Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 97 (2019)

Reversible Pebble Games and the Relation Between Tree-Like and General Resolution Space (Extended Version)*

Jacobo Torán[‡] and Florian Wörz[‡]

Universität Ulm, Germany
{jacobo.toran, florian.woerz}@uni-ulm.de

February 12, 2020

Abstract

We show a new connection between the space measure in tree-like resolution and the reversible pebble game in graphs. Using this connection, we provide several formula classes for which there is a logarithmic factor separation between the space complexity measure in tree-like and general resolution. We show that these separations are not far from optimal by proving upper bounds for tree-like resolution space in terms of general resolution clause and variable space. In particular, we show that for any formula F, its tree-like resolution space is upper bounded by $\mathrm{space}(\pi)\log\big(\mathrm{time}(\pi)\big)$, where π is any general resolution refutation of F. This holds considering as $\mathrm{space}(\pi)$ the clause space of the refutation as well as considering its variable space. For the concrete case of Tseitin formulas, we are able to improve this bound to the optimal bound $\mathrm{space}(\pi)\log n$, where n is the number of vertices of the corresponding graph.

1 Introduction

Resolution is one of the best-studied systems for refuting unsatisfiable propositional formulas. This is due to its theoretical simplicity, as well as its practical importance since it is the proof system at the root of many modern SAT solvers. Several complexity measures for the analysis of resolution refutations have been used in the last decades. In this paper, we will mainly concentrate on space bounds, which measure the amount of memory that is needed in a resolution refutation. Intuitively, the clause space (CS) measures the number of clauses required simultaneously in a refutation, while the variable space (VS) measures the maximum number of distinct variables kept simultaneously in memory during this process. Experimental results have shown that space measures for resolution correlate well with the hardness of refuting unsatisfiable formulas with SAT solvers in practice [ABLM08, JMNŽ12].

Tree-like resolution is a restricted kind of resolution that is especially important since the original DPLL algorithm [DP60, DLL62] on which many SAT solvers are based, is equivalent to

*This is the extended version of the STACS '20 paper. The authors would like to thank the reviewers of the conference for many insightful comments.

‡This research and both authors were supported by the Deutsche Forschungsgemeinschaft (DFG) under project number 430150230, "Complexity measures for solving propositional formulas".

ISSN 1433-8092

this restriction of the resolution system. Contrary to general resolution, in tree-like resolution, if a clause is needed more than once in a refutation, it has to be rederived each time. It is known that general resolution can be exponentially more efficient than tree-like resolution in terms of length (number of clauses in a refutation) [BEGJ98, BIW04]. In [BIW04], the authors give an almost optimal separation between general and tree-like resolution. They show that for each natural number n, there are unsatisfiable formulas in $\mathrm{O}(n)$ variables that have resolution refutations of length L, linear in n, but for which any tree-like resolution refutation of the formula requires length $\exp\left(\Omega(\frac{L}{\log L})\right)$. They also give an almost matching upper bound of $\exp\left(\mathrm{O}(\frac{L \log\log L}{\log L})\right)$ for the tree-like resolution length of any formula that can be refuted in length L by general resolution.

In this paper we study space separations between general and tree-like resolution. Space separations are much more modest than the ones for length. It is known from [ET01] that all space measures considered in this paper for a formula with n variables are between constant and $n + 2$. Also, it is not hard to see that variable space coincides in general and tree-like resolution. Therefore, we only consider the clause space measure for the case of tree-like resolution. The first space separation between general and tree-like resolution was given in [ET03]. There, a family of formulas $(F_n)_{n=1}^{\infty}$ was presented which require tree-resolution clause space s_n but has a general resolution refutation in clause space $c \cdot s_n$, for some constant $c < 1$, where s_n is logarithmic in the number of variables of the formulas. More recently, in [JMNŽ12], a family of formulas $(F_n)_{n=1}^{\infty}$ is presented with $\mathrm{O}(n)$ variables that can be refuted by general resolution in constant clause space but requires $\Theta(\log n)$ tree-like resolution space, thus showing that both measures are fundamentally different.

In this paper, we present a systematic study of tree-like resolution space providing upper bounds for this measure, which show that the logarithmic factor in the separation of [JMNŽ12] as well as in other separations provided here are basically optimal. Our main tools are several versions of pebbling games played on graphs, which have been extensively used in the past for analysing different computation models and in particular for analysing proof systems (see [Nor15] for an excellent survey). We formally define these games in the preliminaries. Intuitively, the idea of the pebble games is to measure the number of pebbles needed by a single player in order to place a pebble on the sink of a directed acyclic graph following certain rules. Black pebbles can only be placed on a vertex if it is a source or if all its direct predecessors already have a pebble, but these pebbles can be removed at any time. White pebbles (modelling non-determinism) can be placed on any vertex at any time but can only be removed if all its direct predecessors contain a pebble. In the reversible pebble game, pebbles can only be placed or removed from a vertex if all the direct predecessors of the vertex contain a pebble. Based on the pebble game, a class of contradictory formulas, called pebbling formulas, was introduced in [BW01]. These formulas have been extremely useful for analysing several proof systems. The reason for this is that some of the pebbling properties of the underlying graphs can be translated into parameters for the complexity of their corresponding pebbling contradictions. Known results of pebbling can therefore be translated into proof complexity results.

The formulas used for the separation between general and tree-like resolution space in [ET03] are pebbling formulas. An examination of this result shows that it relies on the fact that the graphs on which the formulas are based have a black-white pebbling price that is smaller than their black pebbling number. With this observation and using existing separation results for pebble games, the separation in [ET03] can be significantly improved. On the one hand, in [BIW04] the authors implicitly show that for any graph G the tree-like clause space of the pebbling contradiction associated with G is at least as large as the black pebbling number of the graph. On the other hand, Nordström shows in [Nor12] that for most of the graph examples existing in the literature with a difference between their black and back-white pebbling numbers, the resolution clause space of a version of the pebbling contradictions based on the graphs[1], is upper bounded by the black-white pebbling number of the graphs. Putting these two facts together, it follows that there are unsatisfiable formulas that have resolution clause space $\mathrm{O}(s)$ (logarithmic in the number of variables of the formulas) while their tree-like resolution clause space is lower bounded by $\Omega(s^2)$. This is the largest separation that can be obtained using this method since it is known that the difference between the black and black-white pebbling number of any graph is at most quadratic [Mey81], and can therefore not explain the logarithmic factor in the separation in [JMNŽ12] where the (pebbling) formulas have constant general resolution clause space.

Our main contribution is a new connection between tree-like resolution clause space and the reversible pebble game. We show that for any graph G, the tree-like resolution space of a (certain kind of) pebbling contradiction [2] of the graph is at least the reversible pebbling number of G and at most twice this number. More interestingly, we show that for *any* unsatisfiable CNF formula F, the tree-like resolution clause space of a refutation of F is at most the reversible pebbling number of any refutation graph of F, not necessarily a tree-like refutation. This result adds one more connection to the rich set of interrelations between pebbling and resolution [Nor15]. A central tool in the proofs of these results is the Raz–McKenzie game [RM99], a two-player game on graphs, and the fact that this game is equivalent to reversible pebbling in a precise sense [Cha13]. The clause space measure for any formula can be exactly characterised in terms of the black pebble game on a refutation graph of the formula [ET01]. We find the fact that tree-like clause space is upper bounded by the reversible pebble game quite surprising.

Using these bound and known results on reversible pebbling [CLNV15, Vin17], we show in Section 4 that there are families of pebbling formulas $(F_n)_{n=1}^{\infty}$ with $\mathrm{O}(n)$ variables, that have general clause space $\mathrm{O}(s)$ and tree-like resolution space $\Omega(s \log n)$ for any function s smaller than $n^{1/2-\varepsilon}$. This separation (as well as the one in [JMNŽ12]) is almost optimal since we also show that for any pebbling formula F, its tree-like clause space is at most $\min_{\mathcal{P}} (\mathsf{space}(\mathcal{P}) \cdot \log \mathsf{time}(\mathcal{P}))$, where $\mathcal{P}$ is a black pebbling of the underlying graph of F. This means that for graphs of size n where the smallest black pebbling space is achieved in a one-shot pebbling strategy, that is, a strategy in which every vertex in the graph is pebbled at most once, the $\log n$ factor in the separation is optimal and the only room for improvement is with graph families in which the space-optimal black pebbling is not one-shot. It is possible that for one such family, the $\log n$ separation factor can be improved to a $\log \mathsf{time}(\mathcal{P})$ factor. We provide, however, for the first time a family of graphs for which the minimum pebbling space is obtained in a strategy that is not one-shot, but for which the clause space separation between general and tree-like resolution is also only a $\log n$ factor. We conjecture that this is optimal, and that this separation cannot be improved for other graph classes. This question is closely related to proving optimal upper bounds for reversible pebbling in terms of black pebbling. Another motivation for providing this new graph family is to increase the set of examples of formulas with concrete resolution space bounds that can be used for the testing of SAT solvers, as done for example in [JMNŽ12].

In Section 5, we prove upper bounds on the tree-like clause space for *any* unsatisfiable CNF formula F in terms of the variable space and clause space for general resolution of the formula. We use the *amortised space measures* for resolution introduced by Razborov in [Raz18], that penalise configurational proofs for being unreasonably long. In his paper he defined the notations $\mathrm{VS}^*(F \vdash \Box) := \min_{\pi : F \vdash \Box} (\mathrm{VS}(\pi) \cdot \log \mathrm{L}(\pi))$ as well as $\mathrm{CS}^*(F \vdash \Box) := \min_{\pi : F \vdash \Box} (\mathrm{CS}(\pi) \cdot \log \mathrm{L}(\pi))$, where $\mathrm{L}(\pi)$ is the length of the configurational proof π. We show

[1] More precisely, the second degree XORification of the pebbling contradiction over the graph as defined in the Preliminaries.

[2] More precisely, the second degree XORification of the pebbling contradiction over the graph as defined in the Preliminaries.

the upper bounds Tree-CS$(F \vdash \Box) \leq \mathrm{VS}^*(F \vdash \Box) + 2$ and Tree-CS$(F \vdash \Box) \leq \mathrm{CS}^*(F \vdash \Box) + 2$. The first inequality is especially interesting since it shows that clause space can be meaningfully bounded in terms of variable space, a question posed by Razborov in [Raz18]: $\mathrm{CS}(F \vdash \Box) \leq \mathrm{VS}^*(F \vdash \Box) + 2$. Again, from the separations in Sections 4 and 6, the only room for improvement in this upper bounds is to decrease the $\log \mathrm{L}(\pi)$ factor to a $\log n$ factor, where n is the size of the formula F.

