 'winning strategy' or (the output nodes of) a shallow protocol. And (the alternation in) the minimization of depth in both games can be modeled by two competing provers, who present bits of the certificates to witness membership/non-membership.

Specializing to the computational problem of BDEP^k_G, an efficient certificate for a particular input $x \in \{0,1\}^n$ should contain precisely the bits of the values relevant to x (at least when k is large, because a certificate containing a full row or column in a table is expensive). This combinatorial consideration motivates the definition of output-relevant circuits.

Proposition 4.2.6 (Thrifty is Relevant). *For $f := \mathsf{BDEP}^k_G$ and $Y \times N \subseteq f^{-1}(1) \times f^{-1}(0)$, a correct protocol Π for $\mathsf{KW}_{Y,N}$ (and hence a correct circuit $\mathcal{C}$ for f), if thrifty, is output-relevant.*

Proof. If Π for $\mathsf{KW}_{Y,N}$ is *not* output-relevant, there is an instance $(x,y) \in Y \times N$ such that $\Pi(x,y)$ outputs a bit position $i \in [n]$ not relevant to (say) x. Flip that bit in x to get x', then $x'_i \neq x_i$ and $x{\restriction_V} = x'{\restriction_V}$. If Π is thrifty, $\Pi(x,y) = \Pi(x',y)$, but then the protocol is incorrect on the instance (x,y) or on (x',y), since either $x_i = y_i$ or $x'_i = y_i$. □

4.3 Upper Bound for Evaluation

Theorem 11 implements a strategy for the Dymond–Tompa game DT_G as a circuit for the evaluation problem BDEP^k_G.

Theorem 11 (Upper Bound for Evaluation). *For any directed acyclic graph G whose Dymond–Tompa game takes h time, there is a (uniform) thrifty circuit $\mathcal{C}$ computing BDEP^k_G of depth $(h-1)\big(k + \lceil \log_2(k+1) \rceil\big) = O(hk)$.*

Proof. Apply Lemma 4.3.2 on $\langle\!\langle \{\tau\}, \perp, \tau \rangle\!\rangle$, $\alpha \in [K]^{\emptyset}$, $j = 0$, $b = 1$. □

Definition 4.3.1 (Bit Equality). Let $\beta\colon \{0,1\}^k \times [k] \times \{0,1\} \to \{0,1\}$ be the *bit-equality* function $\beta(z,j,b) := z{\restriction_j} \oplus b \oplus 1$, where $\oplus$ denotes addition mod 2.

We recall Definitions 3.4.1 and 3.4.2 and Footnote 3, from § 3.4.

Lemma 4.3.2 (Upper Bound for Evaluation, Induction). *For any configuration $\langle\!\langle P, \perp, c \rangle\!\rangle$ of DT_G with $\textsc{Val}(\langle\!\langle P, \perp, c \rangle\!\rangle) =: h$, any values $\alpha \in [K]^{P \setminus \{c\}}$ on $P \setminus \{c\}$, any $j \in [k]$, $b \in \{0,1\}$, there is a (uniform) thrifty circuit $\mathcal{C}$ for BDEP^k_G of depth $(h-1)\big(k + \lceil \log_2(k+1) \rceil\big)$, so that any $x \in \{0,1\}^n$ with $x{\restriction_{P \setminus \{c\}}} = \alpha$ satisfies $\mathcal{C}(x) = \beta(x{\restriction_{v_c}}, j, b)$.*

Proof. Since negation does not increase depth, assume $b = 1$. If $h = 1$, then $\delta^{\text{in}}(c) \subseteq P$, hence α contains all v_a for $a \in \delta^{\text{in}}(c)$. Now the input gate at the jth position of the $\alpha{\restriction_{\delta^{\text{in}}(c)}}$ entry of t_a is a circuit $\mathcal{C}$ of depth zero satisfying the conditions, establishing the base case. If $h > 1$, let $r \in V_c(P) \setminus P$ be such that $\max\{\textsc{Val}(C_L), \textsc{Val}(C_R)\} < h$ where $C_L := \langle\!\langle P', \perp, r \rangle\!\rangle$ and $C_R := \langle\!\langle P', \perp, c \rangle\!\rangle$ for $P' := P \cup \{r\}$ (Corollary 3.4.9). Let $C_\Lambda := \langle\!\langle P' \setminus \{c\}, \perp, r \rangle\!\rangle$, then $\textsc{Val}(C_\Lambda) = \textsc{Val}(C_L)$ by Lemma 3.4.5. Consider a circuit $\mathcal{C}$ constructed as follows: for every $v \in [K] \cong \{0,1\}^k$, let $\alpha_v \in [K]^{P' \setminus \{c\}}$ be such that $\alpha_v{\restriction_{P \setminus \{c\}}} = \alpha$ and $\alpha_v{\restriction_r} = v$.

For any $i \in [k]$, induction hypothesis on $\langle C_\Lambda, \alpha, i, v\restriction_i\rangle$ gives a circuit $\mathcal{C}_\Lambda^{v,i}$, and induction hypothesis on $\langle C_R, \alpha_v, j, b\rangle$ gives a circuit $\mathcal{C}_R^v$, satisfying the conditions. Construct $\mathcal{C} := \bigvee_{v\in[K]}\big(\mathcal{C}_R^v \wedge \bigwedge_{i\in[k]} \mathcal{C}_\Lambda^{v,i}\big)$, then $\mathrm{depth}(\mathcal{C}) \le \max_{v\in[K]}\big\{\mathrm{depth}(\mathcal{C}_R^v), \max_{i\in[k]}\{\mathrm{depth}(\mathcal{C}_\Lambda^{v,i})\}\big\} + \big(k + \lceil\log_2(k+1)\rceil\big)$, and for x with $x\restriction_{P\setminus\{c\}} = \alpha$, $\mathcal{C}(x) = \mathcal{C}_R^v(x)$ for $v := x\restriction_r$. $\square$

4.4 Adversary Argument: when Raz–McKenzie meet Karchmer–Wigderson

Our lower bounds are based on the extension by Raz–McKenzie [85] of the adversary argument by Edmonds–Impagliazzo–Rudich–Sgall [38]. We construct below an interface between the Karchmer–Wigderson (communication) game and the Raz–McKenzie (pebble) game. Note that there is no direct mapping between the two parties in the Karchmer–Wigderson (co-operative) game and the two players in the Raz–McKenzie (competitive) pebble game: the interface between the two games is not straightforward. Also, unlike the case for monotone circuits where it is possible to abstract away the adversary argument as a communication game (called Dart), it appears necessary in the non-monotone case to directly run the adversary argument over the circuit.

Fix a DAG G whose Raz–McKenzie pebble game takes h time, and consider an output-relevant protocol solving BDEP_G^k. It will be shown that the communication game of Karchmer–Wigderson for the evaluation problem $\mathsf{KW}_{\mathsf{BDEP}_G^k}$ must take $\Omega(hk)$ bits of communication for an output-relevant protocol. Recall that the two parties in a communication game want to locate a bit where their inputs differ. Intuitively, for an adversary to foil the two parties, the adversary wants to achieve two conflicting goals: (1) to provide a pair of inputs satisfying the promise of being different; and (2) to hide the difference of a particular input pair among many input pairs, so that the difference is hard for the parties to locate. For hiding the difference, the adversary would maintain a symmetry between the two parties, so that the many input pairs they get look similar (called *same* below). To delay the discovery of the difference (called *different* below) by the two parties, the adversary escalates the decision tree complexity of the Raz–McKenzie pebble game to the communication complexity. For output-relevant protocols, it suffices for the adversary to hide the difference locally, so that no *different* vertices have all its in-neighbors *same* (see Lemma 4.4.4).

We come up with an adversary (for the Karchmer–Wigderson game) that does the following (to play the Raz–McKenzie pebble game for h moves): she keeps track of a set A of alive vertices over G, and also a set C of common values over (effectively) A (i. e., v_a for $a \in A$) that can be given to both parties (Notation 4.1.1). The adversary maintains the symmetry between the parties for all the values over A (by keeping C large) until she is forced to kill some vertex in A. Whenever she kills a vertex, she makes a move for *Pebbler* to pebble the newly killed vertex, and then makes a move for *Colorer* under the optimal strategy against *Pebbler*. The parties must spend $\Omega(k)$ bits of communication on average to force the adversary to kill a vertex. And the adversary against an output-relevant protocol is still in good shape unless h vertices are dead (pebbled), because G takes h time to play the Raz–McKenzie pebble game.

A bit more precisely, consider the product space $X := [K]^V$ with $V := V(G)$, interpreted as the set of possible values given to the two parties. At every node g of the protocol Π, any vertex $a \in V$ is either *same* (under symmetry) or *different* (symmetry is broken) for the two parties. Denote $S := S_g \subseteq V$ as the set of *same* vertices, and $D := D_g := V \setminus S$ as the set of *different* vertices at the node g.[4] The meaning of *same* and *different* vertices is as follows (Definition 4.4.1): for every *different* vertex $d \in D$, there are two non-empty sets $P_d^0, P_d^1 \subset X{\restriction}_d$ of disjoint promised values ($P_d^0 \cap P_d^1 = \emptyset$), such that P_d^1 are some values at d that can be given to Party 1, and P_d^0 are some values at d that can be given to Party 0; and there is a set $C := C_g \subseteq X{\restriction}_S$ of common values over S that can be given to both parties. In addition, a sub-rectangle $Y \times N := Y_g \times N_g \subseteq f_g^{-1}(1) \times f_g^{-1}(0)$ will be associated to a node g.

Definition 4.4.1 (Coherent Data)**.** The data $\langle Y, N, S, D, C, \langle P_d^0, P_d^1 \rangle_{d \in D} \rangle$, where $D = V \setminus S$ and $C \subseteq X{\restriction}_S$, is *coherent* at a node g if (i) $Y \times N \subseteq f_g^{-1}(1) \times f_g^{-1}(0)$; and (ii) for any common value $c \in C$, there are YES instance $x \in Y$ and NO instance $y \in N$, so that they agree with c over S (i. e., $x{\restriction}_S = c = y{\restriction}_S$), and are as promised over D (i. e., for any *different* vertex $d \in D$, we have $x{\restriction}_d \in P_d^1$ and $y{\restriction}_d \in P_d^0$).

It will be shown in §4.5 how the adversary maintains the data $\langle Y, N, S, D, C, \langle P_d^1, P_d^0 \rangle_{d \in D} \rangle$ at different nodes g of Π. Consider the subset of *same* vertices whose values have high entropy under C, and call them alive.

Definition 4.4.2 (Alive Vertices)**.** A vertex $a \in V$ is said to be alive (under C) if $\textsc{AveDeg}_a(C) \geq 8 \cdot K^{19/20}$ (see Definition A.0.22). Let $A := A_g \subseteq S$ be the set of alive vertices (at node g).

The idea is that, if the *same* vertices S, *different* vertices D, and the alive vertices A form a safe configuration (Definition 4.4.3) and the data is coherent (Definition 4.4.1) at a node g, then the adversary is in good shape at g: namely, node g cannot be an output node of the protocol Π, and the two parties need to continue their communication (Lemma 4.4.4). Note that when $\langle S, D, A \rangle$ is in a safe configuration at a node g, A is non-empty, hence $|C{\restriction}_a| \geq \textsc{AveDeg}_a(C) \gg 0$ for any $a \in A$ and C is non-empty.

Definition 4.4.3 (Safe Configuration)**.** The triple $\langle S, D, A \rangle$ with $\emptyset \subset A \subseteq S \subseteq V$ and $D = V \setminus S$ is said to be in a safe configuration if every *different* vertex $d \in D$ has at least one in-neighbor $d' \in \delta^{\text{in}}(d)$ such that $d' \in D \cup A$ is *different* or alive.