Finally, in Section 6, we give optimal separations for the space in tree-like resolution for the class of Tseitin formulas. We show that for any graph G with n vertices and odd marking χ, the inequalities Tree-CS$(\mathrm{Ts}(G,\chi) \vdash \Box) \leq \mathrm{CS}(\mathrm{Ts}(G,\chi) \vdash \Box) \cdot \log n + 2$ and Tree-CS$(\mathrm{Ts}(G,\chi) \vdash \Box) \leq \mathrm{VS}(\mathrm{Ts}(G,\chi) \vdash \Box) \cdot \log n + 2$ hold, thus improving the upper bound from the previous sections from logarithmic in the resolution length down to a $\log n$ factor. We also provide a class of formulas with a matching space separation showing that this is optimal.

2 Preliminaries

For a positive integer n we write $[n]$ to denote the set of integers $\{1, 2, \ldots, n\}$. The base of all logarithms in this paper is 2. The *size of a graph* is the number of vertices of the graph. Given a directed acyclic graph (*DAG*) $G = (V, E)$, we say that a vertex u is a *direct predecessor* of a vertex v, if there exists a directed edge from u to v. We denote by $\mathrm{pred}_G(v)$ the *set of all direct predecessors* of v in G. The *maximal in-degree* of a graph G is defined to be $\max_{v \in V} |\mathrm{pred}_G(v)|$. A vertex in a DAG with no incoming edges is called a *source* and a vertex with no outgoing edges is called a *sink*.

2.1 Pebble Games

Black pebbling was first mentioned implicitly in [PH70], while black-white pebbling was introduced in [CS76] and has been studied extensively during the 1980s.

Note, that there exist several variants of the (black-white) pebble game in the literature. In this paper, we focus on the variant without *sliding* and requiring the sink of the graph to be pebbled at the end. For differences between these variants, we refer to the survey [Nor15], from which we borrowed most of our notation. For the following definitions, let $G = (V, E)$ be a DAG with a unique sink vertex z.

Definition 1 (Black and black-white pebble games)**.** The *black-white pebble game* on G is the following one-player game: At any time i of the game, we have a *pebble configuration* $\mathbb{P}_i := (B_i, W_i)$, where $B_i \cap W_i = \varnothing$ and $B_i \subseteq V$ is the set of black pebbles and $W_i \subseteq V$ is the set of white pebbles, respectively. A pebble configuration $\mathbb{P}_{i-1} = (B_{i-1}, W_{i-1})$ can be changed to $\mathbb{P}_i = (B_i, W_i)$ by applying exactly one of the following rules:

Black pebble placement on v: If all direct predecessors of an empty vertex v have pebbles on them, a black pebble may be placed on v. More formally, letting $B_i = B_{i-1} \cup \{v\}$ and $W_i = W_{i-1}$ is allowed if $v \notin B_{i-1} \cup W_{i-1}$ and $\mathrm{pred}_G(v) \subseteq B_{i-1} \cup W_{i-1}$. In particular, a black pebble can always be placed on an empty source vertex s, since $\mathrm{pred}_G(s) = \varnothing$.

Black pebble removal from v: A black pebble may be removed from any vertex at any time. Formally, if $v \in B_{i-1}$, then we can set $B_i = B_{i-1} \setminus \{v\}$ and $W_i = W_{i-1}$.

White pebble placement on v: A white pebble may be placed on any empty vertex at any time. Formally, if $v \notin B_{i-1} \cup W_{i-1}$, then we can set $B_i = B_{i-1}$ and $W_i = W_{i-1} \cup \{v\}$.

White pebble removal from v: If all direct predecessors of a white-pebbled vertex v have pebbles on them, the white pebble on v may be removed. Formally, letting $B_i = B_{i-1}$ and

$W_i = W_{i-1} \setminus \{v\}$ is allowed if $v \in W_{i-1}$ and $\mathrm{pred}_G(v) \subseteq B_{i-1} \cup W_{i-1}$. In particular, a white pebble can always be removed from a source vertex.

A *black-white pebbling* of G is a sequence of pebble configurations $\mathcal{P} = (\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_t)$ such that $\mathbb{P}_0 = (\varnothing, \varnothing)$, $\mathbb{P}_t = (\{z\}, \varnothing)$, and for all $i \in [t]$ it holds that $\mathbb{P}_i$ can be obtained from $\mathbb{P}_{i-1}$ by applying exactly one of the above-stated rules.

A *black pebbling* is a pebbling where $W_i = \varnothing$ for all $i \in [t]$.

Definition 2 (Pebbling time, space, and price)**.** The *time* of a pebbling $\mathcal{P} = (\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_t)$ is $\mathsf{time}(\mathcal{P}) := t$ and the *space* of it is $\mathsf{space}(\mathcal{P}) := \max_{i \in [t]} |B_i \cup W_i|$. The *black-white pebbling price* (also known as the *pebbling measure* or *pebbling number*) of G, which we will denote by $\mathsf{BW}(G)$, is the minimum space of any black-white pebbling of G. The *(black) pebbling price* of G, denoted by $\mathsf{Black}(G)$, is the minimum space of any black pebbling of G.

Observation 3 (Trivial pebbling, [Nor15])**.** Any DAG G has a black pebbling in space at most $|V(G)|$ and time at most $2 \cdot |V(G)|$ simultaneously.

Definition 4 (One-shot pebbling)**.** A pebbling is *one-shot* if each $v \in V$ is pebbled at most once.

Finally, we mention the reversible pebble game introduced in [Ben89]. In the reversible pebble game, the moves performed in reverse order should also constitute a legal black pebbling, which means that the rules for pebble placements and removals have to become symmetric.

Definition 5 (Reversible pebble game)**.** The *reversible pebble game* on G is the following one-player game: At any time i of the game, we have a pebble configuration $\mathbb{P}_i \subseteq V$. A pebble configuration $\mathbb{P}_{i-1}$ can be changed to $\mathbb{P}_i$ by applying exactly one of the following rules:

Pebble placement on v: If all direct predecessors of an empty vertex v have pebbles on them, a pebble may be placed on v. More formally, letting $\mathbb{P}_i = \mathbb{P}_{i-1} \cup \{v\}$ is allowed if $v \notin \mathbb{P}_{i-1}$ and $\mathrm{pred}_G(v) \subseteq \mathbb{P}_{i-1}$. In particular, a pebble can always be placed on an empty source vertex s, since $\mathrm{pred}_G(s) = \varnothing$.

Reversible pebble removal from v: If all direct predecessors of a pebbled vertex v have pebbles on them, the pebble on v may be removed. Formally, letting $\mathbb{P}_i = \mathbb{P}_{i-1} \setminus \{v\}$ is allowed if $v \in \mathbb{P}_{i-1}$ and $\mathrm{pred}_G(v) \subseteq \mathbb{P}_{i-1}$. In particular, a pebble can always be removed from a source vertex.

A *reversible pebbling* of G is a sequence of pebble configurations $\mathcal{P} = (\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_t)$ such that $\mathbb{P}_0 = \varnothing$, $\mathbb{P}_t = \{z\}$, and for all $i \in [t]$ it holds that $\mathbb{P}_i$ can be obtained from $\mathbb{P}_{i-1}$ by applying exactly one of the above-stated rules.

Definition 6 (Reversible pebbling time, space, and price)**.** The *time* of a reversible pebbling $\mathcal{P} = (\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_t)$ is $\mathsf{time}(\mathcal{P}) := t$ and the *space* of it is $\mathsf{space}(\mathcal{P}) := \max_{i \in [t]} |\mathbb{P}_i|$. The *reversible pebbling price* of G, which we will denote by $\mathsf{Rev}(G)$, is the minimum space of any reversible pebbling of G.

2.2 Resolution

A *literal* over a Boolean variable x is either x itself (also denoted as x^1) or its negation $\overline{x}$ (also denoted as x^0). A *clause* $C = a_1 \vee \cdots \vee a_\ell$ is a (possibly empty) disjunction of literals a_i over pairwise disjoint variables. The *set of variables occurring in a clause* C will be denoted by $\mathrm{Vars}(C)$. A clause C is called *unit* if $|\mathrm{Vars}(C)| = 1$. We let $\Box$ denote the contradictory *empty clause* (the clause without any literals). A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. It is often advantageous to think of clauses and CNF formulas as sets. The notion of the set of variables in a clause is extended to CNF formulas by taking unions. A CNF formula is a k-*CNF*, if all clauses in it have at most k variables. An *assignment/restriction* α for a CNF formula F is a function that maps some subset of $\mathrm{Vars}(F)$ to $\{0, 1\}$. It is applied to F, which we denote by $F\restriction_\alpha$, in the usual way (see e. g. [BW01, ST13]). We denote the *empty assignment* with $\varnothing$.

The standard definition of a *resolution derivation* of a clause D from a CNF formula F (denoted by $\pi : F \vdash D$) is an ordered sequence of clauses $\pi = (C_1, \ldots, C_t)$ such that $C_t = D$, and each clause C_i, for $i \in [t]$, is either an *axiom clause* $C_i \in F$ or is derived from clauses C_j and C_k with $j, k < i$ by the *resolution rule*

$$\frac{B \vee x \qquad C \vee \overline{x}}{B \vee C}\,. \tag{1}$$

In the resolution rule (1), we call $B \vee x$ and $C \vee \overline{x}$ the *parents* and $B \vee C$ the *resolvent*. A derivation $\pi : F \vdash \Box$ of the empty clause from an unsatisfiable CNF formula F is called *refutation*. Note, that resolution is a sound and complete proof system for unsatisfiable formulas in CNF.

To study space in resolution, we consider the following definitions of the resolution proof system from [ET01, ABRW02].

Definition 7 (Configuration-style resolution)**.** A *resolution refutation* $\pi : F \vdash \Box$ of an unsatisfiable CNF formula F is an ordered sequence of *memory configurations* (sets of clauses) $\pi = (\mathbb{M}_0, \ldots, \mathbb{M}_t)$ such that $\mathbb{M}_0 = \varnothing$, $\Box \in \mathbb{M}_t$ and for each $i \in [t]$, the configuration $\mathbb{M}_i$ is obtained from $\mathbb{M}_{i-1}$ by applying exactly one of the following rules:

Axiom Download: $\mathbb{M}_i = \mathbb{M}_{i-1} \cup \{C\}$ for some axiom clause $C \in F$.

Erasure: $\mathbb{M}_i = \mathbb{M}_{i-1} \setminus \{C\}$ for some $C \in \mathbb{M}_{i-1}$.[3]

Inference: $\mathbb{M}_i = \mathbb{M}_{i-1} \cup \{D\}$ for some resolvent D inferred from $C_1, C_2 \in \mathbb{M}_i$ by the resolution rule (1).

The proof π is said to be *tree-like*, if we replace the inference rule with the following rule [ET01]:

Tree-like Inference: $\mathbb{M}_i = (\mathbb{M}_{i-1} \cup \{D\}) \setminus \{C_1, C_2\}$ for some resolvent D inferred from $C_1, C_2 \in \mathbb{M}_i$ by the resolution rule (1), i. e., we delete both parent clauses immediately.

To every configurational refutation π we can associate a *refutation-DAG* G_π, with the clauses of the refutation labelling the vertices of the DAG and with edges from the parents to the resolvent for each application of the resolution rule (1). There might be several different derivations of a clause C during the course of the refutation, but if so, we can label each occurrence of C with a timestamp when it was derived and keep track of which copy of C is used where (cf. [Nor15]). Using this representation, if π is tree-like, then G_π is a tree.

Definition 8 (Complexity measures for resolution)**.** The *length*[4] of a resolution refutation $\pi = (\mathbb{M}_0, \ldots, \mathbb{M}_t)$ is defined to be $\mathrm{L}(\pi) := t$.

[3] In some publications, the authors allow for subsets of the previous memory configuration to be erased. We will not allow this, since our version is more suitable when working with pebbling. Note, that not allowing subset-erasures can at most double the amount of configurations in a refutation. See also footnote 4.

[4] Note, that in the literature, the length of a proof π is sometimes defined to be the total number of axiom downloads and inferences made in π, i. e., the total number of clauses counted with repetitions. We, however, also consider the amount of erasure steps, since this is more natural when working with pebbling. Counting the erasure steps can, however, only increase the length measure by a factor of 2, since every clause being deleted has to be downloaded or inferred prior to its deletion and thus was already counted once in the length measure.

The *clause space* of a memory configuration $\mathbb{M}$ is defined as $\mathrm{CS}(\mathbb{M}) := |\mathbb{M}|$, i. e., the number of clauses in $\mathbb{M}$. The *variable space* of a memory configuration $\mathbb{M}$ is defined as $\mathrm{VS}(\mathbb{M}) := |\bigcup_{C\in\mathbb{M}} \mathrm{Vars}(C)|$, i.e., the number of distinct variables mentioned in $\mathbb{M}$.[5]

The clause space (variable space) of a refutation $\pi = (\mathbb{M}_0, \ldots, \mathbb{M}_t)$ is defined by $\mathrm{CS}(\pi) := \max_{i\in[t]} \mathrm{CS}(\mathbb{M}_i)$ and $\mathrm{VS}(\pi) := \max_{i\in[t]} \mathrm{VS}(\mathbb{M}_i)$, respectively.

Taking the minimum over all refutations of a formula F, we define $\mathrm{L}(F \vdash \Box) := \min_{\pi:F\vdash\Box} \mathrm{L}(\pi)$, $\mathrm{CS}(F \vdash \Box) := \min_{\pi:F\vdash\Box} \mathrm{CS}(\pi)$ and $\mathrm{VS}(F \vdash \Box) := \min_{\pi:F\vdash\Box} \mathrm{VS}(\pi)$ as the length, clause space and variable space of refuting F in resolution, respectively. We define Tree-$\mathrm{CS}(F \vdash \Box) := \min_{\pi':F\vdash\Box} \mathrm{CS}(\pi')$, where the minimum is taken over all tree-like refutations π' of the formula F.

Proposition 9 ([ET01])**.** *Let F be an unsatisfiable formula. Then it holds $\mathrm{CS}(F \vdash \Box) = \min_{\pi:F\vdash\Box} \mathsf{Black}(G_\pi)$.*

Razborov introduced *amortised space measures* for resolution in [Raz18], that penalise configurational proofs for being unreasonably long.

Definition 10 (Amortised space measures for resolution)**.** The *amortised clause space* (*amortised variable space*) of a resolution refutation π is defined by $\mathrm{CS}^*(\pi) := \mathrm{CS}(\pi) \cdot \log \mathrm{L}(\pi)$ and $\mathrm{VS}^*(\pi) := \mathrm{VS}(\pi) \cdot \log \mathrm{L}(\pi)$, respectively.

Taking the minimum over all resolution refutations of a formula F, we define $\mathrm{CS}^*(F \vdash \Box) := \min_{\pi:F\vdash\Box} \mathrm{CS}^*(\pi)$ and $\mathrm{VS}^*(F \vdash \Box) := \min_{\pi:F\vdash\Box} \mathrm{VS}^*(\pi)$.

2.3 Formula Families

Pebbling Formulas and Their XORification

In the last years, there has been renewed interest in pebbling in the context of proof complexity. This is so, because pebbling results can be partially translated into proof complexity results by studying so-called pebbling formulas [BW01, BN11]. These are unsatisfiable CNF formulas encoding the pebble game played on a DAG G. We define them next.

Definition 11 (Pebbling formulas)**.** Let $G = (V, E)$ be a DAG with a set of sources $S \subseteq V$ and a unique sink z. We identify every vertex $v \in V$ with a Boolean variable v. The *pebbling contradiction* over G, denoted Peb_G, is the conjunction of the following clauses:

- for all sources $s \in S$, a unit clause s, (*source axioms*)
- for all non-source vertices v, the clause $\bigvee_{u\in\mathrm{pred}_G(v)} \overline{u} \vee v$, (*pebbling axioms*)
- for the unique sink z, the unit clause $\overline{z}$. (*sink axiom*)

Often, it turns out, that the formulas in Definition 11 are a bit too easy to refute. A good way to make them slightly harder is to substitute some suitable Boolean function $f(x_1, \ldots, x_d)$ of arity d for each variable x and expand the result into CNF. This general case is discussed in [Nor15]. We restrict ourselves to the special case of the second degree XORification.

For notational convenience, we assume that the formula F we are trying to make harder only has variables x, y, z, et cetera, without subscripts, so that $x_1, x_2, y_1, y_2, z_1, z_2$, et cetera, are new variables not occurring in F.

[5]The term variable space was used for different concepts in proof complexity. Following the (now established) definition, we refer to the total number of literals in a memory configuration counted with repetitions as *total space*.

Definition 12 (Substitution formulas, [BN08])**.** For a positive literal x define the *XORification* of x to be $x[\oplus_2] := \{x_1 \vee x_2, \overline{x_1} \vee \overline{x_2}\}$. For a negative literal $\overline{y}$, the XORification is $\overline{y}[\oplus_2] := \{y_1 \vee \overline{y_2}, \overline{y_1} \vee y_2\}$. The XORification of a clause $C = a_1 \vee \cdots \vee a_k$ is the CNF formula

$$C[\oplus_2] := \bigwedge_{C_1 \in a_1[\oplus_2]} \cdots \bigwedge_{C_k \in a_k[\oplus_2]} (C_1 \vee \cdots \vee C_k)$$

and the XORification of a CNF formula F is $F[\oplus_2] := \bigwedge_{C \in F} C[\oplus_2]$.

Remark 13 ([BN08])**.** If G has n vertices and maximal in-degree ℓ, then $\mathrm{Peb}_G[\oplus_2]$ is an unsatisfiable $2(\ell+1)$-CNF formula with at most $2^{\ell+1} \cdot n$ clauses over $2n$ variables.

Definition 14 (Non-authoritarian function, [BN11])**.** A Boolean function $f(x_1, \ldots, x_d)$ is called *k-non-authoritarian* if no restriction ρ to $\{x_1, \ldots, x_d\}$ of size $|\rho| \leq k$ can fix the value of f.

Tseitin Formulas

Tseitin formulas encode the combinatorial principle that for all graphs the sum of the degrees of the vertices is *even*. This class of formulas was introduced in [Tse68] and has been extremely useful for the analysis of proof systems.

Definition 15 (Tseitin formulas)**.** Let $G = (V, E)$ be a connected undirected graph and let $\chi \colon V \to \{0, 1\}$ be a *marking* of the vertices of G. A marking χ is called *odd* if it satisfies the property $\sum_{v \in V} \chi(v) \equiv 1 \pmod 2$ otherwise it is called *even*. Associate to every edge $e \in E$ a propositional variable e. The CNF formula $\mathrm{PARITY}_{v,\chi(v)}$ states that the parity of the values of the edges that have vertex v as endpoint coincides with $\chi(v)$, i. e.,

$$\mathrm{PARITY}_{v,\chi(v)} := \bigwedge \left\{ \bigvee_{e \ni v} e^{a(e)} \;:\; a(e) \in \{0,1\}, \text{ such that } \bigoplus_{e \ni v} (a(e) \oplus 1) \not\equiv \chi(v) \right\}.$$

Then, the *Tseitin formula* associated to the graph G and the marking χ is the CNF formula defined by $\mathrm{Ts}(G, \chi) := \bigwedge_{v \in V} \mathrm{PARITY}_{v,\chi(v)}$.

For a partial truth assignment α, applying α to $\mathrm{Ts}(G, \chi)$ corresponds to the following simplification of the underlying graph: Setting a variable $e = \{u, v\}$ to 0 corresponds to deleting the edge e in the graph, and setting it to 1 corresponds to deleting the edge from the graph and toggling the value of $\chi(u)$ and $\chi(v)$ in G. We denote by $G\restriction_\alpha$ and by $\chi\restriction_\alpha$ the remaining graph and marking after applying α according to this process.

Fact 16 ([Tse68, Urq87, ET01])**.** *Let χ be an odd marking of of a connected graph G and e an edge in G that, when deleted divides G in two connected components G_1 and G_2. Then for $i \in \{1, 2\}$ there is a partial assignment α_i of variable e so that $\chi\restriction_{\alpha_i}$ is an odd marking of G_i.*

2.4 Combinatorial Games for Tree-Like Clause Space in Resolution

Important tools for our results are two two-player combinatorial games. The Prover-Delayer game is played on *formulas* and was introduced in [PI00] in order to prove lower bounds for tree-like resolution length. Later it was shown in [ET03] that the game exactly characterises tree-like resolution space. The Raz–McKenzie game is played on *DAGs* and was introduced in [RM99] as a tool for studying the depth complexity of decision trees for search problems.

Definition 17 (Prover-Delayer game)**.** The *Prover-Delayer game*, as described in [PI00, ET03, BIW04], is a combinatorial game between two players, called *Prover* (he), and *Delayer* (she), played on an unsatisfiable CNF formula F. The goal of Prover is to falsify some initial clause of F, which he can always achieve, since the formula is unsatisfiable; however, Delayer tries to retard this as much as possible. The game is played in rounds. Each round starts with Prover querying the value of a variable. Delayer can give one of three answers: 0, 1, or $*$. If 0 or 1 is chosen by Delayer, no points are scored by her and the queried variable is set to the chosen bit. If Delayer answers $*$, then Prover gets to decide the value of that variable, and Delayer scores one point. This is the only way in which points can be scored. The game finishes when any clause in F has been falsified (all its literals are set to 0) by the partial assignment constructed this way. If this is not the case, the next round begins. The aim of Delayer is to win as many points as possible, while Prover aims to minimise this quantity.