Lemma 4.4.4 (Adversary is in Good Shape)**.** *Consider a correct, output-relevant protocol Π for KW_f where $f := \mathsf{BDEP}_G^k$.* ***If*** *at a node g of the protocol Π, there are sets $Y \subseteq f_g^{-1}(1)$, $N \subseteq f_g^{-1}(0)$, $S \subseteq V$ (same), $D := V \setminus S$ (different), and common values $C \subseteq X{\restriction}_S$ over the*

[4]Formally, the sets Y, N, S, D, C, and A can be different for different gate g, but for cleaner notation we may drop the reference to a gate g when it is clear from the context.

*same vertices S, and for any different vertex $d \in D$, there are non-empty sets $P_d^0, P_d^1 \subset X{\upharpoonright}_d$ of disjoint promised values ($P_d^0 \cap P_d^1 = \emptyset$) at d, such that (1) $\langle S, D, A\rangle$ forms a safe configuration where A are the alive vertices (under C); and (2) the data $\left\langle Y, N, S, D, C, \langle P_d^0, P_d^1\rangle_{d\in D}\right\rangle$ is coherent at g, **then** the node g cannot be an output node of Π.*

Proof. Assume that g is a leaf node of Π outputting a bit position $i \in [n]$. The bit position i specifies a variable of t_a for some $a \in V$. Now consider separately whether a is *same* or *different*.

- $a \in S$ is *same*: since C is non-empty, pick any $c \in C$, and by coherence there is an instance $(x, y) \in f_g^{-1}(1) \times f_g^{-1}(0)$ so that $x{\upharpoonright}_S = c = y{\upharpoonright}_S$, hence $x{\upharpoonright}_a = y{\upharpoonright}_a$. If Π is output-relevant, i is a bit (position) relevant to both x and y, so the relevant entries of x and y are the same at a (i. e., $x{\upharpoonright}_{\delta^{\text{in}}(a)} = y{\upharpoonright}_{\delta^{\text{in}}(a)}$, see item (1) of Definition 4.2.3) and $x_i = y_i$, so Π cannot be correct.

- $a \in D$ is *different*: since $\langle S, D, A\rangle$ forms a safe configuration, a has an in-neighbor $a' \in \delta^{\text{in}}(a)$ which is *different* or alive.

  - $a' \in D$ is *different*: since C is non-empty, pick any $c \in C$, and by coherence there is an instance $(x, y) \in f_g^{-1}(1) \times f_g^{-1}(0)$ as promised at a' (i. e., $x{\upharpoonright}_{a'} \in P_{a'}^1$ and $y{\upharpoonright}_{a'} \in P_{a'}^0$). Since $P_{a'}^0$ is disjoint from $P_{a'}^1$, $x{\upharpoonright}_{a'} \neq y{\upharpoonright}_{a'}$.
  - $a' \in A$ is alive: note that $|C{\upharpoonright}_{a'}| \geq \textsc{AveDeg}_{a'}(C) \gg 1$, hence there are distinct $c^1 \neq c^0 \in C{\upharpoonright}_{a'}$. Since $A \subseteq S$, $a' \in S$ is *same*, by coherence there is $x \in f_g^{-1}(1)$ with $x{\upharpoonright}_{a'} = c^1$, and by coherence there is $y \in f_g^{-1}(0)$ with $y{\upharpoonright}_{a'} = c^0$. Hence $x{\upharpoonright}_{a'} \neq y{\upharpoonright}_{a'}$.

  In both cases, there is an instance $(x, y) \in f_g^{-1}(1) \times f_g^{-1}(0)$ such that $x{\upharpoonright}_{a'} \neq y{\upharpoonright}_{a'}$, thus the relevant entries of x and y are different at a (i. e., $x{\upharpoonright}_{\delta^{\text{in}}(a)} \neq y{\upharpoonright}_{\delta^{\text{in}}(a)}$). Then i cannot be a bit (position) relevant to both x and y, and Π cannot be output-relevant. $\square$

To conclude this subsection, the above adversary argument is connected with the Raz–McKenzie pebble game below.

Definition 4.4.5 (Initial Conditions)**.** At the root node r of the protocol Π for KW_f where $f := \mathsf{BDEP}_G^k$, $Y_r := f_r^{-1}(1) = f^{-1}(1)$, $N_r := f_r^{-1}(0) = f^{-1}(0)$, all vertices except the sink vertex τ are *same*, so $D_r := \{\tau\}$ and $S_r := V \setminus D_r$. Let $C_r := X{\upharpoonright}_S$, $P_\tau^1 :=$ k-bit strings whose zeroth bit is 1 and $P_\tau^0 :=$ k-bit strings whose zeroth bit is 0.

Note that initially, all *same* vertices are alive ($A_r = S_r$), the data $\left\langle Y_r, N_r, S_r, D_r, C_r, \langle P_d^0, P_d^1\rangle_{d\in D}\right\rangle$ is coherent, and $\langle S_r, D_r, A_r\rangle$ is in a safe configuration. Throughout the protocol, dead vertices (i. e., non-alive vertices, $V \setminus A$) are precisely the vertices pebbled by *Pebbler* in the Raz–McKenzie pebble game, and the initial conditions correspond to the adversary making the first move of *Pebbler* to pebble w, and making the first (forced) move of *Colorer* to color τ red (i. e., $P = V \setminus A$ throughout, and the initial configuration is C_G^{RM} in RM_G, see

Definition 3.4.12). Later in the protocol, when some vertex $a \in V$ loses too much entropy and dies, the adversary makes a move for *Pebbler* to pebble a, and then makes a move for *Colorer* under the optimal strategy against *Pebbler*, and (1) keeps a as *same* if *Colorer* colors a blue; or (2) marks a as *different* if *Colorer* colors a red. (Thus $B = S \setminus A$ throughout, see Definition 3.4.12.)

Claim 4.4.6 (Safe Till it is Over)**.** *Till the Raz–McKenzie pebble game is over, $\langle S, D, A\rangle$ remains a safe configuration.*

Proof. If $\langle S, D, A\rangle$ is not safe, then some *different* vertex $d \in D$ has all its in-neighbors *same* and dead, i. e., $\delta^{\text{in}}(d) \subseteq S \setminus A$. Since dead vertices are pebbled $V \setminus A = P$ and dead vertices are blue if *same* $B = S \setminus A$ (see Definition 3.4.12), it follows that $d \in P \setminus B$ is red while $\delta^{\text{in}}(d) \subseteq B$ are blue, so the Raz–McKenzie pebble game is over. □

4.5 Recursive Lower Bound

This subsection formally proves Theorem 12, by enforcing the pebbling strategy in § 4.4 with information theoretic (counting) arguments (Appendix A). Throughout this subsection, fix a directed acyclic graph $G = (V, E)$ whose Raz–McKenzie pebble game takes h time, and an output-relevant protocol Π for KW_f where $f := \mathsf{BDEP}^k_G$. For any real number $\alpha \geq 0$ and integer $0 \leq t \leq |V|$, consider the set of all Karchmer–Wigderson games $\mathsf{KW}_{Y,N}$ satisfying

- there is a node $g \in \Pi$ such that $Y \subseteq f_g^{-1}(1)$ and $N \subseteq f_g^{-1}(0)$, where the boolean function f is BDEP^k_G;
- there are sets $S \subseteq V$, $D = V \setminus S$, and for every $d \in D$, there are disjoint sets $P^0_d, P^1_d \subset X{\restriction_d}$, such that $Y{\restriction_d} \subseteq P^1_d$ and $N{\restriction_d} \subseteq P^0_d$;
- there is a large set $C \subseteq X{\restriction_A}$ of values when restricted to a set $A \subseteq S$ of alive co-ordinates (under C, Definition 4.4.2), $|A| = t$, with at most α bits known about C, that is, $\alpha \geq \log_2\bigl(|X{\restriction_A}|/|C|\bigr) = tk - \log_2(|C|)$;
- C is thick, $\textsc{Thickness}(C) \geq K^{17/20}$ (Definition A.0.23 in Appendix A);
- C is common to both Y and N over A, in the sense that $C \subseteq (Y{\restriction_A}) \cap (N{\restriction_A})$;
- the data $\bigl\langle Y, N, S, D, C, \langle P^0_d, P^1_d\rangle_{d\in D}\bigr\rangle$ is coherent at g; and
- the pebble configuration corresponding to $\langle S, D, A\rangle$ has value at least $t - |V| + h$, i. e., $\textsc{Val}(\langle\!\langle P, \perp, B\rangle\!\rangle) \geq t - |V| + h$ where $P := V \setminus A$ and $B := S \setminus A$ (Definitions 3.4.12 and 3.4.13).

Any such game $\mathsf{KW}_{Y,N}$ has data $\bigl\langle g, Y, N, S, D, C, \langle P^0_d, P^1_d\rangle_{d\in D}, A\bigr\rangle$, and denote its communication complexity under output-relevant protocols by $\mathrm{CC}^{\textsc{OutRel}} := \mathrm{CC}^{\textsc{OutRel}}(\mathsf{KW}_{Y,N})$. Note that $\langle S, D, A\rangle$ is in a safe configuration when $t > |V| - h + 1$ by Claim 4.4.6. In this case the data satisfy the conditions of Lemma 4.4.4. In addition to these data, the extra parameters α and t specify respectively the amount of information known about the common values C and the number of alive vertices $|A|$. Denote the collection of such games with parameters α and t as $\textsc{Games}\big[\alpha, t\big]$.

Definition 4.5.1 (Complexity Measure)**.** Let $\textsc{Comp}\big[\alpha, t\big]$ be the minimum communication complexity by output-relevant protocols solving any Karchmer–Wigderson game in $\textsc{Games}\big[\alpha, t\big]$. That is,

$$\textsc{Comp}\big[\alpha, t\big] := \min_{\mathsf{KW}_{Y,N} \in \textsc{Games}\big[\alpha,t\big]} \mathrm{CC}^{\textsc{OutRel}}(\mathsf{KW}_{Y,N}) \ .$$

The following lemma lower bounds the complexity measure, and follows the proof of the main theorem of Raz and McKenzie [85, §6].

Claim 4.5.2 (Recursive Lower Bound)**.** *When $K \geq |V|^{20}$ and $t > |V| - h + 1$,*

$$\textsc{Comp}\big[\alpha, t\big] \geq \min\Big\{ \ \textsc{Comp}\big[\alpha + 2, t\big] + 1, \ \textsc{Comp}\big[\alpha - \tfrac{1}{20}k + 3, t - 1\big] \ \Big\} \ .$$

In particular,

$$\textsc{Comp}\big[\alpha, t\big] \geq \frac{1}{2}\Big[(t - |V| + h - 1)\big(\frac{k}{20} - 3\big) - \alpha\Big] \ .$$

Proof. Consider a Karchmer–Wigderson game $\mathsf{KW}_{Y,N}$ in $\textsc{Games}\big[\alpha, t\big]$ with data $\big\langle g, Y, N, S, D, C, \langle P^0_d, P^1_d\rangle_{d\in D}, A\big\rangle$. There are two cases:

1. for every $j \in A$, $\textsc{AveDeg}_j(C) \geq 8 \cdot K^{19/20}$, and
2. for some $j \in A$, $\textsc{AveDeg}_j(C) < 8 \cdot K^{19/20}$.