Definition 18 (Game value of the Prover-Delayer game)**.** Let F be an unsatisfiable CNF formula. The game value of the Prover-Delayer game played on F, denoted by $\mathsf{PD}(F)$, is the greatest number of points Delayer can score on F against an optimal strategy of Prover.

The Prover-Delayer game exactly characterises the tree-like clause space of a formula. The constant term of the original result in [ET03, Theorem 2.2] was slightly modified to match our definitions of clause space and the pebble game (without sliding).

Theorem 19 ([ET03])**.** *Let F be an unsatisfiable CNF formula. Then*

$$\text{Tree-CS}(F \vdash \Box) = \mathsf{PD}(F) + 2.$$

Definition 20 (Raz–McKenzie game)**.** The *Raz–McKenzie game* is played on a single-sink DAG G by two players, *Pebbler* and *Colourer*. The game is played in rounds. In the first round, Pebbler places a pebble on the sink and Colourer colours the pebble red. In all subsequent rounds, Pebbler places a pebble on an arbitrary empty vertex of G and Colourer colours this new pebble either red or blue. The game ends when there is a vertex with a red pebble that is either a source vertex or all its direct predecessors in the graph have blue pebbles.

Definition 21 (Raz–McKenzie price)**.** The *Raz–McKenzie price* $\mathsf{R\text{-}Mc}(G)$ of a single sink DAG G is the smallest number r such that Pebbler has a strategy to make the game end in at most r rounds against an optimal strategy of Colourer.

In [Cha13] it was shown that the reversible pebbling price and the Raz–McKenzie price coincide for any single-sink DAG.

Theorem 22 ([Cha13])**.** *For any single-sink DAG G we have* $\mathsf{R\text{-}Mc}(G) = \mathsf{Rev}(G)$.

3 Separations From Known Pebbling Results

Using some known results, we show that a separation between the black and black-white pebbling price of a graph can lead to a separation between the space in tree-like and general resolution for the corresponding pebbling formulas. Then we present some pebbling results where these separations are achieved.

In [BIW04], the following result for the $\vee_2$ substitution formulas was proven (with a different additive constant). It is not hard to see that the result also holds for the $\oplus_2$ function.

Theorem 23. *For any DAG G it holds* $\mathsf{Black}(G) - 1 \leq \text{Tree-CS}(\text{Peb}_G[\oplus_2] \vdash \Box)$.

The next result is considered as folklore. The idea behind it is that the pebbling formula can be resolved following the order in which the vertices of the graph are being pebbled. The constant in the O-notation depends on the maximal in-degree of the graph.

Theorem 24. *For any DAG G it holds* $\mathrm{CS}(\mathrm{Peb}_G[\oplus_2] \vdash \Box) = \mathrm{O}(\mathsf{Black}(G))$.

For the examples of graph families stated bellow, for which separations between the black and black-white pebbling prices are known, Nordström showed in [Nor12, Theorems 1.6 and 1.8] that the clause space of their corresponding pebbling formulas is upper bounded by the black-white pebbling price of the graphs.

Theorem 25 ([KS91])**.** *There is a family $(G_s)_{s=1}^{\infty}$ of bounded in-degree DAGS whose size is polynomial in s such that* $\mathsf{BW}(G_s) = \mathrm{O}(s)$ *but* $\mathsf{Black}(G_s) = \Omega(\frac{s \log s}{\log \log s})$.

Kalyanasundaram and Schnitger [KS91] improved this to a quadratic separation.

Theorem 26 ([KS91])**.** *There is a family $(G_s)_{s=1}^{\infty}$ of bounded in-degree DAGs whose size is* $\exp(\Theta(s \log s))$ *such that* $\mathsf{BW}(G_s) \leq 3s + 1$ *but* $\mathsf{Black}(G_s) \geq s^2$.

Note, however, that the graphs yielding the optimal quadratic separation are not of size polynomial in s, as opposed to the first result that holds for polynomial-size graphs. Nordström showed that for the pebbling formulas of these graphs families, resolution has the strength of black-white pebbling.

Theorem 27 ([Nor12])**.** *For any graph G belonging to the two mentioned graph families from Kalyanasundaram and Schnitger,* $\mathrm{CS}(\mathrm{Peb}_G[\oplus_2] \vdash \Box) \leq \mathsf{BW}(G)$.

This means that for the mentioned graph examples, the black pebbling price is a lower bound for the tree resolution space of the corresponding formula while the black-white pebbling price is an upper bound for the general resolution clause space. Putting these results together we obtain:

Corollary 28. *There is a family of unsatisfiable formulas $(F_s)_{s=1}^{\infty}$ of size polynomial in s such that* $\mathrm{CS}(F_s \vdash \Box) = \mathrm{O}(s)$ *but* $\text{Tree-CS}(F_s \vdash \Box) = \Omega(\frac{s \log s}{\log \log s})$.

Corollary 29. *There is a family of unsatisfiable formulas $(F_s)_{s=1}^{\infty}$ of DAGs of size* $\exp(\Theta(s \log s))$ *such that* $\mathrm{CS}(F_s \vdash \Box) = \mathrm{O}(s)$ *but* $\text{Tree-CS}(F_s \vdash \Box) = \Omega(s^2)$.

These are the best separations that can be obtained using this method, since it was proved in [Mey81] that the difference between the black and black-white pebbling price of any DAG can be at most quadratic. In the next sections we show better separations by using a new connection between tree-like resolution clause space and the reversible pebble game.

4 Separations Between Tree-Like and General Resolution Space for Pebbling Formulas Using the Raz–McKenzie Game

We will now establish a connection between tree-like clause space in resolution and the Raz–McKenzie price. We simplify the proof by following the intuition behind the game and identify the colour blue with 1 and the colour red with 0.

Theorem 30. *For any single-sink DAG G it holds*

$$\mathsf{R\text{-}Mc}(G) + 2 \leq \text{Tree-CS}(\mathrm{Peb}_G[\oplus_2] \vdash \Box) \leq 2 \cdot \mathsf{R\text{-}Mc}(G) + 2.$$

Proof. Let G be a fixed DAG with a unique sink. We prove that $\mathsf{R\text{-}Mc}(G) \leq \mathsf{PD}(\mathrm{Peb}_G[\oplus_2])$ and $\mathsf{PD}(\mathrm{Peb}_G[\oplus_2]) \leq 2 \cdot \mathsf{R\text{-}Mc}(G)$. The result then follows from Theorem 19.

(1) We first show the inequality $\mathsf{PD}(\mathrm{Peb}_G[\oplus_2]) \leq 2 \cdot \mathsf{R\text{-}Mc}(G) =: 2r$ by giving a strategy for Prover, such that Delayer can score at most $2r$ points. Prover basically simulates the strategy of Pebbler in the Raz–McKenzie game: If Pebbler pebbles a vertex v of G, Prover will query the variables v_1 and v_2 of $\mathrm{Peb}_G[\oplus_2]$ in this order. The Raz–McKenzie game ends after at most r rounds. We will argue, that the Prover-Delayer game also ends after at most $2r$ queries. Thus, Delayer only gets a chance to score $2r$ points (if a variable pair gets queried for the first time, she can always answer $*$; only the second variable of the pair matters due to the XORification). In case the second variable of a pair gets queried, the best choice Delayer has is to follow the strategy of Colourer (Colourer is following an optimal strategy, thus, if Delayer had a better answer, this would correspond to a better answer for Colourer) and to ensure that $v_1 \oplus v_2$ is true under her constructed assignment if v is coloured 1; and false if v is coloured 0. At the end of the Raz–McKenzie game either a source vertex s in G is coloured 0, or a vertex v of G is coloured 0, while all its direct predecessors are coloured 1. In the first case, the source s being coloured 0 leads to the falsification of the corresponding source axiom $s[\oplus_2]$ by Delayer. In the second case, Delayer will falsify a clause of the corresponding pebbling axiom $\left(\bigwedge_{u \in \mathrm{pred}_G(v)} \overline{u} \vee v\right)[\oplus_2]$.

(2) Next, we show the inequality $\mathsf{PD}(\mathrm{Peb}_G[\oplus_2]) \geq \mathsf{R\text{-}Mc}(G) =: r$ by giving a strategy for Delayer, such that under any strategy of Prover, she scores at least r points. By Definition 21, there is a strategy of Colourer, such that Pebbler has to pebble r vertices to end the game. Delayer will essentially copy this strategy: The first time a variable pair gets queried, she can answer $*$. The second time, she can copy the response of Colourer. Thus, she scores at least r points. $\square$

Note 31. Theorem 30 can easily be generalised to arbitrary k-non-authoritarian functions (the second degree XORification only being a special case of a 1-non-authoritarian function): If f_d is a k-non-authoritarian function of arity d and G is DAG with a unique sink, then $\mathsf{R\text{-}Mc}(G) \leq \mathsf{PD}(\mathrm{Peb}_G[f_d]) \leq (k+1) \cdot \mathsf{R\text{-}Mc}(G)$.

From the equivalence between the Raz–McKenzie game and reversible pebbling we get:

Corollary 32. *It holds* $\mathsf{Rev}(G) + 2 \leq \text{Tree-CS}(\mathrm{Peb}_G[\oplus_2] \vdash \square) \leq 2 \cdot \mathsf{Rev}(G) + 2$ *for all DAGs* G *with a unique sink.*

From this result and Theorem 24 it follows that for any graph G with a gap between its black and reversible pebbling prices, the same separation can be obtained between the general and tree-like clause space of the corresponding pebbling formula. We mention some examples for which such a separation is known:

- The *path graphs*. Consider P_n to be a directed path with n vertices. Bennett [Ben89] noticed, that these graphs provide a separation between black and reversible pebbling, proving that $\mathsf{Rev}(P_n) = \lceil \log n \rceil$. It was shown in [JMNŽ12], using a direct proof, that $\mathrm{CS}(\mathrm{Peb}_{P_n}[\oplus_2] \vdash \square) = \mathrm{O}(1)$ while $\text{Tree-CS}(\mathrm{Peb}_{P_n}[\oplus_2] \vdash \square) = \Theta(\log n)$.
- The *road graphs* from [CLNV15] provide a class of graphs for which the black pebbling price is non-constant and the reversible pebbling number is larger by a logarithmic factor.

Theorem 33 ([CLNV15])**.** *For any function* $s(n) = \mathrm{O}(n^{1/2-\varepsilon})$ *with* $0 < \varepsilon < \frac{1}{2}$ *constant there is a family of DAGs* $(G_n)_{n=1}^{\infty}$ *of size* $\Theta(n)$ *with a single sink and maximal in-degree* 2 *such that* $\mathsf{Black}(G_n) = \mathrm{O}(s(n))$ *and* $\mathsf{Rev}(G_n) = \Omega(s(n) \log n)$.

Corollary 34. *For any function $s(n) = \mathrm{O}(n^{1/2-\varepsilon})$ with $0 < \varepsilon < \frac{1}{2}$ constant there is a family of pebbling formulas $(\mathrm{Peb}_{G_n}[\oplus_2])_{n=1}^{\infty}$ with $\Theta(n)$ variables such that $\mathrm{CS}(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \Box) = \mathrm{O}(s(n))$ and $\text{Tree-CS}(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \Box) = \Omega(s(n) \log n)$.*

The logarithmic factor in the number of vertices is almost the largest separation that can be obtained using this method since it is known that the reversible pebbling price can be upper bounded in terms of black pebbling space and time:

Theorem 35 ([Krá04])**.** *If a DAG G has a black pebbling of time t and space s, the graph G has a reversible pebbling price of at most $s\lceil \log t \rceil$.*

By virtue of this result and Corollary 32 we obtain:

Corollary 36. *For any DAG G with a unique sink vertex it holds*

$$\text{Tree-CS}(\mathrm{Peb}_G[\oplus_2] \vdash \Box) = \mathrm{O}\left(\min_{\mathcal{P}} \left(\mathsf{space}(\mathcal{P}) \cdot \log \mathsf{time}(\mathcal{P})\right)\right),$$

where the minimum is taken over all black pebblings $\mathcal{P}$ of G.

This shows that the given separations cannot be improved for graphs for which the minimum black pebbling space is obtained with a one-shot strategy as it is the case for the path and road graphs, since the pebbling time for such a strategy is n. We present the first graph class for which the best pebbling strategy is not one-shot with a separation between black and reversible pebbling space. We do not obtain, however, any better separation than the $\log n$ factor obtained in the previous examples. We conjecture that this is in fact optimal. Our graphs $\hat{G}(c,k)$ are simplified versions of the original Carlson–Savage graphs [CS82]. Another adaptation of the original graphs is the family $\Gamma(c,r)$ studied in [Nor15], for which an upper bound on the reversible pebble price was recently shown in [dR19]. We have simplified the graphs, eliminating the original pyramids since we are not analysing the black-white pebbling price, but our lower bound on reversible pebbling can be adapted to the original graphs or those in the family $(\Gamma(c,r))_{c,r=1}^{\infty}$.

We have depicted the graphs of the following definition in Figures 1 and 2.

Definition 37 (Simplified Carlson–Savage graphs)**.** The class of DAGs $(G(c,k))_{c,k=1}^{\infty}$ with parameters $c, k \geq 1$ is inductively defined in k. The base case $G(c,1)$ is the graph with one source node connected to c sink nodes. The graph $G(c,k+1)$ is composed of the graph $G(c,k)$ and c spines. A *spine* is just a path of length $2c^2k$. The last node of each of the spines is a sink for $G(c,k+1)$. A spine is divided into $2ck$ *sections* of c consecutive vertices each. For each section and for each i with $1 \leq i \leq c$, there is an edge from the i-th sink of $G(c,k)$ to the i-th vertex in the section. In order to have single sink graphs, for $k \geq 2$ we also define $\hat{G}(c,k)$ exactly as $G(c,k)$ but with just one spine at the k-th level (all other levels have c spines). The last vertex of this spine is the only sink of $\hat{G}(c,k)$. For all c, the graph $\hat{G}(c,1)$ consists of just one edge.

Lemma 38. *The following claims hold:*

(i) $\hat{G}(c,k)$ *has* $\Theta(c^3k^2)$ *vertices,*

(ii) $\mathsf{Black}(\hat{G}(c,k)) \leq k+1$ *for any* $c, k \geq 1$*, while*

(iii) $\mathsf{Rev}(\hat{G}(c,k)) \geq \min\{c, (k-1)\log c + \log(k!)\}$ *for any* $c, k \geq 1$*.*

Proof. The first part follows easily by inductive counting.

For part (ii) of the lemma, we show inductively over k that any sink of $G(c,k)$ can be pebbled using $k+1$ pebbles. The result follows since $\hat{G}(c,k)$ is a subgraph of $G(c,k)$. The claim is trivial

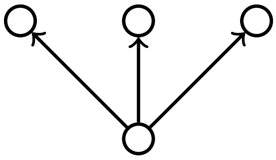

Figure 1: Base case $G(3,1)$ for the simplified Carlson–Savage graph with 3 spines and sinks.

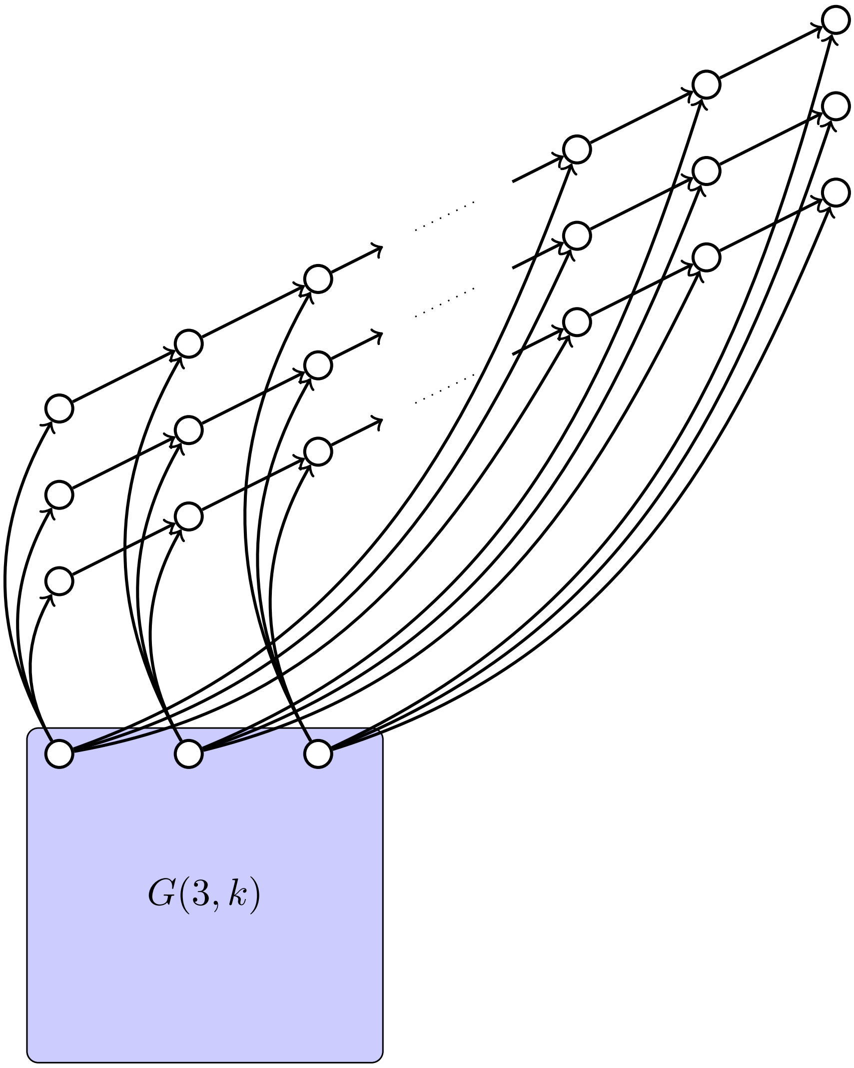

Figure 2: Inductive definition of the simplified Carlson–Savage graph $G(3,k+1)$ with 3 spines and sinks.

for $k = 1$. For bigger values of k, the first vertex in any of the spines in $G(c, k)$ can be pebbled by placing a pebble on the corresponding sink of $G(c, k-1)$, removing all the pebbles except this one, and then pebbling the first vertex in the spine. The following strategy can be used for any other vertex v in the spine once its direct predecessor in the spine is pebbled: remove all the pebbles in the graph except the one on the direct spine predecessor of v, pebble the sink connected to v in $G(c, k-1)$, remove all the pebbles except the 2 on the direct predecessors of v, and then place a pebble on v. For this, by the induction hypothesis, at most $k+1$ pebbles are needed.

Part (iii) is more involved. We use the equivalence between reversible pebbling and the Raz–McKenzie game and show, also by induction over k, that the number of rounds to finish a game on $\hat{G}(c, k)$ starting from a configuration in which less than c vertices have been coloured blue, and no vertex in the unique spine of $\hat{G}(c, k)$ (except the sink) is coloured, is at least $\min\{c, (k-1)\log c + \log(k!)\}$. We give a strategy for Colourer obtaining this bound on the number of rounds. The base case is trivial. For $k \geq 2$, initially the only vertex coloured red is the unique sink of $\hat{G}(c, k)$. Let us denote the unique spine from $\hat{G}(c, k)$ as the *k-spine*. The game is divided in k stages (starting at stage k and finishing at stage 1). Stage k finishes when there is a blue vertex in the k-spine at a distance less than $2c$ from a red vertex. In stage k, if Colourer gives the colour red to a vertex v, this vertex has to be in the k-spine. If some vertex in $G(c, k-1)$ is queried by Pebbler, Colourer always answers with the blue colour. Because of this, the game cannot finish before the end of stage k. For simplicity we may assume that the first vertex of the k-spine has been coloured blue (for free, this can only make the strategy of Colourer harder), also for the clarity of exposition let us say that the k-spine is directed from left to right. The strategy of Colourer on the k-spine is to keep the gap between the rightmost blue vertex a (initially the initial node of the spine) and the leftmost red vertex b (initially the sink) as large as possible. That is, for any queried vertex v in the k-spine, if v lies at the left of a, it is coloured blue, if it is at the right of b it is coloured red and otherwise (i.e., if v is between a and b) if the distance from a to v is smaller that or equal to the distance from v to b, then v is coloured blue, otherwise it is coloured red. This strategy is followed by Colourer as long as the gap between a and b is at least $2c$. Once it is smaller than $2c$, stage k ends. If at this moment at least c vertices have been queried, there have been at least c rounds and the result follows. Otherwise there has to be a spine in $G(c, k-1)$ without any coloured vertex on it (there are c spines). Let us call t the sink of this spine and t' its rightmost uncoloured successor in the k-spine. We can suppose that at this moment Colourer colours (for free) t, t' as well as all uncoloured vertices to the right of t' in the k-spine with colour red, and all the uncoloured vertices to the left of t' in the k-spine with blue. Again this only makes the strategy of Colourer harder since we are not counting these rounds. But now the game has been reduced to the instance of the graph $\hat{G}(c, k-1)$ containing the sink t. The number of rounds in stage k is at least $\log(\frac{2c^2k}{2c}) = \log c + \log k$ (this would happen with a binary search strategy of Pebbler on the k-spine). If in all the stages less than c vertices are queried, by induction, the rounds to finish the game on $\hat{G}(c, k-1)$ are at least $(k-2)\log c + \log((k-1)!)$. Adding these rounds to those from stage k we get the result. ☐

Theorem 39. *For any function $s(n) = \Theta(n^{1/5-\varepsilon})$ with $0 < \varepsilon < \frac{1}{5}$ constant there is a family of pebbling formulas $(\mathrm{Peb}_{G_n}[\oplus_2])_{n=1}^{\infty}$ with $\mathrm{O}(n)$ variables such that $\mathrm{CS}(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \Box) = \mathrm{O}(s(n))$ and $\text{Tree-CS}(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \Box) = \Omega(s(n)\log n)$, and the best strategy for pebbling the graphs G_n is not one-shot.*

Proof. We show that for any such function s there is a graph family $(\hat{G}(c(n), \lceil s(n) \rceil))_{n=1}^{\infty}$ with the corresponding gap between its black and reversible pebbling prices. The result follows from Corollary 32.