Then the first half of the lemma follows from Claims 4.5.3 and 4.5.4 below. Induction then gives the second half. □

Claim 4.5.3 (Recursive Lower Bound, Alive Case)**.** *Assume $t > |V| - h + 1$. If for every $j \in A$, we have $\textsc{AveDeg}_j(C) \geq 8 \cdot K^{19/20}$, then*

$$\mathrm{CC}^{\textsc{OutRel}} \geq \textsc{Comp}\big[\alpha + 2, t\big] + 1 \ .$$

Proof. Recall that Π denotes the output-relevant protocol solving the game. The node g cannot be an output node of Π by Lemma 4.4.4. Assume without loss of generality that Player 1 transmits the first bit at node $g \in \Pi$, which partitions Y into two sets Y_0 and Y_1 (respectively at nodes g_0 and g_1, children of g). Now, we have

$$|(Y_0{\restriction_A}) \cap (N{\restriction_A})| \geq |C|/2 \qquad \text{or} \qquad |(Y_1{\restriction_A}) \cap (N{\restriction_A})| \geq |C|/2 \ .$$

Assume the former without loss of generality, and let $C' := (Y_0\restriction_A) \cap (N\restriction_A)$. The assumption on average degree, together with Lemma A.0.24, gives $\textsc{AveDeg}_j(C') \geq 4 \cdot K^{19/20}$ for every $j \in A$. Now Lemma A.0.27 gives a set $C'' \subseteq C'$ with $|C''| \geq |C'|/2 \geq |C|/4$ and $\textsc{Thickness}(C') \geq K^{17/20}$. Let $Y'' := \{x \in Y \colon x\restriction_A \in C''\}$ be the subset of Y consistent with C'' when restricted to A. Then $\mathsf{KW}_{Y'',N}$ is in $\textsc{Games}[\alpha+2, t]$, and the lemma follows. (A bit more precisely, $\mathsf{KW}_{Y,N}$ is the same as $\mathsf{KW}_{Y'',N}$, except that g is updated to g_0, Y to Y'', and C to C''.) ☐

Claim 4.5.4 (Recursive Lower Bound, Dead Case). *Assume that $K \geq |V|^{20}$. If for some $j \in A$, we have $\textsc{AveDeg}_j(C) < 8 \cdot K^{19/20}$, then*

$$\mathrm{CC}^{\textsc{OutRel}} \geq \textsc{Comp}\big[\alpha - \tfrac{1}{20}k + 3, t-1\big] \ .$$

Proof. We have $\textsc{AveDeg}_j(C) < 8 \cdot K^{19/20}$ and $\textsc{Thickness}(C) \geq K^{17/20}$. Let $A' := A \setminus \{j\}$ and $C' := C\restriction_{A'}$. Now Lemmas A.0.25 and A.0.26 give

$$\frac{|C'|}{|X\restriction_{A'}|} > \frac{|C|}{|X\restriction_A|}\frac{1}{8}K^{1/20} \qquad \text{and} \qquad \textsc{Thickness}(C') \geq K^{17/20} \ ,$$

hence $\log_2\big(|X\restriction_{A'}|/|C'|\big) < \alpha - \frac{1}{20}k + 3$. After making j dead, update $\langle S, D, A\rangle$ to $\langle S', D', A'\rangle$ so that $\textsc{Val}(\langle\!\langle P', \perp, B'\rangle\!\rangle) \geq \textsc{Val}(\langle\!\langle P, \perp, B\rangle\!\rangle) - 1$, where $P' = V \setminus A'$, $B' = S' \setminus A'$, $P = V \setminus A$, and $B = S \setminus A$ (Definitions 3.4.12 and 3.4.13). In case j is made *different*, we need two new sets P_j^0, P_j^1 of promised values at j, as given by Claim 4.5.5 (together with Y', N'). Otherwise, j is made *same*, and let $Y' := Y$ and $N' := N$. In either case, the game $\mathsf{KW}_{Y',N'}$ with data $\langle g, Y', N', S', D', C', \langle P_d^0, P_d^1\rangle_{d\in D'}, A'\rangle$ is in $\textsc{Games}\big[\alpha - \frac{1}{20}k + 3, t-1\big]$, and the lemma follows. ☐

Claim 4.5.5 (Symmetry Breaking). *For $K \geq |V|^{20}$, if Y, N, A and C are such that $C \subseteq (Y\restriction_A) \cap (N\restriction_A)$, given $j \in A$ with $\textsc{MinDeg}_j(C) \geq K^{17/20}$, let $A' := A \setminus \{j\}$ and $C' := C\restriction_{A'}$, then there exist $Y' \subseteq Y$, $N' \subseteq N$, and disjoint $P_j^0, P_j^1 \subset X\restriction_j$, such that $C' \subseteq (Y'\restriction_{A'}) \cap (N'\restriction_{A'})$ and $Y'\restriction_j \subseteq P_j^1$ and $N'\restriction_j \subseteq P_j^0$.*

Proof. Randomly partition $X\restriction_j$ into P_j^1 and P_j^0, by including each string in P_j^1 independently with probability half, and let $P_j^0 := X\restriction_j \setminus P_j^1$. Let Y' be the subset of Y which when projected to j is in P_j^1, and similarly define N' from N and P_j^0. Now $C' \subseteq (Y'\restriction_{A'}) \cap (N'\restriction_{A'})$ fails to hold only when there is a $c' \in C'$ such that all extensions of c' are in P_j^1, or all are in P_j^0. Since $\textsc{MinDeg}_j(C) \geq K^{17/20}$, this happens with probability at most $|C''| \cdot 2^{-K^{17/20}+1} \leq K^{|V|} \cdot 2^{-K^{17/20}+1} \leq 2^{K^{1/20}\log_2 K - K^{17/20}+1} \ll 1$. Hence the claimed sets exist with overwhelming probability.[5] ☐

Theorem 12 (Lower Bound for Evaluation). *For any directed acyclic graph G whose Raz–McKenzie pebble game takes h time, if $2^k \geq |V|^{20}$, then any output-relevant circuit computing BDEP_G^k has depth at least $(h-1)(\frac{k}{40} - 2) = \Omega(hk)$.*

[5]Alternatively, the existence of the claimed set can be demonstrated by a deterministic greedy algorithm.

Chapter 5

Resolution Refutations

This chapter connects the pebble games to some parameters of some refinements of resolution refutations. Namely, concerning a family of unsatisfiable CNFs called pebbling contradictions, the pebble cost in any of the pebble games controls the scaling of the minimum depth of resolution refutations, and of the minimum size of tree-like resolution refutations.

5.1 Size Lower Bound from Depth

For further background on resolution refutations of unsatisfiable formulas, see e. g., [80, 98]. The empty, unsatisfiable formula is denoted as $\perp$.

Urquhart [98] escalated the depth complexity of a resolution refutation to a size lower bound on tree-like resolution refutations, based on the Prover/Delayer game introduced by Pudlák–Impagliazzo [84] and employed by Ben-Sasson–Impagliazzo–Wigderson [12], with the substitution construction of Alekhnovich–Razborov [11] (denoted $\Sigma^{\oplus}$ below; for generalizations, see [13]).

Lemma 5.1.1 (Size Lower Bound from Depth [98, Theorem 5.4])**.** *If $\mathsf{Depth}(\Sigma \vdash \perp) \geq k$, then any tree-like resolution refutation of $\Sigma^{\oplus}$ has size at least 2^k.*

Based on Ben-Sasson–Wigderson [14] which extends Raz–McKenzie [85], Urquhart then constructed a pebbling contradiction formula [98, Theorem 4.6] by escalating the hardness of black pebble game [82], separating the width and depth of resolution refutations. We will see that it suffices to escalate the hardness of reversible black pebble game, which turns out to be connected to the depth complexity of search problems.

Recall that the pebbling contradiction for a graph G is an unsatisfiable formula with one boolean variable per vertex, capturing the logic that (1) all source variables are true; (2) truth propagates through the graph; and (3) some sink variable is false.

Definition 5.1.2 (Pebbling Contradictions)**.** Let Σ_G denote the pebbling contradiction over G, which is a CNF boolean formula defined as follows. Σ_G has one boolean variable v for each vertex $v \in G$. Σ_G is the conjunction over the following clauses, and hence is unsatisfiable.

- for all source vertex v in G, Σ_G has a clause with a single positive literal v;
- for all non-source vertex v in G having in-neighbors $\delta^{\text{in}}(v)$, Σ_G has a clause $v \vee \bigvee_{u \in \delta^{\text{in}}(v)} \bar{u}$; and
- for the sink vertex τ of G, Σ_G has a clause with a single negative literal $\bar{\tau}$.

5.2 Tight Bounds for Tree-Like Resolution

For an unsatisfiable formula Σ, we will need the well-known isomorphism between (regular) tree-like resolution refutations for Σ and decision trees solving the search problem for Σ [12, Lemma 7].

Theorem 3 (Depth of Pebbling Contradictions)**.** *Fix a directed acyclic graph $G = (V, E)$ with a unique sink τ. The depth complexity of resolution refutation for Σ_G is* exactly *the pebble cost in the Raz–McKenzie pebble game to pebble the sink vertex of $\hat{G}$, where $\hat{G} := (V \cup \{\hat{\tau}\}, E \cup \{(\tau, \hat{\tau})\})$ is G augmented with an extra vertex $\hat{\tau}$ as the new sink.*

Proof. Concerning depth complexity, assume the resolution refutation is tree-like without loss of generality. Note that a minimum depth tree-like resolution must be regular (as pointed out by Urquhart [98], this is proved by Grigori Tseitin [95], or alternatively this follows from a simple tree pruning argument [97]). Now it corresponds to a valid strategy in the Raz–McKenzie pebble game over $\hat{G}$. For the other direction, any valid strategy in the Raz–McKenzie pebble game over $\hat{G}$ clearly gives a (regular) tree-like resolution refutation for Σ_G. □

Theorem 4 (Tight Size Bounds for Tree-Like Resolution)**.** *The tree-like resolution refutation of $\Sigma_G^{\oplus}$ has size complexity $2^{\Theta(\text{VAL}(G))}$.*

Proof. Since $\text{VAL}(\hat{G}) \geq \text{VAL}(G)$, Lemma 5.1.1 and Theorem 3 give the lower bound. For the upper bound, we show a tree-like resolution refutation of depth $O\big(\text{VAL}(G)\big)$, using the fact that $\text{VAL}(\hat{G}) \leq \text{VAL}(G) + 1$. Note that a Raz–McKenzie strategy over $\hat{G}$ of value $\text{VAL}(\hat{G})$ naturally gives a decision tree for $\Sigma_G^{\oplus}$ of depth $2\text{VAL}(\hat{G})$, which in turn gives a resolution refutation of depth $2\text{VAL}(\hat{G})$. □

Finally, we extend the lower bound on depth complexity to k-DNF-resolution refutations introduced by Krajíček [63]. For its motivation, see e. g., the survey by Nordström [80]. We follow the standard to treat a term (i. e., a conjunction of literals) dually as a collection of literals.

Definition 5.2.1 (k-DNF-Resolution)**.** Lines in a k-DNF-resolution refutation are k-DNF formulas, derived using the following inference rules (A and B denote k-DNF formulas, S and T denote k-terms, and $l_1, \dots, l_k$ denote literals):

k-cut $\dfrac{(l_1 \wedge \cdots \wedge l_{k'}) \vee A \quad \neg l_1 \vee \cdots \vee \neg l_{k'} \vee B}{A \vee B}$, where $k' \leq k$.

$\wedge$-introduction $\dfrac{A \vee S \quad A \vee T}{A \vee (S \wedge T)}$, where $|S \cup T| \leq k$.

$\wedge$-elimination $\dfrac{A \vee S}{A \vee T}$, where $T \subseteq S$.

Weakening $\dfrac{A}{A \vee B}$, for any k-DNF formula B.

Theorem 5. *Any k-DNF-resolution refutation of Σ_G has depth at least $1 + (\textsc{Val}(G) - 1)/k$.*

Proof. Imagine an adversary, who keeps track of a k-DNF formula A_d at 'depth' d in the refutation and a restriction ρ_d, satisfying the invariant that (1) ρ_d falsifies A_d; and (2) the configuration corresponding to ρ_d has value at least $\textsc{Val}(\hat{G}) - (d-1)k$, i. e., let B_d be the variables assigned True under ρ_d, R_d False (the extra sink of $\hat{G}$ is always assumed False), then $\textsc{Val}((\!|B_d, R_d|\!)) \geq \textsc{Val}(\hat{G}) - (d-1)k$.

The adversary starts with the unsatisfiable k-DNF formula $A_1 := \perp$ and $\rho_1 :=$ the unique sink of $\hat{G}$ is False, satisfying the invariant at 'depth' $d := 1$. If the adversary hits an axiom formula A_d from Σ_G, then ρ_d falsifies A_d, i. e., $\textsc{Val}((\!|B_d, R_d|\!)) = 1$, giving the required depth on the refutation.