Given any such space function $s(n) = \Theta(n^{1/5-\varepsilon})$ with $0 < \varepsilon < \frac{1}{5}$ constant, we define $c(n) := \lceil s(n) \cdot \log n \rceil$. This allows us to consider the graphs $\hat{G}(c(n), \lceil s(n) \rceil)$. By Lemma 38 (i), this graph has $\mathrm{O}\big(c(n)^3 \cdot \lceil s(n) \rceil^2\big) = \mathrm{O}\big(s(n)^5 \cdot \log^3 n\big) = \mathrm{O}\big(n^{1-5\varepsilon} \cdot \log^3 n\big) = \mathrm{O}(n)$ vertices. By Lemma 38 (ii), the graph has a black pebbling number upper bounded by $\lceil s(n) \rceil + 1 = \mathrm{O}(s(n))$. It only remains to show, that the reversible pebbling number of the graph is asymptotically lower bounded by $s(n) \log n$. For this, we consider two cases.

Case 1: $\min \Big\{ c(n), (s(n) - 1) \log c(n) + \log \big(s(n)!\big) \Big\} = c(n)$. In this case, Lemma 38 (iii) implies, that the reversible pebbling number of the graph is lower bounded by $c(n)$, which, by definition, is greater than or equal to $s(n) \log n$.

Case 2: $\min \Big\{ c(n), (s(n) - 1) \log c(n) + \log \big(s(n)!\big) \Big\} = (s(n) - 1) \log c(n) + \log \big(s(n)!\big)$. In this case, one can notice, that already the first term, i. e., $(s(n) - 1) \log c(n)$ is in

$$\begin{aligned}\Omega\Big((s(n) - 1) \log \big(s(n) \log n\big)\Big) &= \Omega\Big((s(n) - 1) \log \big(s(n)\big) + (s(n) - 1) \log \log n\Big) \\ &= \Omega\Big((s(n) - 1)(1/5 - \varepsilon) \log n + (s(n) - 1) \log \log n\Big) = \Omega(s(n) \log n). \quad \square\end{aligned}$$

5 Upper Bounds for Tree-CS for General Formulas

Next, we provide generalisations of Corollary 36 for general formulas.

Theorem 40. *For any unsatisfiable formula F it holds*

$$\text{Tree-CS}(F \vdash \Box) \leq \mathrm{VS}^*(F \vdash \Box) + 2 = \min_{\pi : F \vdash \Box} \big(\mathrm{VS}(\pi) \cdot \log \mathrm{L}(\pi)\big) + 2.$$

Proof. Consider a configurational refutation $\pi = (\mathbb{M}_0, \ldots, \mathbb{M}_t)$ of F. Let α be the current partial assignment constructed in the Prover-Delayer game played on the formula F. At the beginning we have $\alpha = \varnothing$. We give a strategy for Prover that allows him to finish the game with at most $\mathrm{VS}(\pi) \cdot \log \mathrm{L}(\pi)$ points scored by Delayer regardless of her answers. The strategy of Prover proceeds in *bisection steps* indexed with k. Prover keeps as an invariant in these steps an interval $I_k = [a_k, b_k] \subseteq [0, t]$ such that $\pi_{[a_k, b_k]} \restriction_\alpha := (\mathbb{M}_{a_k} \restriction_\alpha, \ldots, \mathbb{M}_{b_k} \restriction_\alpha)$ is a configurational refutation of $F \restriction_\alpha$ for all k. Initially, $I_0 := [0, t]$, thus $\pi_{[0,t]} \restriction_\varnothing = \pi$ is obviously a refutation of $F \restriction_\varnothing = F$. In each bisection step, Prover starts querying the variables present in the configuration $\mathbb{M}_{m_k}$, with $m_k = \lfloor \frac{a_k + b_k}{2} \rfloor$, that have not been assigned yet, in any order. If Delayer answers $*$ to some variable, Prover will assign 0 to it (actually, Prover could assign any value). In this way α is extended to all the variables in the configuration $\mathbb{M}_{m_k}$. Prover then proceeds according to the following cases:

(i) If after the assignment to the queried variables, a clause in the configuration $\mathbb{M}_{m_k}$ is falsified, Prover continues with the upper half of the proof (i. e., he sets $I_{k+1} = [a_{k+1}, b_{k+1}] := [a_k, m_k]$) and proceeds with the next bisection step.

(ii) If after the assignment to the queried variables, all the clauses in $\mathbb{M}_{m_k}$ are satisfied, Prover continues with the lower half of the proof (i. e., he sets $I_{k+1} = [a_{k+1}, b_{k+1}] := [m_k, b_k]$) and proceeds with the next bisection step.

Prover queries at most $\mathrm{VS}(\pi)$ variables in each bisection step. It remains to show that the invariant is indeed kept and that Prover wins the game by following this strategy.

First, we show inductively, that the invariant is kept, i. e., after each step, $I_{k+1} = [a_{k+1}, b_{k+1}] \subseteq [0, t]$ is such that $(\mathbb{M}_{a_{k+1}} \restriction_\alpha, \ldots, \mathbb{M}_{b_{k+1}} \restriction_\alpha)$ is a configurational refutation of $F \restriction_\alpha$. In case (i) this is true by following the Resolution Restriction Lemma (see e.g. [ST13]) because $\mathbb{M}_{b_{k+1}} \restriction_\alpha = \mathbb{M}_{m_k} \restriction_\alpha$ contains the empty clause and thus $(\mathbb{M}_{a_{k+1}} \restriction_\alpha, \ldots, \mathbb{M}_{m_k} \restriction_\alpha)$ is a configurational refutation of $F \restriction_\alpha$.

In case (ii) we have $\mathbb{M}_{m_k}\!\restriction_\alpha = \varnothing$ and $\mathbb{M}_{b_k}\!\restriction_\alpha \ni \Box$ by the induction hypothesis, yet π was a refutation for F. Hence, for $i \in (a_k, b_k)$ the axioms contained in the memory configurations $\mathbb{M}_i\!\restriction_\alpha$ must be downloaded from $F\!\restriction_\alpha$. Thus, $(\mathbb{M}_{a_{k+1}}\!\restriction_\alpha, \ldots, \mathbb{M}_{b_{k+1}}\!\restriction_\alpha)$ is a legal refutation of $F\!\restriction_\alpha$.

Prover has to win the game since for every k, the formula $F\!\restriction_\alpha$ has a configurational refutation, namely $\pi_{I_k}\!\restriction_\alpha$, of length upper bounded by $\frac{1}{2}\mathrm{L}(\pi_{I_{k-1}})$. The strategy proceeds until $F\!\restriction_\alpha$ has a configurational refutation of length 1. Then, $\Box \in F\!\restriction_\alpha$. In other words, the constructed assignment α falsifies a clause in F and Prover wins the game.

Summarising, Prover queries at most $\mathrm{VS}(\pi)$ variables in each bisection step and since there are at most $\lceil \log \mathrm{L}(\pi) \rceil$ configurations that get queried, Prover in total queries at most $\mathrm{VS}(\pi) \cdot \log \mathrm{L}(\pi)$ variables. Theorem 19 yields the desired inequality. □

We prove now that Theorem 40 also works for clause space. For this, we show that the tree-like clause space of a formula F is always upper bounded by the reversible pebble game played on a refutation of F. Note, that the minimum in the theorem is taken over all possible refutations of F, not only over the tree-like ones.

The inequality in Theorem 41 works only in one direction. For example the formula with a clause with n negated variables and n unit clauses containing one of the variables each, has constant tree-resolution space while the reversible pebbling price for any refutation graph is at least $\log n$. It is possible however that the gap between tree-like resolution space and the best reversible pebbling price of any refutation graph from Theorem 41 could be improved for formulas with non-constant tree-like resolution space.

Theorem 41. *For any unsatisfiable formula F with n variables it holds*

$$\begin{aligned} \text{Tree-CS}(F \vdash \Box) &\leq \min_{\pi : F \vdash \Box} \mathsf{Rev}(G_\pi) + 2, \quad \textit{and} \\ \min_{\pi : F \vdash \Box} \mathsf{Rev}(G_\pi) &\leq \text{Tree-CS}(F \vdash \Box)(\lceil \log n \rceil + 1). \end{aligned}$$

Proof. Let F be an unsatisfiable formula with n variables.

For proving the first inequality, let π be a resolution refutation of F with a refutation-graph G_π and $\mathsf{Rev}(G_\pi) =: k$. We will use Theorem 19, as well as Theorem 22 applied to G_π: It suffices to give a strategy for Prover in the Prover-Delayer game played on F under which he has to pay at most k points. Prover basically simulates the strategy of Pebbler in the Raz–McKenzie game played on G_π, which coincides with reversible pebbling. By doing so, a partial assignment α falsifying an initial clause of F will be produced. The game is divided in stages. Initially the partial assignment is the empty assignment. In each stage, if Pebbler chooses a clause $C \in V(G_\pi)$, Prover queries the variables in C not yet assigned by α one by one, extending the partial assignment α with the answers of Delayer, until either:

(i) the clause C is satisfied or falsified by α, or

(ii) a variable x in C is given value $*$ by Delayer.

In case (i), Prover moves to the next stage, simulating the strategy of Pebbler assuming Colourer has given clause C the colour $C\!\restriction_\alpha$. In case (ii), Prover extends α by assigning x with the value that satisfies C and moves to the next stage, simulating the strategy of Pebbler, assuming Colourer has given clause C the colour 1. The game is played until α falsifies a clause in F. After at most k stages the Raz–McKenzie games finishes and therefore Delayer can score at most k points. It is only left to show that at the end of the game a clause in F is falsified by α. When the Raz–McKenzie game finishes, either a source in G_π is assigned colour 0 by Colourer, or a vertex with all its direct predecessors being coloured 1 is coloured 0. Since α defines Colourer's answers, the first situation corresponds to α falsifying a clause in F. The second situation is not possible since for any partial assignment α it cannot be that α satisfies two parent clauses in a resolution proof, while falsifying their resolvent.

For the proof of the second inequality, let $k := \text{Tree-CS}(F \vdash \Box)$. By Proposition 9, we know that there is a refutation π of F whose underlying graph G_π is a tree with black pebbling price k. We can suppose that the refutation is *regular*, that is, in every path from the empty clause to a clause in F in the refutation tree, each variable is resolved at most once [ET01, Theorem 5.1]. This implies that the depth of the tree is at most n. For any node v in the refutation tree let T_v be the subtree of G_π rooted at v. For the sake of convenience, we refer to $\mathsf{Black}(T_v)$ as the *pebbling number of* v.

We show by induction on κ that for any vertex v in G_π, if $\mathsf{Black}(T_v) = \kappa$ then there is a strategy for Pebbler in the Raz–McKenzie game on T_v with most $\kappa(\lceil \log n \rceil + 1)$ rounds. For the base case $\kappa = 1$, the vertex v must be a leaf node and the game needs only one round. For $\kappa > 1$, the game starts, according to the rules, by Pebbler querying the root v of the subtree and Colourer answering 0. We consider two cases, depending on whether for both predecessors v_1 and v_2 of v in G_π, $\mathsf{Black}(T_{v_1}) = \mathsf{Black}(T_{v_2}) = \kappa - 1$ or not. In the former case, Pebbler queries one of them, say v_1. If the answer is 0, he continues on T_{v_1} and otherwise continues on T_{v_2}. By induction, the number of rounds in this case is at most $2 + (\kappa - 1)(\lceil \log n \rceil + 1) \le \kappa(\lceil \log n \rceil + 1)$. In case, it is not true, that $\mathsf{Black}(T_{v_1}) = \mathsf{Black}(T_{v_2}) = \kappa - 1$, since $\mathsf{Black}(T_v) = \kappa$, and G_π is a tree, one of the trees T_{v_1} or T_{v_2} leading to v must have pebbling number κ and the other one must have pebbling number smaller than κ. Pebbler considers the path of nodes starting at v and going towards the leaves, having all the nodes in the path pebbling number κ, until a node u is reached, for which both predecessors have pebbling number $\kappa - 1$. Such a node u must exist because G_π is a tree. Let u_1 be one of the predecessors of u. The length of the path from v to u_1 is at most n since the refutation is regular. Pebbler queries the vertices in the path between v and u_1 with binary search, until a vertex t is found that is coloured with colour 0 by Colourer, while its predecessor in the path $v \rightsquigarrow u_1$ has been coloured 1. At this point, Pebbler continues playing the game on the tree rooted at the uncoloured predecessor of t. It is also possible that all the queried nodes in the path from v to u_1 (including u_1) are coloured 0 by Colourer. In this case Pebbler continues with T_{u_1}. In all situations at most $1 + \lceil \log n \rceil$ vertices have been queried and the game has been reduced to a subgraph with smaller pebbling number. $\Box$

Corollary 42. *For any unsatisfiable formula F it holds*

$$\text{Tree-CS}(F \vdash \Box) \le \text{CS}^*(F \vdash \Box) + 2 = \min_{\pi : F \vdash \Box} \left(\text{CS}(\pi) \cdot \log \text{L}(\pi)\right) + 2.$$

Proof. By Theorem 35, $\min_{\pi : F \vdash \Box} \mathsf{Rev}(G_\pi) + 2 \le \min_{\mathcal{P}} \left(\mathsf{space}(\mathcal{P}) \cdot \log \mathsf{time}(\mathcal{P})\right) + 2$, where the minimum is taken over all black pebblings $\mathcal{P}$ of G_π. The result follows with (a slight adaption of) Proposition 9 since every black pebbling $\mathcal{P}$ of G_π defines a configurational refutation of F with clause space equal to $\mathsf{space}(\mathcal{P})$ and length $\mathsf{time}(\mathcal{P})$. $\Box$

6 Optimal Separations for Tseitin Formulas

In this section, we prove optimal separations between tree-like clause space and variable space, as well as clause space in the context of Tseitin formulas. This complements the relations between clause space and variable space of Tseitin formulas recently given in [GTT18].

Theorem 43. *For any connected graph G with n vertices and odd marking χ we have*

$$\text{Tree-CS}(\text{Ts}(G,\chi) \vdash \Box) \le \text{CS}(\text{Ts}(G,\chi) \vdash \Box) \cdot \log n + 2, \quad \textit{and}$$
$$\text{Tree-CS}(\text{Ts}(G,\chi) \vdash \Box) \le \text{VS}(\text{Ts}(G,\chi) \vdash \Box) \cdot \log n + 2.$$

Proof. The proof is based on the one for the lower bound for CS of Tseitin formulas from [Tor99]. Let $G = (V, E)$ be a connected graph with n vertices, χ an odd marking, and $\pi = (\mathbb{M}_0, \ldots, \mathbb{M}_t)$ a refutation of $\mathrm{Ts}(G, \chi)$ with $\mathrm{CS}(\pi) =: k$. We use π to give a strategy for Prover in the Prover-Delayer game for which he has to pay at most $k \log n$ points.

We say that a partial assignment α of some of the variables in $\mathrm{Ts}(G, \chi)$ is *non-splitting* if after applying α to the formula, the resulting graph still has a connected component with an odd marking (*odd component*) of size at least $\lceil \frac{|V|}{2} \rceil$, and the rest are components with even markings. Consider the last configuration $\mathbb{M}_s$ in π for which there is a partial assignment α fulfilling:

(i) α simultaneously satisfies all clauses in $\mathbb{M}_s$, and

(ii) α is non-splitting.

This stage must exist since before the initial step the empty truth assignment is trivially a non-splitting partial assignment satisfying the clauses in $\mathbb{M}_0 = \varnothing$. At the end, the last configuration $\mathbb{M}_t$ in the refutation contains the empty clause which cannot be satisfied by any assignment. Thus, stage s must exist in between.

The step from s to $s+1$ was no deletion step (otherwise this would be a contradiction to the maximality of s). The only new clause in $\mathbb{M}_{s+1}$ must be an axiom C of $\mathrm{Ts}(G, \chi)$ since any other clause that could be added to the list of clauses in memory at stage $s+1$ would be a resolvent of two clauses from stage s, but in this case any partial assignment satisfying the clauses at stage s would also satisfy those at $s+1$. For some vertex v in G, this axiom clause C introduced at stage $s+1$ belongs to the formula $\mathrm{PARITY}_{v,\chi(v)}$. Let α be a partial assignment of minimal size satisfying the conditions at stage s. It is possible to extend α to satisfy the clause $C\restriction_\alpha$ since v either belongs to an even component in $(G\restriction_\alpha, \chi\restriction_\alpha)$ or to the large odd component in this graph and therefore $C\restriction_\alpha \neq \square$. Because of this, vertex v must belong to the unique odd component since otherwise α could be extended in a non-splitting way.

Let $C\restriction_\alpha = (\ell_1, \ldots, \ell_m)$, $m \geq 1$, where the ℓ_i's are literals corresponding to the edges with endpoint v in $G\restriction_\alpha$. Observe that deleting any of these edges e_i in $C\restriction_\alpha$ cuts the connected component of v in two pieces because otherwise assigning any value to the corresponding edge would not modify the size of the connected components in $C\restriction_\alpha$ and there would be a non-splitting way to extend α to e satisfying C. Also, any component remaining after assigning all the literals in $C\restriction_\alpha$ must have size at most $\lfloor \frac{|V|}{2} \rfloor$ since otherwise there would be a way to extend α satisfying C and producing an odd marking for the largest such component (Fact 16), and this extension would be non-splitting.

The strategy of Prover is to query the variables assigned in α thus paying at most $k-1$ points and obtaining a partial assignment γ from Delayer. If at this point one of the connected components of size at most $\lfloor \frac{|V|}{2} \rfloor$ is odd then Prover moves to this component and starts playing the game on it. Otherwise Prover queries the variables in $C\restriction_\gamma$ one by one. If for a variable e_i Delayer answers with $*$, Prover just has to assign e_i so that the smallest of the two components that appear in $G\restriction_\gamma$ after assigning e_i is odd (not necessarily satisfying $C\restriction_\gamma$). This is always possible because of Fact 16. If no $*$ is answered, Prover queries the next variable until no variable in $C\restriction_\gamma$ is left. Let γ' be the assignment obtained this way. Either γ' falsifies C and the game ends, or it satisfies the clause and in this case all the components (odd or even) remaining after applying γ' have size at most $\lfloor \frac{|V|}{2} \rfloor$. In every case, after applying γ' Prover wins the game or there is an odd connected component of size at most half as large as the initial graph. The original problem has been reduced to another in a graph with at most $\frac{n}{2}$ many vertices. Also Prover has to pay at most k for obtaining γ'. After repeating this process at most $\log n$ times, an initial clause is falsified.

The second part of the theorem is a little simpler and follows by considering a configurational proof π of variable space k. Everything in the above proof works in the same way, observing

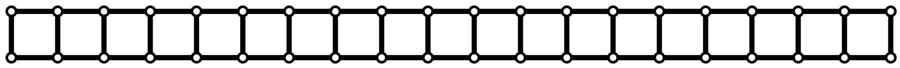

Figure 3: The grid graph $G_{2\times 20}$.

that the partial assignment α satisfying all clauses in memory at stage s, when extended to all the variables in the new clause at stage $s+1$ needs to assign at most k variables (all those included in the configuration) and is either splitting or falsifies the axiom. Observe that this implies that in every configurational proof there is a point in which every assignment to the variables in the configuration is spliting. $\square$

Next, we show, that the upper bounds in Theorem 43 are tight by proving that there is a family of Tseitin formulas that provide matching lower bounds. These are formulas corresponding to grid graphs with constant width (see Figure 3), which can be considered as the Tseitin version of path graphs.