Otherwise, we have A_d and ρ_d, where A_d is the result (i. e., on the bottom row) of an inference rule. We will locate A_{d+1} as one of the formulas on the top row of the inference rule, and update ρ_{d+1} appropriately. For the $\wedge$-elimination rule and the weakening rule, the adversary takes A_{d+1} as the only formula on the top row, and takes $\rho_{d+1} := \rho_d$ to maintain the invariant. For the $\wedge$-introduction rule, the adversary takes $\rho_{d+1} := \rho_d$, and takes A_{d+1} to be a formula on the top row that is falsified by ρ_{d+1}.

For the remaining, interesting case of a k-cut rule, the adversary maintains A_{d+1} and ρ_{d+1} by the recurrence of the Raz–McKenzie pebble game (Proposition 3.5.2). There are at most k fresh variables among the literals $l_1, \ldots, l_{k'}$ outside of the domain of ρ_d. At least one assignment to the fresh variables gives an extension ρ_{d+1} to ρ_d such that $\textsc{Val}((\!|B_{d+1}, R_{d+1}|\!)) \geq \textsc{Val}((\!|B_d, R_d|\!)) - k$, and at least one formula A_{d+1} on the top row is falsified by ρ_{d+1}. □

Chapter 6

Space Complexity of Pebble Games

This chapter settles the space complexity of computing the pebble costs in the Bennett–Dymond–Tompa–Raz–McKenzie pebble game, and the minimum depth of resolution refutation. Recall that we want to show Theorem 7 via Theorem 6, which are restated below.

Theorem 7 (PSPACE-Completeness)**.** *It is PSPACE-complete to compute (1) the pebble cost in the Bennett–Dymond–Tompa–Raz–McKenzie pebble game; or (2) the minimum depth of resolution refutation.*

Theorem 6 (Log-Space Reduction)**.** *There is a logspace algorithm that, given a quantified boolean formula φ with m clauses over n variables, outputs a graph $\hat{G}_\varphi$, such that φ is satisfiable iff the pebble cost of $\hat{G}_F$ is at most $\gamma + 1$, where $\gamma := 7 + m + 3n + \alpha_n$ and α_n is the number of universal quantifiers in φ. Moreover, after deleting the sink node of G_φ, the resulting graph also has a unique sink node.*

6.1 Gadgets

Quantifier Gadgets

Throughout this subsection, assume γ_i pebbles are given for some integer $\gamma_i \geq 5$.

Following previous works [43,52], truth values are represented using the gadget in Fig. B.1. Say variable x_i is in *true position* if nodes x_i and $\bar{x}'_i$ in the gadget are pebbled, and x'_i is unpebbled. Say variable x_i is in *false position* if nodes $\bar{x}_i$ and x'_i in the gadget are pebbled, and $\bar{x}'_i$ is unpebbled.

For cleaner diagrams, the gadget on the LHS of Fig. B.2 means that k extra source nodes precede node v. Note that node v may have other immediate predecessors in addition to the extra source nodes. For example, the node q_i in Fig. B.3 has $\gamma_i - 2$ immediate predecessors: $\gamma_i - 5$ extra source nodes, plus x_i, $\bar{x}_i$, and q_{i-1}.

The existentially quantified variable $\exists x_i$ has a gadget as shown in Fig. B.3. Note that node q_{i-1} comes from the gadget of the adjacent quantified variable $Q_{i-1}x_{i-1}$.

The universally quantified variable $\forall x_i$ has a gadget as shown in Fig. B.4. Note that node q_{i-1} comes from the gadget of the adjacent quantified variable $Q_{i-1}x_{i-1}$.

Clause Gadgets

The gadget for the j-th clause, $l_{j,1} \vee l_{j,2} \vee l_{j,3}$, has its skeleton shown in Fig. B.5. Assume that the literals $l_{j,1}, l_{j,2}, l_{j,3}$ are over distinct variables. Note that in Fig. B.5 the six nodes $l_{j,1}$, $l'_{j,1}$ $l_{j,2}$, $l'_{j,2}$ $l_{j,3}$, and $l'_{j,3}$ come from the quantifier gadgets corresponding to the variables in literals $l_{j,1}$, $l_{j,2}$ and $l_{j,3}$. Say literal $l_{j,1}$ is in *true position* if node $l_{j,1}$ is pebbled but node $l'_{j,1}$ is unpebbled. Say literal $l_{j,1}$ is in *false position* if node $l'_{j,1}$ is pebbled (regardless of whether node $l_{j,1}$ is pebbled). Let the *corresponding node* for literal $l_{j,1}$ in true (resp. false) position be node $l_{j,1}$ (resp. $l'_{j,1}$). Define similarly for literals $l_{j,2}$ and $l_{j,3}$.

For example, if literal $l_{j,1}$ is in true position, literals $l_{j,2}, l_{j,3}$ false position, then their corresponding nodes are $l_{j,1}, l'_{j,2}, l'_{j,3}$. Let $C_j := \{a_j, b_j, c_j, d_j, e_j, f_j, u_j, v_j, w_j, p_j\}$ be nodes not shared with the quantifier gadgets. The gadget behaves like a disjunction in the sense that at least one literal is in true position if, and only if, six additional pebbles are needed to reach a certain configuration (Lemmas 6.1.1 to 6.1.3).

Lemma 6.1.1 (Six is Sufficient)**.** *Assume at least one of the literals $l_{j,1}, l_{j,2}, l_{j,3}$ is in true position, and the rest are in false position. Let L_j be their corresponding nodes. Six pebbles (in addition to those on L_j) are* sufficient *to reach the configuration where precisely $L_j \cup \{p_j\}$ are pebbled, starting from the configuration where precisely L_j are pebbled, under the constraint that L_j remain pebbled.*

Proof. Assume exactly one literal is in true position for the moment. Without loss of generality, assume literal $l_{j,1}$ is in true position, and literals $l_{j,2}, l_{j,3}$ are in false position. Their corresponding nodes are $L_j = \{l_{j,1}, l'_{j,2}, l'_{j,3}\}$.

Under the constraint that L_j remain pebbled, consider the following strategy to pebble p_j using six pebbles (in addition to L_j):

1. Pebble $l_{j,2}$ then b_j then e_j, unpebble b_j then $l_{j,2}$. Now precisely $L_j \cup \{e_j\}$ are pebbled.
2. Pebble $l_{j,3}$ then c_j then f_j then w_j, unpebble f_j then c_j then $l_{j,3}$. Now precisely $L_j \cup \{w_j, e_j\}$ are pebbled.
3. Pebble a_j then d_j then u_j, unpebble d_j then a_j. Now precisely $L_j \cup \{u_j, w_j, e_j\}$ are pebbled.
4. Pebble $l_{j,2}$ then b_j, unpebble e_j then b_j then $l_{j,2}$. Now precisely $L_j \cup \{u_j, w_j\}$ are pebbled.
5. Pebble $l_{j,3}$ then c_j then f_j, unpebble c_j then $l_{j,3}$. Now precisely $L_j \cup \{u_j, w_j, f_j\}$ are pebbled.

6. Pebble a_j then d_j, unpebble a_j, pebble v_j. Now precisely $L_j \cup \{u_j, v_j, w_j, d_j, f_j\}$ are pebbled.

7. Pebble p_j. Now precisely $L_j \cup \{p_j, u_j, v_j, w_j, d_j, f_j\}$ are pebbled.

In order to pebble precisely $L_j \cup \{p_j\}$, do the following instead.

i. Run the above steps 1, 2, 3, 4, 5, 6 and 7 to pebble p_j.

ii. Run the *reverses* of the above steps 6, 5, 4, 3, 2 and 1. For example, the reverse of step 6 is to unpebble v_j, pebble a_j, unpebble d_j then a_j. After running the reverse of step 6, precisely $L_j \cup \{u_j, w_j, f_j\} \cup \{p_j\}$ are pebbled. Further after running the reverses of steps 5, 4, 3, 2 and 1, precisely $L_j \cup \{p_j\}$ are pebbled.

This completes the proof if exactly one literal is in true position.

If more literals are in true position, note that it does not take more pebbles. For example, assume literals $l_{j,1}$ and $l_{j,2}$ are in true position, literal $l_{j,3}$ in false position. In this case, the corresponding nodes are $L_j = \{l_{j,1}, l_{j,2}, l'_{j,3}\}$. Then the above steps 1–7 take no more then six additional pebbles, by ignoring pebble or unpebble moves on $l_{j,2}$. □

Lemma 6.1.2 (Six is Necessary)**.** *Assume each of literals $l_{j,1}$, $l_{j,2}$, or $l_{j,3}$ is in true position or false position. Let L_j be their corresponding nodes. Six pebbles (in addition to those on L_j) are* necessary *to reach the configuration where precisely $L_j \cup \{p_j\}$ are pebbled, starting from the configuration where precisely L_j are pebbled, under the constraint that L_j remain pebbled.*

Proof. Consider the first time t_1 such that p_j remains pebbled since t_1. A pebble move put a pebble on p_j at t_1, hence all of u_j, v_j and w_j are pebbled at t_1. At the end, all of u_j, v_j and w_j are unpebbled. Let t_2 be the first time after t_1 such that one of u_j, v_j or w_j is unpebbled at time $t_2 + 1$. Without loss of generality, assume u_j is unpebbled at time $t_2 + 1$, then d_j and e_j are pebbled at time t_2. Moreover, all of u_j, v_j and w_j remain pebbled between time t_1 and t_2. At time t_2, the six nodes p_j, u_j, v_j, w_j, d_j and e_j are pebbled. □

Lemma 6.1.3 (Seven is Necessary)**.** *Assume all of the literals $l_{j,1}$, $l_{j,2}$ and $l_{j,3}$ are in false position. Let $L_j := \{l'_{j,1}, l'_{j,2}, l'_{j,3}\}$ be their corresponding nodes. Seven pebbles (in addition to those on L_j) are* necessary *to reach the configuration where precisely $L_j \cup \{p_j\}$ are pebbled, starting from the configuration where precisely L_j are pebbled, under the constraint that L_j remain pebbled. The seven pebbles are needed at a time to pebble or unpebble some node in C_j.*

Proof. Define t_1 and t_2 as in the proof of Lemma 6.1.2. If there are at least seven pebbled nodes addition to L_j at time t_2 (when $u_j \in C_j$ is unpebbled), then we are done. Otherwise, in addition to L_j, precisely the six nodes $\{p_j, u_j, v_j, w_j, d_j, e_j\}$ are pebbled at time t_2, hence precisely five nodes $P := \{p_j, v_j, w_j, d_j, e_j\}$ are pebbled at time $t_2 + 1$. Consider the first time t_3 after t_2 such that some node in P is unpebbled at time t_3+1, which must happen by the end. Between t_2 and t_3, all of P are pebbled. The node unpebbled at time t_3+1 cannot be p_j (which remains pebbled since t_1), and if (i) it is d_j, then a_j is pebbled at time t_3; since a_j is not pebbled at time t_2+1, $a_j \in C_j$ is first pebbled at some time t_4 between t_2 and t_3, and both a_j and $l_{j,1}$ are pebbled at t_4, giving a total of seven pebbles together with P; and if (ii) it is e_j, this case is symmetric to the first case; and if (iii) it is v_j, essentially the same argument as in the first case would do, by considering f_j instead of a_j, and if (iv) it is w_j, this case is symmetric to the third case. □

This motivates modifying Fig. B.5 to Fig. B.7, using the notation in Fig. B.6. Henceforth, assume β_j pebbles are given for some integer $\beta_j \geq 7$. For cleaner diagrams, introduce a dashed region notation on the LHS of Fig. B.6 to add extra sources and a common immediate predecessor to every node in a region (cf. Fig. B.2). Augment the clause gadget as in Fig. B.7 to add $\beta_j - 7$ sources and a common predecessor p_{j-1} to each node in C_j. For Lemmas 6.1.4 and 6.1.6, treat p_{j-1} as a source node.

Lemma 6.1.4 (β_j is Sufficient). *Assume at least one of the literals $l_{j,1}, l_{j,2}, l_{j,3}$ is in true position, and the rest are in false position. Let L_j be their corresponding nodes. With β_j additional pebbles, the configuration $L_j \cup \{p_j\}$* can *be reached, starting from the configuration L_j, under the constraint that L_j remain pebbled.*

Proof. Modify the proof of Lemma 6.1.1. Add an initial move to pebble p_{j-1}. Before each pebble or unpebble move to a node in C_j, first pebble all extra sources of that node using $\beta_j - 7$ pebbles. After each such move, remove all $\beta_j - 7$ pebbles. Add a final move to unpebble p_{j-1}. At most β_j pebbles are used. □

Lemma 6.1.5 (β_j is Necessary). *Assume at least one of the literals $l_{j,1}, l_{j,2}, l_{j,3}$ is in true position, and the rest are in false position. Let L_j be their corresponding nodes. To reach the configuration $L_j \cup \{p_j\}$, starting from the configuration L_j, under the constraint that L_j remain pebbled, there is a time such that β_j additional pebbles are used, and at which a node in C_j is being pebbled or unpebbled.*

Proof. Consider time t_2 as in the proof of Lemma 6.1.2. Six pebbles (in addition to those on $L_j \cup \{p_{j-1}\}$ or extra sources) are present in the augmented clause gadget in Fig. B.7. Since a node in C_j is being pebbled or unpebbled, its $\beta_j - 7$ extra sources are pebbled. This gives a total of β_j pebbles (in addition to L_j). □

Lemma 6.1.6 (β_j is Not Enough). *Assume all of the literals $l_{j,1}$, $l_{j,2}$ and $l_{j,3}$ are in false position. Let $L_j := \{l'_{j,1}, l'_{j,2}, l'_{j,3}\}$ be their corresponding nodes. With β_j additional pebbles, the configuration $L_j \cup \{p_j\}$* cannot *be reached, starting from the configuration L_j, under the constraint that L_j remain pebbled.*

Proof. Fix a strategy to pebble $L_j \cup \{p_j\}$ from L_j. By Lemma 6.1.3, there must be a time t when seven pebbles (in addition to those on $L_j \cup \{p_{j-1}\}$ or the extra sources) are present in the augmented clause gadget in Fig. B.7. Moreover, at time t, a node in C_j is being pebbled or unpebbled, thus its $\beta_j - 7$ extra sources are pebbled. This requires $\beta_j + 1$ pebbles (in addition to L_j). □

Overall Construction

Fix a quantified boolean formula $\varphi := Q_n x_n Q_{n-1} x_{n-1} \dots Q_1 x_1 F$ with n quantified variables, where $F := \bigwedge_{j=1}^{m} l_{j,1} \vee l_{j,2} \vee l_{j,3}$ is a 3CNF formula with m clauses. Assume that the three literals in every clause are over distinct variables.

Construction 6.1.7. Construct a directed graph G_φ as follows. Let α_i be the number of universal quantifiers among the inner-most i quantifiers $Q_i, Q_{i-1}, \dots, Q_1$. Let $\beta_j := 6 + j$ and $\gamma_i := \beta_m + 3i + \alpha_i$. Note that $\beta_j = \beta_{j-1} + 1$ and $\gamma_0 = \beta_m$. If Q_i is an existential quantifier, then $\gamma_i = \gamma_{i-1} + 3$; if Q_i is a universal quantifier, then $\gamma_i = \gamma_{i-1} + 4$. Let $\gamma := \gamma_n + 1 = 7 + m + 3n + \alpha_n$.

Construct a node p_0 with five extra sources. For $1 \leq j \leq m$, construct the dashed region in the augmented clause gadget for clause j as in Fig. B.7. Let $q_0 := p_m$. For $1 \leq i \leq n$, construct a quantifier gadget for the quantified variable $Q_i x_i$ as in Fig. B.3 or Fig. B.4. Construct a node τ with $\gamma - 2$ extra sources, and add an edge from q_n to τ. For $1 \leq j \leq m$, for $1 \leq k \leq 3$, connect according to the literal $l_{j,k}$, i. e., add an edge from the node $l_{j,k}$ (in some quantifier gadget) to the corresponding node in the augmented clause gadget for clause j as in Fig. B.7. This completes the construction.

For an example, see Fig. B.8.

Definition 6.1.8 (Partial Assignment)**.** A partial assignment $\rho\colon [n] \to \{0, 1, *\}$ is a partial function from variables to boolean values.[1] Say variable x_i is pebbled *according to* ρ if (1) x_i is unassigned (i. e., $\rho(i) = *$) and no node in Fig. B.1 is pebbled; or (2) x_i is assigned to true (i. e., $\rho(i) = 1$) and variable x_i is precisely in true position (i. e., precisely x_i and $\bar{x}'_i$ are pebbled in Fig. B.1); order (3) x_i is assigned to false (i. e., $\rho(i) = 0$) and variable x_i is precisely in false position (i. e., precisely $\bar{x}_i$ and x'_i are pebbled in Fig. B.1). Let U_ρ denote the union of nodes, over all variables, pebbled according to ρ. For brevity, call a partial assignment ρ an *i-assignment* if it leaves exactly the inner-most i variables unassigned, i. e., $\rho(k) = *$ iff $k \leq i$.

Definition 6.1.9 (Restriction)**.** Fix a quantified boolean formula $\varphi := Q_n x_n Q_{n-1} x_{n-1} \dots Q_1 x_1 F$, where F is a 3CNF formula. For an i-assignment ρ, the restriction of φ by ρ is $\varphi\!\restriction_\rho := Q_i x_i Q_{i-1} x_{i-1} \dots Q_1 x_1 (F\!\restriction_\rho)$, where $F\!\restriction_\rho$ denotes the CNF with variables assigned according to ρ.

[1] In this chapter, we define $[n] := \{1, 2, \dots, n\}$.

Notation 6.1.10 (Predecessors)**.** Say node v is a (not necessarily proper) *predecessor* of node u if there is a path (of possibly zero length) from v to u. Denote the set of predecessors of node u by V_u.

To simplify the description of pebbling strategies, when a move is made to pebble or unpebble a node v with extra sources, it means to prepend a sequence of moves to pebble all extra sources of v, and to append a sequence of moves to unpebble all extra sources of v. It can be verify that the extra sources can be pebbled within the claimed number of pebbles. Also, all time intervals mentioned below are inclusive, i. e., the start and end times are included.

Lemma 6.1.11. *For $0 \le i \le n$, for any i-assignment ρ, if $\varphi\lceil_\rho$ is true, then starting from configuration U_ρ, there is a strategy to use γ_i pebbles (in addition to those on U_ρ) to reach configuration $U_\rho \cup \{q_i\}$, under the constraint that U_ρ remain pebbled.*

Proof. When $i = 0$, recall that $q_0 = p_m$ and $\gamma_0 = \beta_m$, then Lemma 6.1.11 in this case follows from Claim 6.1.12.

Claim 6.1.12. *For $0 \le j \le m$, for any 0-assignment ρ, if the first j clauses of F are satisfied by ρ, then starting from the configuration U_ρ, there is a strategy S_j to use β_j pebbles (in addition to those on U_ρ) to reach configuration $U_\rho \cup \{p_j\}$, under the constraint that U_ρ remain pebbled.*

Proof. When $j = 0$, the strategy S_0 is to pebble the extra sources of p_0, pebble p_0, then unpebble the extra sources of p_0, using $\beta_0 = 6$ additional pebbles.

When $j > 0$, assume that the first j clauses of F are satisfied by ρ. That is, the first $j-1$ clauses of F are satisfied by ρ, and at least one literal of clause j is true under ρ. Then Claim 6.1.12 follows from the proof of Lemma 6.1.4: to pebble p_{j-1} as the initial move, run S_{j-1} instead; to unpebble p_{j-1} as the final move, run the *reverse* of S_{j-1} instead. At most β_j additional pebbles are needed, since $\beta_j = \beta_{j-1} + 1$. □

When $i > 0$, there are two cases. Let ρ_1 be the $(i-1)$-assignment obtained from ρ by setting x_i to true (i. e., $\rho_1(k) := 1$ if $k = i$, and $\rho_1(k) := \rho(k)$ otherwise), and ρ_0 be obtained from ρ by setting x_i to false.

(Existential Quantifier) If $Q_i = \exists$, since $\varphi\lceil_\rho$ is true, at least one of $\varphi\lceil_{\rho_1}$ or $\varphi\lceil_{\rho_0}$ is true. Assume the former without loss of generality, consider the following strategy. Focus on Fig. B.3.

1. Pebble x'_i and x_i, unpebble x'_i, pebble $\bar{x}'_i$. Now precisely nodes x_i and $\bar{x}'_i$ are pebbled, and variable x_i is pebbled according to ρ_1.

2. Apply induction hypothesis on ρ_1 to pebble q_{i-1}. Now precisely nodes x_i, $\bar{x}'_i$ and q_{i-1} are pebbled.

3. Pebble $\bar{x}_i$. Now precisely nodes x_i, $\bar{x}_i$, $\bar{x}'_i$ and q_{i-1} are pebbled.

4. Pebble q_i.

The claimed strategy is to run steps 1–4, and then the *reverses* of steps 3–1. At most γ_i additional pebbles are used, since $\gamma_i = \gamma_{i-1} + 3$.

(Universal Quantifier) If $Q_i = \forall$, since $\varphi\lceil_\rho$ is true, both $\varphi\lceil_{\rho_1}$ and $\varphi\lceil_{\rho_0}$ are true. Consider the following strategy. Focus on Fig. B.4.

1. Pebble x'_i and x_i, unpebble x'_i, pebble $\bar{x}'_i$. Now precisely nodes x_i and $\bar{x}'_i$ are pebbled, and variable x_i is pebbled according to ρ_1.
2. Apply induction hypothesis on ρ_1 to pebble q_{i-1}. Now precisely nodes x_i, $\bar{x}'_i$ and q_{i-1} are pebbled.
3. Pebble h_i. Now precisely nodes x_i, $\bar{x}'_i$, q_{i-1} and h_i are pebbled.
4. Apply (the reverse strategy in) induction hypothesis on ρ_1 to unpebble q_{i-1}. Now precisely nodes x_i, $\bar{x}'_i$ and h_i are pebbled.
5. Run the reverse of step 1 to unpebble x_i and $\bar{x}'_i$. Now precisely node h_i is pebbled.
6. Pebble $\bar{x}'_i$ and $\bar{x}_i$, unpebble $\bar{x}'_i$, pebble x'_i. Now precisely nodes $\bar{x}_i$, x'_i and h_i are pebbled, and variable x_i is pebbled according to ρ_0.
7. Apply induction hypothesis on ρ_0 to pebble q_{i-1}. Now precisely nodes $\bar{x}_i$, x'_i, h_i and q_{i-1} are pebbled.
8. Pebble q_i.

The claimed strategy is to run steps 1–8, and then the *reverses* of steps 7–1. At most γ_i additional pebbles are used, since $\gamma_i = \gamma_{i-1} + 4$. □

Lemma 6.1.13. *For $0 \le i \le n$, for any i-assignment ρ, if starting from configuration U_ρ, there is a strategy to use γ_i pebbles (in addition to those on U_ρ) to reach configuration $U_\rho \cup \{q_i\}$, under the constraint that U_ρ remain pebbled, then $\varphi\lceil_\rho$ is true.*

Proof. When $i = 0$, recall that $q_0 = p_m$ and $\gamma_0 = \beta_m$, then Lemma 6.1.13 in this case follows from Claim 6.1.14.