Definition 44 (Grid graphs)**.** For a natural number $\ell \geq 1$, the *grid graphs* $G_{2\times\ell}$ as depicted in Figure 3 are given by the vertex set $V(G_{2\times\ell}) := [2] \times [\ell]$ and the edge set

$$E(G_{2\times\ell}) := \Big\{\{(i,j),(i',j')\} \,:\, i,i' \in [2],\ j,j' \in [\ell],\ \text{and } |i-i'|+|j-j'| = 1\Big\}.$$

Theorem 45. *For the family of Tseitin formulas* $(\mathrm{Ts}(G_{2\times\ell},\chi_\ell))_{\ell=1}^{\infty}$ *with* $3\ell-2$ *variables it holds* $\text{Tree-CS}(\mathrm{Ts}(G_\ell,\chi_\ell)\vdash\Box) = \Theta(\log\ell)$, $\mathrm{CS}(\mathrm{Ts}(G_\ell,\chi_\ell)\vdash\Box) = \mathrm{O}(1)$, *and* $\mathrm{VS}(\mathrm{Ts}(G_\ell,\chi_\ell)\vdash\Box) = \mathrm{O}(1)$.

Proof. To show the lower bound on tree-like clause space with Theorem 19, we give a strategy for Delayer such that he scores $\Omega(\log\ell)$ points playing on $G_{2\times\ell}$. In the following, for a subgraph G' of $G_{2\times\ell}$, we define

$$\mathrm{Block}(G') := \max\{b \in \mathbb{N} :\ \text{there is a subgraph of } G' \text{ that is isomorphic to } G_{2\times b}\}.$$

The strategy of Delayer is as follows:

(a) If an edge e in an even component is queried, Delayer should answer according to some assignment satisfying this component.

(b) If an edge e in an odd component is queried, Delayer proceed as follows:

   (i) If the deletion of e does not increase the number of connected components in G, Delayer should answer $*$.

   (ii) If the deletion of e cuts the graph and both endpoints of e are separated in different connected components, Delayer should answer in a way, that from these two components, the component G' with largest $\mathrm{Block}(G')$ receives the odd marking.

At the beginning of the Prover-Delayer game we have $\mathrm{Block}(G_{2\times\ell}) = \ell$. After each assignment of a variable in the game we have $\mathrm{Block}(G') \geq \lfloor\frac{1}{2}\,\mathrm{Block}(G)\rfloor$, where we let G denote the underlying graph before the assignment and G' the graph after the assignment: Notice, that rule (b)(ii) guarantees that the component with the largest Block-value always receives an odd marking. If Delayer plays according to this strategy, we must have $\mathrm{Block}(G) = 0$ at the beginning of some round. This means that the Block-value, starting the game with $G_{2\times\ell}$, has to change at least $\Omega(\log\ell)$ times before the game can end. It is easy to see, that if the Block-value changes in a step, the number of connected components does not increase in this step. According to rule (b) (i), Delayer has answered $*$ in this round and has scored a point.

For the second part, consider the variables (edges) ordered (from left to right) with $\{(1,j),(2,j)\} \prec \{(1,j),(1,j+1)\} \prec \{(2,j),(2,j+1)\}$ and edges with lower j defined to be smaller (with respect to $\prec$) than those with higher j for $1 \leq j \leq \ell-1$, and consider a resolution refutation completely resolving the variables in decreasing order (from right to left). That is, the clauses containing variable $\{(1,\ell),(2,\ell)\}$ will be first resolved with all clauses containing this variable in negated form (in case it is possible to resolve), and so on. Since the graph has degree at most 3, there is a small number of clauses containing this variable. Also observe that after resolving the last three variables in the ordering in this way, the set of derived clauses plus the initial clauses contain a subset of clauses encoding the formula $\mathrm{Ts}(G_{2\times(\ell-1)},\chi')$ for some odd marking χ'. The set of newly derived clauses in this subset has constant size, and the number of clauses in all the resolution configurations until this point is also constant. Continuing in this order with the complete resolution of all the variables, we obtain a refutation of $\mathrm{Ts}(G_{2\times\ell},\chi)$ with constant clause and variable space. $\square$

7 Conclusions and Open Problems

By introducing a new connection between tree-like resolution space and the reversible pebble game, we have studied the relation between tree-like space and space measures for general resolution, obtaining almost optimal separations between these measures. We conjecture that these separations are optimal and that in fact, the $\log(\mathrm{time}(\pi))$ factors in the upper bounds of Theorems 35 and 40 and Corollaries 36 and 42 can be improved to a $\log n$ factor (n being the number of graph vertices or formula size, depending on the setting). We have been able to prove this for the restricted case of the Tseitin contradictions.

We have seen that a source for obtaining space separations between tree-like and general resolution are graph classes with a gap between their reversible and black pebbling prices and we have provided a new class of such graphs. An interesting question is whether there exists a graph class with such a separation for a space function larger than $n^{1/2}$.

References

[ABLM08] Carlos Ansótegui, María Luisa Bonet, Jordi Levy, and Felip Manyà. Measuring the hardness of SAT instances. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI '08)*, pages 222–228, July 2008. Cited on page(s) 1.

[ABRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002. Preliminary version in *STOC '00*. Cited on page(s) 6.

[BEGJ98] María Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. Exponential separations between restricted resolution and cutting planes proof systems. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS '98)*, pages 638–647, November 1998. Cited on page(s) 2.

[Ben89] Charles H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, August 1989. Cited on page(s) 5, 11.

[BIW04] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, September 2004. Cited on page(s) 2, 9.

[BN08] Eli Ben-Sasson and Jakob Nordström. Short proofs may be spacious: An optimal separation of space and length in resolution. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '08)*, pages 709–718, October 2008. Cited on page(s) 8.

[BN11] Eli Ben-Sasson and Jakob Nordström. Understanding space in proof complexity: Separations and trade-offs via substitutions. In *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS '11)*, pages 401–416, January 2011. Full-length version available at http://eccc.hpi-web.de/report/2010/125/. Cited on page(s) 7, 8.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC '99*. Cited on page(s) 2, 6, 7.

[Cha13] Siu Man Chan. Just a pebble game. In *Proceedings of the 28th Annual IEEE Conference on Computational Complexity (CCC '13)*, pages 133–143, June 2013. Cited on page(s) 3, 9.

[CLNV15] Siu Man Chan, Massimo Lauria, Jakob Nordström, and Marc Vinyals. Hardness of approximation in PSPACE and separation results for pebble games (Extended abstract). In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS '15)*, pages 466–485, October 2015. Full-length version in [Vin17]. Cited on page(s) 3, 11.

[CS76] Stephen A. Cook and Ravi Sethi. Storage requirements for deterministic polynomial time recognizable languages. *Journal of Computer and System Sciences*, 13(1):25–37, 1976. Preliminary version in *STOC '74*. Cited on page(s) 4.

[CS82] David A. Carlson and John E. Savage. Extreme time-space tradeoffs for graphs with small space requirements. *Information Processing Letters*, 14(5):223–227, 1982. Cited on page(s) 12.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962. Cited on page(s) 1.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960. Cited on page(s) 1.

[dR19] Susanna F. de Rezende. *Lower Bounds and Trade-offs in Proof Complexity*. PhD thesis, KTH Royal Institute of Technology, June 2019. Available at http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A1318061&dswid=-4968. Cited on page(s) 12.

[ET01] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001. Preliminary versions of these results appeared in *STACS '99* and *CSL '99*. Cited on page(s) 2, 3, 6, 7, 8, 17.

[ET03] Juan Luis Esteban and Jacobo Torán. A combinatorial characterization of treelike resolution space. *Information Processing Letters*, 87(6):295–300, 2003. Cited on page(s) 2, 8, 9.

[GTT18] Nicola Galesi, Navid Talebanfard, and Jacobo Torán. Cops-robber games and the resolution of Tseitin formulas. In *Proceedings of the 21th International Conference on Theory and Applications of Satisfiability Testing (SAT '18)*, volume 10929 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 2018. Cited on page(s) 17.

[JMNŽ12] Matti Järvisalo, Arie Matsliah, Jakob Nordström, and Stanislav Živný. Relating proof complexity measures and practical hardness of SAT. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP '12)*, volume 7514 of *Lecture Notes in Computer Science*, pages 316–331. Springer, October 2012. Cited on page(s) 1, 2, 3, 11.

[Krá04] Richard Královič. Time and space complexity of reversible pebbling. *RAIRO – Theoretical Informatics and Applications*, 38(02):137–161, April 2004. Cited on page(s) 12.

[KS91] Balasubramanian Kalyanasundaram and George Schnitger. On the power of white pebbles. *Combinatorica*, 11(2):157–171, June 1991. Preliminary version in *STOC '88*. Cited on page(s) 10.

[Mey81] Friedhelm Meyer auf der Heide. A comparison of two variations of a pebble game on graphs. *Theoretical Computer Science*, 13(3):315–322, 1981. Cited on page(s) 3, 10.

[Nor12] Jakob Nordström. On the relative strength of pebbling and resolution. *ACM Transactions on Computational Logic*, 13(2):16:1–16:43, April 2012. Preliminary version in *CCC '10*. Cited on page(s) 2, 10.

[Nor15] Jakob Nordström. New wine into old wineskins: A survey of some pebbling classics with supplemental results. Manuscript in preparation. Current draft version available at `http://www.csc.kth.se/~jakobn/research/PebblingSurveyTMP.pdf`, 2015. Cited on page(s) 2, 3, 4, 5, 6, 7, 12.

[PH70] Michael S. Paterson and Carl E. Hewitt. Comparative schematology. In *Record of the Project MAC Conference on Concurrent Systems and Parallel Computation*, pages 119–127, 1970. Cited on page(s) 4.

[PI00] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-SAT (preliminary version). In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '00)*, pages 128–136, January 2000. Cited on page(s) 8, 9.

[Raz18] Alexander A. Razborov. On space and depth in resolution. *Computational Complexity*, 27(3):511–559, 2018. Cited on page(s) 3, 4, 7.

[RM99] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, March 1999. Preliminary version in *FOCS '97*. Cited on page(s) 3, 8.

[ST13] Uwe Schöning and Jacobo Torán. *The Satisfiability Problem: Algorithms and Analyses*, volume 3 of *Mathematics for Applications (Mathematik für Anwendungen)*. Lehmanns Media, 2013. Cited on page(s) 6, 15.

[Tor99] Jacobo Torán. Lower bounds for space in resolution. In *Proceedings of the 13th International Workshop on Computer Science Logic (CSL '99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 362–373. Springer, 1999. Cited on page(s) 18.

[Tse68] Grigori Tseitin. The complexity of a deduction in the propositional predicate calculus. *Zapiski Nauchnyh Seminarov Leningradskogo Otdelenija matematicheskogo Instituta im. V. A. Steklova akademii Nauk SSSR (LOMI)*, 8:234–259, 1968. In Russian. Cited on page(s) 8.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, January 1987. Cited on page(s) 8.

[Vin17] Marc Vinyals. *Space in Proof Complexity*. PhD thesis, KTH Royal Institute of Technology, May 2017. Available at `http://www.csc.kth.se/~jakobn/project-proofcplx/docs/MV_PhDthesis.pdf`. Cited on page(s) 3, 21.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il