Claim 6.1.14. *For $0 \le j \le m$, for any 0-assignment ρ, if starting from the configuration U_ρ, there is a strategy to use β_j pebbles (in addition to those on U_ρ) to reach configuration $U_\rho \cup \{p_j\}$, under the constraint that U_ρ remain pebbled, then the first j clauses of F are satisfied by ρ.*

Proof. When $j = 0$, the first 0 clauses of F are vacuously satisfied by ρ. When $j > 0$, assume there is a strategy S_j to use β_j additional pebbles to reach configuration $U_\rho \cup \{p_j\}$ starting from U_ρ. By Lemma 6.1.6, at least one literal of clause j is true under ρ. Let t be the time when β_j pebbles are used as given by Lemma 6.1.5. When restricted to the predecessors of p_{j-1}, the sub-strategy of S_j since time t gives a strategy S_{j-1} to reach configuration U_ρ starting from $U_\rho \cup \{p_{j-1}\}$ using at most $\beta_j - 1 = \beta_{j-1}$ additional pebbles (because p_j remains pebbled between time t and the end). Induction hypothesis on (the reverse of) S_{j-1} shows that the first $j-1$ clauses of F are satisfied by ρ. Hence the first j clauses of F are satisfied by ρ. □

When $i > 0$, there are two cases. Let ρ_1 be the $(i-1)$-assignment obtained from ρ by setting x_i to true (i. e., $\rho_1(k) := 1$ if $k = i$, and $\rho_1(k) := \rho(k)$ otherwise), and ρ_0 be obtained from ρ by setting x_i to false. Let $P := V_{q_{i-1}} \setminus \{x_i, \bar{x}_i, x'_i, \bar{x}'_i\}$ be the predecessors of q_{i-1} outside of the variable gadget in Fig. B.1.

(Existential Quantifier) If $Q_i = \exists$, focus on Fig. B.3. Consider the first time t since when q_i remains pebbled till the end. At time t, both x_i and $\bar{x}_i$ are pebbled.

Claim 6.1.15 (No Double True). *Either x'_i or $\bar{x}'_i$ is pebbled at time t.*

Proof. Assume not, prove by induction on time since t that q_i, x_i and $\bar{x}_i$ are pebbled, but x'_i and $\bar{x}'_i$ are unpebbled. This is true by assumption at time t. There are three pebbles on q_i, x_i and $\bar{x}_i$, hence x'_i cannot be pebbled using γ_i pebbles, and neither can $\bar{x}'_i$. Hence x_i and $\bar{x}_i$ cannot be unpebbled. This contradicts that x_i and $\bar{x}_i$ are unpebbled at the end in U_ρ. □

At time t, the nodes q_i, x_i, $\bar{x}_i$ and q_{i-1} are pebbled. All $\gamma_i - 5$ extra sources of q_i are also pebbled. Using γ_i pebbles, at most one of x'_i or $\bar{x}'_i$ can be pebbled at time t. By Claim 6.1.15, exactly one of them is pebbled, say $\bar{x}'_i$, then variable x_i is in true position. No other node is pebbled.

Claim 6.1.16. *Until all of P are unpebbled, nodes x_i and $\bar{x}'_i$ remain pebbled, and x'_i remains unpebbled.*

Proof. At time t, nodes q_i, x_i and $\bar{x}'_i$ are pebbled, and x'_i is unpebbled. If some node of P is pebbled, then this accounts for four out of γ_i pebbles. Hence $\bar{x}'_i$ cannot be unpebbled, and x'_i cannot be pebbled. Hence x_i cannot be unpebbled. □

The three nodes q_i, x_i and $\bar{x}'_i$ remain pebbled until all of P are unpebbled. So at most $\gamma_i - 3 = \gamma_{i-1}$ pebbles can be used in this time interval. By induction hypothesis on ρ_1, $\varphi\lceil_{\rho_1}$ is true. The other case, where x'_i is pebbled at time t, shows that $\varphi\lceil_{\rho_0}$ is true. In either case, $\varphi\lceil_\rho$ is true.

(Universal Quantifier) If $Q_i = \forall$, focus on Fig. B.4. Consider the first time t_1 since when q_i remains pebbled till the end. At time t_1, nodes q_i, h_i, $\bar{x}_i$, x'_i and q_{i-1} are pebbled. All $\gamma_i - 5$ extra sources of q_i are also pebbled. Using γ_i pebbles, no other node is pebbled at t_1, and variable x_i is in false position. Let t_2 be the first time after t_1 such that all of $P = V_{q_{i-1}} \setminus \{x_i, \bar{x}_i, x'_i, \bar{x}'_i\}$ are unpebbled.

Claim 6.1.17. *Until t_2, nodes $\bar{x}_i$, x'_i and h_i remain pebbled, node $\bar{x}'_i$ remain unpebbled.*

Proof. At time t_1, nodes q_i, $\bar{x}_i$, x'_i and h_i are pebbled, node $\bar{x}'_i$ is unpebbled. If some node of P is pebbled, then this accounts for five out of γ_i pebbles. Hence x'_i cannot be unpebbled, and $\bar{x}'_i$ cannot be pebbled. Hence h_i and $\bar{x}_i$ cannot be unpebbled. □

The four nodes q_i, h_i, $\bar{x}_i$ and x'_i remain pebbled between t_1 and t_2. So at most $\gamma_i - 4 = \gamma_{i-1}$ pebbles can be used between t_1 and t_2. By induction hypothesis on ρ_0, $\varphi\lceil_{\rho_0}$ is true.

Let t_3 be the first time after t_2 such that h_i is unpebbled at $t_3 + 1$. At t_3, nodes q_{i-1}, x_i and $\bar{x}'_i$ are pebbled. Let t_4 be the first time such that at all time between t_4 and t_3, at least one of x_i or x'_i is pebbled. Let t_5 be the first time such that at all time between t_5 and t_3, at least one of $\bar{x}_i$ or $\bar{x}'_i$ is pebbled. Let t_6 be the first time such that at all time between t_6 and t_3, some node of P is pebbled. Note that t_6 is after t_2. Let $t_7 := \max\{t_4, t_5, t_6\}$ be the last event of the three, clearly t_7 is after t_2.

Claim 6.1.18 (Fixed Position)**.** *Between t_7 and t_3, there is no move to pebble or unpebble x'_i or $\bar{x}'_i$.*

Proof. Both nodes q_i and h_i remain pebbled between t_2 to t_3. Between t_7 and t_3, at least one pebble is on x_i or x'_i, at least one pebble is on $\bar{x}_i$ or $\bar{x}'_i$, and at least one pebble is on P. This accounts for five out of γ_i pebbles, so not all extra sources of x'_i or of $\bar{x}'_i$ can be pebbled. □

At time t_3, the five nodes q_i, h_i, x_i, $\bar{x}'_i$ and q_{i-1} are pebbled. All $\gamma_i - 5$ extra sources of h_i are also pebbled. Hence no other node is pebbled, and variable x_i is in true position. By Claim 6.1.18, $t_7 = t_6$, and at $t_7 - 1$ all of P is unpebbled. It follows that the four nodes q_i, h_i, x_i and $\bar{x}'_i$ remain pebbled between $t_7 - 1$ and t_3. At most $\gamma_i - 4 = \gamma_{i-1}$ pebbles can be used in this time interval. By induction hypothesis on ρ_1, $\varphi\lceil_{\rho_1}$ is true.

Since $\varphi\lceil_{\rho_1}$ is true and $\varphi\lceil_{\rho_0}$ is true, $\varphi\lceil_{\rho}$ is true. □

Theorem 13. *The sink node τ of G_φ can be pebbled using γ pebbles iff φ is true.*

Proof Sketch. By Lemmas 6.1.11 and 6.1.13. □

Theorem 14. *Consider $\hat{G}_\varphi := (V \cup \{\hat{\tau}\}, E \cup \{(\tau, \hat{\tau})\})$, which is $G_\varphi =: (V, E)$ augmented with an extra node $\hat{\tau}$ as the new sink. There is a pebbling strategy using $\gamma + 1$ pebbles, starting from the empty configuration, to the configuration where precisely $\hat{\tau}$ is pebbled, iff φ is true.*

Chapter 7

Some Related Approaches

We recall below some related approaches for separating complexity classes mostly around P.

Multi-Party Communication Complexity As an approach to separate ACC^0 from P, researchers considered the multi-player pointer jumping problem [20, 25, 100], with the aim of proving a sufficiently strong lower bound in the number-on-forehead multi-party (simultaneous message) communication model. A variant of the problem with a tree structure, called tree pointer jumping problem [100], is like the tree evaluation problem with information flowing in the reverse direction (from root to leaves).

Extension to Karchmer–Wigderson framework Aaronson–Wigderson [1] extended the Karchmer–Wigderson framework [59] to consider a refereed communication game between two parties (verifiers) and an additional prover, where a sufficiently strong lower bound on communication complexity would separate NL from NP. Kol–Raz [62] extended the Aaronson–Wigderson framework of refereed communication game to a competing-prover protocol with two verifiers and two provers, and suggested it as an approach for separating NC from P.

Block-Respecting Simulations Lipton–Williams [73] recently suggested that a sufficiently strong lower bound on depth (e. g., $n^{1-O(1)}$) may be able to separate NC from P, even with a very weak lower bound on size (e. g., $n^{1+\Omega(1)}$), by using a block-respecting simulation to trade depth for size and non-uniformity. The idea of proving lower bounds by trading depth for size was due to Allender–Koucký [5].

Combinatorial Invariants Mulmuley–Sohoni [78, 79] advocated the study of symmetry and invariants of the computational problems as an approach for separating VP from VNP, the non-uniform and algebraic analogue of P versus NP. One motivation is that Mulmuley [76] applied semi-algebraic geometry to give a non-uniform and algebraic separation of $\mathsf{alg\text{-}NC}$ from $\mathsf{alg\text{-}P}$ and $\mathsf{alg\text{-}NC}^i$ from $\mathsf{alg\text{-}NC}^{i+1}$ on a restricted model of PRAM without bit operations, setting the stage for proving stronger lower bounds. Another motivation is that properties described by certain combinatorial invariants are unlikely to be large or natural in the sense of Razborov–Rudich [89], see e. g., Mulmuley [77, §4.3].

Our Approach For comparison, our approach is closer to the competing prover protocols [62] than to the multi-party communication complexity approach [20, 25, 100], due to the way that information is shared among the small number of parties involved (similar to [1, 59]). Also, the study of the DAG evaluation problem (BDEP^k_G) or the Generation Problem might provide the depth lower bounds required by block-respecting simulations [73] (recall the pebbling results in § 1.2). In terms of combinatorial invariants, instead of considering representation-theoretic, algebro-geometric invariants [78, 79], we have been considering enumerative-combinatorial invariants shaped by pebbling strategies [27, 83] on monotone models. Our approach is inspired by the consideration of thrifty branching programs [33].

Chapter 8

Future Directions

We discuss some open problems below.

Problem 8.0.19. Is it possible to connect other resources of the pebble games?

For example, this thesis did not discuss the rounds in the Dymond–Tompa pebble game (or Raz–McKenzie pebble game), or the time in the reversible pebble game. It is of interest, since some resources of (the interpreted variant of) the Dymond–Tompa game [99] capture other computational resources, e. g., bounded alternations.

Problem 8.0.20. Would it help to prove lower bounds by considering the uniformity of the circuits?

It is not hard to see that BDEP_G^k is not solvable by AC^0 circuits when G is the pyramid graph of height $h = n^{\Theta(1)}$. Namely, when $k \approx \frac{1}{4}\log n$ and $h \approx n^{1/4}$, the average sensitivity of BDEP_G^k is $n^{\Theta(1)}$ (while any function computed by AC^0 circuits has average sensitivity $\log^{O(1)} n$ [71]). It follows that BDEP_G^k is not computable by AC^0-uniform AC^0 circuits. Is it possible to relax the uniformity or the complexity of the circuits in this lower bound?

Problem 8.0.21. The Dymond–Tompa game lower bounds the scaling in complexity, for the problem of Generation on monotone switching networks, and for the problem of iterated indexing on output-relevant circuits, over any directed acyclic graph and for a wide range of parameters. To what extent, and on how general a model, does this correspondence hold?

The thrifty hypothesis of Cook, McKenzie, Wehr, Braverman, and Santhanam [33] can be rephrased as the conjecture that this correspondence holds for the black pebble game on the iterated indexing problem over the graph of binary trees, and on the model of branching programs, up to constant factors.

It would be interesting to refute or to establish the optimality of (the interpreted variant of) the Dymond–Tompa pebbling algorithms for space or parallel time: either (1) we get more space-efficient algorithms for graph reachability, or faster parallel speed-up for any

P-complete problem (e. g., linear programming, semi-definite programming, circuit evaluation);[1] or (2) we get very strong complexity results, e. g., $\mathsf{L} \subset \mathsf{NL} \subset \mathsf{NC} \subset \mathsf{P}$ and $\mathsf{NC}^i \subset \mathsf{NC}^{i+1}$, and $\mathsf{DTime}[t] \not\subseteq \mathsf{ATime}\big[o(t/\log t)\big]$.

[1]Note that we consider $\mathsf{ATime}[\cdot]$ as parallel time, so some improvements [72, 103] do not apply.

Appendix A

Bounds on Information

This appendix collects the information theoretic (counting) arguments of Raz and McKenzie [85] used in this work. It may be a good idea to consult [85] (and also [38] that inspired [85], e. g., on the notion of predictability) for the intuition behind the information theoretic arguments (used in the depth lower bounds in restricted models [18, 39, 56, 85]).

Let $X := [K]^\ell$ be an ℓ-fold product space and let C be a subset of X. Given a co-ordinate $j \in [\ell]$, define the bipartite graph $\textsc{Graph}_j(C) := \langle V_L, V_R, E\rangle$, where $V_L := C{\restriction}_j$ and $V_R := C{\restriction}_{[\ell]\setminus j}$ and $(v_L, v_R) \in E$ iff there is a $c \in C$ such that $c{\restriction}_j = v_L \in V_L$ and $c{\restriction}_{[\ell]\setminus j} = v_R \in V_R$.

Definition A.0.22 (Average Degree [85]). Given $j \in [\ell]$, we have $\textsc{Graph}_j(C) = \langle V_L, V_R, E\rangle$ and

$$\textsc{AveDeg}_j(C) := \frac{|E|}{|V_R|} = \frac{|C|}{|C{\restriction}_{[\ell]\setminus j}|} \ .$$

Definition A.0.23 (Min Degree and Thickness [85]). Given $j \in [\ell]$, we have $\textsc{Graph}_j(C) = \langle V_L, V_R, E\rangle$ and

$$\textsc{MinDeg}_j(C) := \min_{v_R \in V_R} \deg(v_R) \ .$$

Note that $\textsc{MinDeg}_j(C) > 0$, by definition of projection. Now

$$\textsc{Thickness}(C) := \min_{j \in [\ell]} \textsc{MinDeg}_j(C) \ .$$

Lemma A.0.24 (Large Size means Large Average Degree [85, Claim 5.1], [38, Lemma 4]). *Let $C' \subseteq C$. Then for any j,*

$$\textsc{AveDeg}_j(C') \geq \frac{|C'|}{|C|} \cdot \textsc{AveDeg}_j(C) \ .$$

Lemma A.0.25 (Entropy Refill). *For any $j \in [\ell]$,*

$$\frac{|C{\restriction}_{[\ell]\setminus j}|}{K^{t-1}} = \frac{|C|}{K^t} \frac{K}{\textsc{AveDeg}_j(C)} \ .$$

Lemma A.0.26 (Dropping Index does not Drop Thickness [85, Claim 5.2]). *For any $j \in [\ell]$,*

$$\textsc{Thickness}(C\restriction_{[\ell]\setminus j}) \geq \textsc{Thickness}(C) \ .$$

Lemma A.0.27 (Distilling Thickness from Average Degree [85, Corollary 5.4]). *Assume that $K \geq \ell^{20}$. If for every j, $\textsc{AveDeg}_j(C) \geq 4 \cdot K^{19/20}$, then there exists $C' \subseteq C$ such that:*

1. $|C'| \geq |C|/2$, *and*
2. $\textsc{Thickness}(C') \geq K^{17/20}$.

Appendix B

Figures

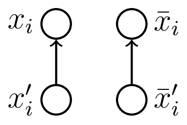

Figure B.1: Variable x_i.

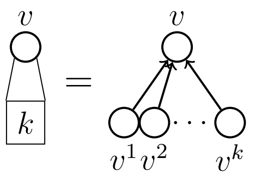

Figure B.2: k extra sources for v.

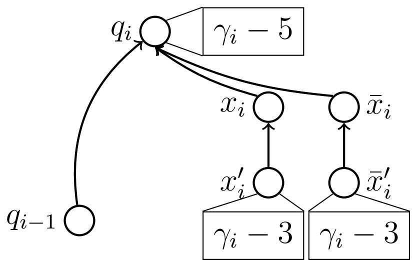

Figure B.3: Existentially quantified variable $\exists x_i$.

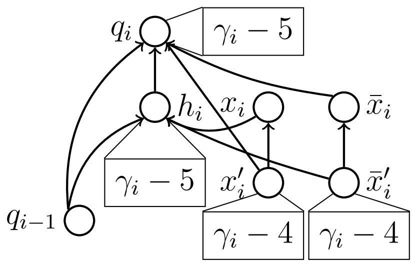

Figure B.4: Universally quantified variable $\forall x_i$.

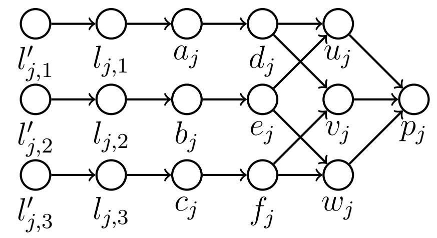

Figure B.5: Clause j.

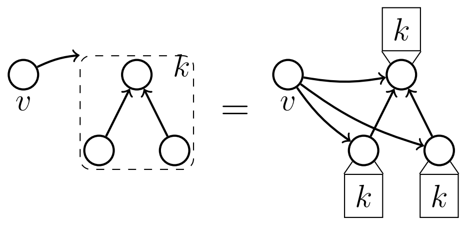

Figure B.6: k extra sources and a common immediate predecessor for all nodes in a dashed region.

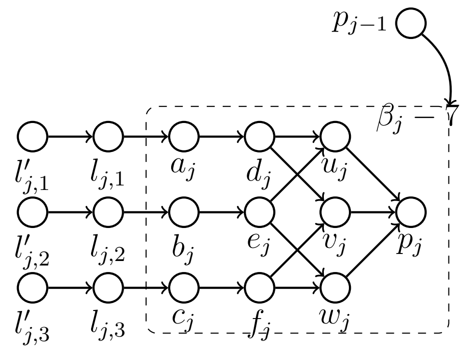

Figure B.7: Augmented gadget for clause j.

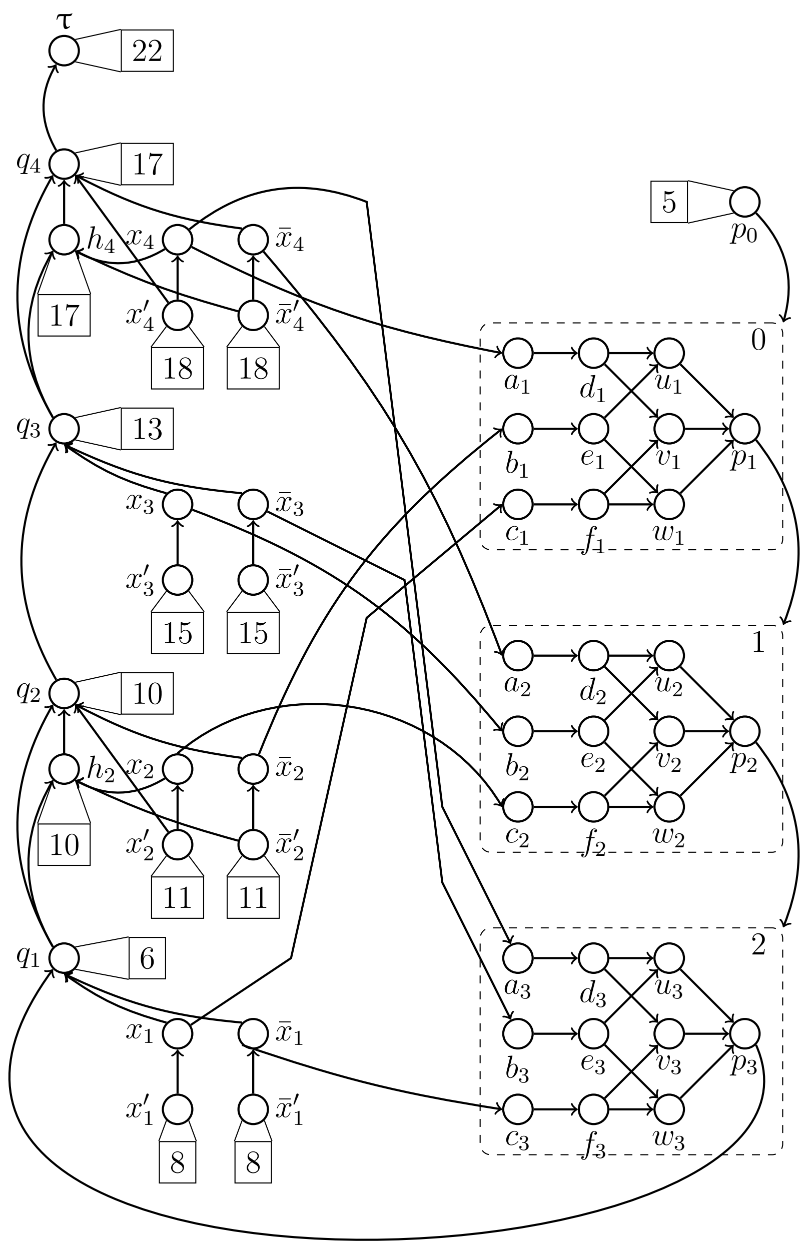

Figure B.8: Example construction for $\varphi := \forall x_4 \exists x_3 \forall x_2 \exists x_1 F$, where $F := (x_4 \vee \bar{x}_2 \vee x_1) \wedge (\bar{x}_4 \vee x_3 \vee x_2) \wedge (x_4 \vee \bar{x}_3 \vee \bar{x}_1)$.

Bibliography

[1] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1):2:1–2:54, February 2009.

[2] Akeo Adachi, Shigeki Iwata, and Takumi Kasai. Some combinatorial game problems require $\Omega(n^k)$ time. *Journal of the ACM*, 31(2):361–376, mar 1984.

[3] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, jan 2002.

[4] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Theory of Computing*, 3(1):81–102, 2007.

[5] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *Journal of the ACM*, 57(3):14:1–14:36, mar 2010.

[6] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[7] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, may 2008.

[8] David A. Mix Barrington and Pierre McKenzie. Oracle branching programs and Logspace versus P. *Information and Computation*, 95(1):96–115, 1991.

[9] Paul Beame, Trinh Huynh, and Toniann Pitassi. Hardness amplification in proof complexity. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 87–96, New York, NY, USA, 2010. ACM.

[10] Paul Beame, Russell Impagliazzo, Toniann Pitassi, and Nathan Segerlind. Formula caching in DPLL. *ACM Transactions on Computation Theory*, 1(3):9:1–9:33, mar 2010.

[11] Eli Ben-Sasson. Size-space tradeoffs for resolution. *SIAM Journal on Computing*, 38(6):2511–2525, jan 2009.

[12] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004.

[13] Eli Ben-Sasson and Jakob Nordström. Understanding space in proof complexity: Separations and trade-offs via substitutions. In Bernard Chazelle, editor, *ICS*, pages 401–416. Tsinghua University Press, 2011.

[14] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, mar 2001.

[15] Charles H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525 –532, nov 1973.

[16] Charles H. Bennett. Time/Space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989.

[17] Christoph Berkholz. On the complexity of finding narrow proofs. In *FOCS*, pages 351–360. IEEE Computer Society, 2012.

[18] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. Exponential separations between restricted resolution and cutting planes proof systems. In *FOCS*, pages 638–647, Palo Alto, California, USA, 1998. IEEE Computer Society.

[19] Ravi B. Boppana and Michael Sipser. The complexity of finite functions. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 757–804. Elsevier and MIT Press, 1990.

[20] Joshua Brody and Amit Chakrabarti. Sublinear communication protocols for multi-party pointer jumping and a related lower bound. In Susanne Albers and Pascal Weil, editors, *25th International Symposium on Theoretical Aspects of Computer Science (STACS 2008)*, volume 1 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 145–165, Dagstuhl, Germany, 2008. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[21] Harry Buhrman, J. Tromp, and Paul Vitanyi. Time and space bounds for reversible simulation. *arXiv:quant-ph/0101133*, jan 2001. Journal of Physics A: Mathematical and General, 34(2001), 6821–6830.

[22] Joshua Buresh-Oppenheim, Matthew Clegg, Russell Impagliazzo, and Toniann Pitassi. Homogenization and the polynomial calculus. *Computational Complexity*, 11(3):91–108, 2002.

[23] Joshua Buresh-Oppenheim, Nicola Galesi, Shlomo Hoory, Avner Magen, and Toniann Pitassi. Rank bounds and integrality gaps for cutting planes procedures. *Theory of Computing*, 2(1):65–90, 2006.

[24] Samuel R. Buss, Stephen A. Cook, Arvind Gupta, and Vijaya Ramachandran. An optimal parallel algorithm for formula evaluation. *SIAM Journal on Computing*, 21(4):755–780, 1992.

[25] Amit Chakrabarti. Lower bounds for multi-player pointer jumping. In *IEEE Conference on Computational Complexity*, pages 33–45. IEEE Computer Society, 2007.

[26] Siu Man Chan. Just a pebble game. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:42, 2013. To appear in the 28th IEEE Conference on Computational Complexity (CCC 2013).

[27] Siu Man Chan and Aaron Potechin. Tight bounds for monotone switching networks via fourier analysis. 2012. Journal version to appear in the Theory of Computing.

[28] Siu On Chan. Approximation resistance from pairwise independent subgroups. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:110, 2012.

[29] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114133, January 1981.

[30] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 174–183, New York, NY, USA, 1996. ACM.

[31] Alan Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the International Conference on Logic, Methodology, and Philosophy of Science*, pages 24–30, 1965.

[32] Stephen A. Cook. An observation on time-storage trade off. *Journal of Computer and System Sciences*, 9:308–316, December 1974.

[33] Stephen A. Cook, Pierre McKenzie, Dustin Wehr, Mark Braverman, and Rahul Santhanam. Pebbles and branching programs for tree evaluation. *ACM Transactions on Computation Theory*, 3(2):4:1–4:43, January 2012.

[34] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, mar 1979.

[35] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, jul 1962.

[36] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, jul 1960.

[37] Patrick W. Dymond and Martin Tompa. Speedups of deterministic machines by synchronous parallel machines. *Journal of Computer and System Sciences*, 30(2):149 – 161, 1985.

[38] Jeff Edmonds, Russell Impagliazzo, Steven Rudich, and Jiri Sgall. Communication complexity towards lower bounds on circuit depth. *Computational Complexity*, 10:210–246, 2001. 10.1007/s00037-001-8195-x.

[39] Yara Elias and Pierre McKenzie. DAG evaluation and the red-blue problem. In *Advances and Applications of Automata on Words and Trees*, 2010. `http://www.dagstuhl.de/Materials/Files/10/10501/10501.McKenziePierre.Slides.pdf`.

[40] Juan Luis Esteban, Nicola Galesi, and Jochen Messner. On the complexity of resolution with bounded conjunctions. *Theoretical Computer Science*, 321(2–3):347–370, aug 2004.

[41] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, nov 2001.

[42] Anna Gál and Jing-Tang Jang. The size and depth of layered boolean circuits. *Information Processing Letters*, 111(5):213–217, feb 2011.

[43] John R. Gilbert, Thomas Lengauer, and Robert Endre Tarjan. The pebbling problem is complete in polynomial space. *SIAM Journal on Computing*, 9(3):513–524, 1980.

[44] Mikael Goldmann and Johan Håstad. A simple lower bound for monotone clique using a communication game. *Information Processing Letters*, 41(4):221 – 226, 1992.

[45] Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory.* Oxford University Press, apr 1995.

[46] Michelangelo Grigni. *Structure in Monotone Complexity.* PhD thesis, Massachusetts Institute of Technology, June 1991.

[47] Michelangelo Grigni and Michael Sipser. Monotone separation of logarithmic space from logarithmic depth. *Journal of Computer and System Sciences*, 50(3):433–437, 1995.

[48] Dima Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theoretical Computer Science*, 259(1–2):613–622, may 2001.

[49] Armin Haken. Counting bottlenecks to show monotone P $\neq$ NP. In *FOCS*, pages 36–40. IEEE Computer Society, 1995.

[50] Johan Håstad and Avi Wigderson. Composition of the universal relation. In *Advances in Computational Complexity Theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 119–134. American Mathematical Society, 1997.

[51] Alexander Hertel and Alasdair Urquhart. Game characterizations and the PSPACE-Completeness of tree resolution space. In Jacques Duparc and Thomas Henzinger, editors, *Computer Science Logic*, volume 4646 of *Lecture Notes in Computer Science*, pages 527–541. Springer Berlin / Heidelberg, 2007.

[52] Philipp Hertel and Toniann Pitassi. The PSPACE-completeness of black-white pebbling. *SIAM Journal on Computing*, 39(6):2622–2682, jan 2010.

[53] John Hopcroft, Wolfgang Paul, and Leslie Valiant. On time versus space. *Journal of the ACM*, 24(2):332–337, apr 1977.

[54] Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space trade-offs in proof complexity. In Howard J. Karloff and Toniann Pitassi, editors, *STOC*, pages 233–248. ACM, 2012.

[55] Matti Järvisalo, Arie Matsliah, Jakob Nordström, and Stanislav Živný. Relating proof complexity measures and practical hardness of SAT. In Michela Milano, editor, *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, pages 316–331. Springer Berlin Heidelberg, jan 2012.

[56] Jan Johannsen. Depth lower bounds for monotone semi-unbounded fan-in circuits. *Theoretical Informatics and Applications*, 35(3):277–286, 2001.

[57] Neil D. Jones and William T. Laaser. Complete problems for deterministic polynomial time. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, pages 40–46, New York, NY, USA, 1974. ACM.

[58] Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3/4):191–204, 1995.

[59] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990.

[60] Takumi Kasai, Akeo Adachi, and Shigeki Iwata. Classes of pebble games and complete problems. *SIAM Journal on Computing*, 8(4):574–586, nov 1979.

[61] Maria Klawe, Wolfgang J. Paul, Nicholas Pippenger, and Mihalis Yannakakis. On monotone formulae with restricted depth. In *STOC*, pages 480–487, New York, NY, USA, 1984. ACM.

[62] Gillat Kol and Ran Raz. Competing provers protocols for circuit evaluation. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 473–484, New York, NY, USA, 2013. ACM.

[63] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1):123–140, 2001.

[64] Richard Královič. Time and space complexity of reversible pebbling. *RAIRO - Theoretical Informatics and Applications*, 38(02):137–161, 2004.

[65] Richard E. Ladner. The circuit value problem is log space complete for p. *SIGACT News*, 7(1):18–20, jan 1975.

[66] Klaus-Jörn Lange, Pierre McKenzie, and Alain Tapp. Reversible space equals deterministic space. *Journal of Computer and System Sciences*, 60(2):354 – 367, 2000.

[67] Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, jan 2001.

[68] Chang-Yeong Lee. Representation of switching circuits by binary-decision programs. *Bell System Technical Journal*, 38(4):985–999, 1959.

[69] Ming Li, John Tromp, and Paul Vitanyi. Reversible simulation of irreversible computation by pebble games. *arXiv:quant-ph/9703009*, mar 1997. Physica D120 (1998) 168-176.

[70] Ming Li and Paul Vitanyi. Reversibility and adiabatic computation: Trading time and space for energy. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1947):769–789, apr 1996.

[71] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM*, 40(3):607–620, jul 1993.

[72] Richard Lipton and Anastasios Viglas. Non-uniform depth of polynomial time and space simulations. In Andrzej Lingas and Bengt Nilsson, editors, *Fundamentals of Computation Theory*, volume 2751 of *Lecture Notes in Computer Science*, pages 323–354. Springer Berlin / Heidelberg, 2003.

[73] Richard J. Lipton and Ryan Williams. Amplifying circuit lower bounds against polynomial time with applications. In *IEEE Conference on Computational Complexity*, pages 1–9, 2012.

[74] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. *SIAM Journal on Discrete Mathematics*, 8(1):119–132, feb 1995.

[75] Pierre McKenzie, 2010. Personal communication.

[76] Ketan D. Mulmuley. Lower bounds in a parallel model without bit operations. *SIAM Journal on Computing*, 28(4):1460–1509, 1999.

[77] Ketan D. Mulmuley. On P vs. NP and geometric complexity theory. *Journal of the ACM*, 58(2):5:1–5:26, apr 2011.

[78] Ketan D. Mulmuley and Milind Sohoni. Geometric complexity theory I: An approach to the P vs. NP and related problems. *SIAM Journal on Computing*, 31(2):496–526, jan 2001.

[79] Ketan D. Mulmuley and Milind Sohoni. Geometric complexity theory II: towards explicit obstructions for embeddings among class varieties. *SIAM Journal on Computing*, 38(3):1175–1206, jan 2008.

[80] Jakob Nordström. Pebble games, proof complexity and time-space trade-offs. 2012. To appear in Logical Methods in Computer Science.

[81] Michael S. Paterson and Carl E. Hewitt. Comparative schematology. In Jack B. Dennis, editor, *Record of the Project MAC conference on concurrent systems and parallel computation*, pages 119–127. ACM, New York, NY, USA, 1970.

[82] Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. Space bounds for a game on graphs. In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, STOC '76, pages 149–160, New York, NY, USA, 1976. ACM.

[83] Aaron Potechin. Bounds on monotone switching networks for directed connectivity. 2010. An updated version to appear in Journal of the ACM.

[84] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-sat. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, SODA '00, pages 128–136, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[85] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999.

[86] Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *Journal of the ACM*, 39(3):736–744, 1992.

[87] Alexander A. Razborov. Lower bounds on the monotone complexity of some boolean functions. *Soviet Mathematics Doklady*, 31(2):354–357, 1985.

[88] Alexander A. Razborov. Lower bounds for deterministic and nondeterministic branching programs. In Lothar Budach, editor, *FCT*, volume 529 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 1991.

[89] Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, aug 1997.

[90] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55:17:1–17:24, September 2008.

[91] Walter Larry Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, jun 1981.

[92] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, apr 1970.

[93] Grant Schoenebeck. Linear level lasserre lower bounds for certain k-CSPs. pages 593–602. IEEE, oct 2008.

[94] Ravi Sethi. Complete register allocation problems. *SIAM Journal on Computing*, 4(3):226–248, sep 1975.

[95] Grigori Samuilovich Tseitin. On the complexity of derivation in propositional calculus. In Anatol Oles'evich Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125. Consultants Bureau, New York, 1970.

[96] Madhur Tulsiani. CSP gaps and reductions in the lasserre hierarchy. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 303–312, New York, NY, USA, 2009. ACM.

[97] Alasdair Urquhart. The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, 1(4):425–467, dec 1995.

[98] Alasdair Urquhart. The depth of resolution proofs. *Studia Logica*, 99(1):349–364, 2011.

[99] H. Venkateswaran and Martin Tompa. A new pebble game that characterizes parallel complexity classes. *SIAM Journal on Computing*, 18(3):533–549, jun 1989.

[100] Emanuele Viola and Avi Wigderson. One-way multiparty communication lower bound for pointer jumping with applications. *Combinatorica*, 29(6):719–743, 2009.

[101] Dustin Wehr. Lower bound for deterministic semantic-incremental branching programs solving GEN. *CoRR*, abs/1101.2705, 2011.

[102] Ryan Williams. Space-efficient reversible simulations. Technical report, 2000.

[103] Ryan Williams. Parallelizing time with polynomial circuits. In *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, SPAA '05, page 171175, New York, NY, USA, 2005. ACM.