Resolution and Pebbling Games

Nicola Galesi[*] Neil Thapen [†]

Abstract

We define a collection of Prover-Delayer games that characterize certain subsystems of resolution. This allows us to give some natural criteria which guarantee lower bounds on the resolution width of a formula, and to extend these results to formulas of unbounded initial width.

We also use games to give upper bounds on proof size, and in particular describe a good strategy for the Prover in a certain game which yields a short refutation of the Linear Ordering principle.

Using previous ideas we devise a new algorithm to automatically generate resolution refutations. On bounded width formulas, our algorithm is as least as good as the width based algorithm of [9]. Moreover, it finds short proofs of the Linear Ordering principle when the variables respect a given order.

Finally we approach the question of proving that a formula F is hard to refute if and only if is "almost" satisfiable. We prove both directions when "almost satisfiable" means that it is hard to distuinguish F from a satisfiable formula using limited pebbling games.

1 Introduction

Propositional resolution is one of the most intensively studied logical systems. Its importance stems both from applied and from theoretical points of view. On one hand it provides the logical basis for almost all of the more important and efficiently implemented automatic theorem provers (see [5]). On the other hand, it has probably been the most studied proof system in the area of proof complexity ([14, 6, 4, 9, 20] among others).

[*]Universitat Politècnica de Catalunya. Dept. LSI. Barcelona, Spain. e-mail: **galesi@lsi.upc.es**. Supported by grant TIC2001-1577-C03-02. Part of this work was done while the author was visiting Oxford University, partly supported by a grant from the London Mathematical Society.

[†]St Hilda's College, University of Oxford, Oxford, UK. email: **neil-thapen@st-hildas.ox.ac.uk**

ISSN 1433-8092

Most of the work to understand the strength of resolution has been concentrated on proving lower bounds for the length of refutations. Recently Ben-Sasson and Wigderson [9] based on ideas of [6] gave a unified approach to obtaining lower bounds. They showed that if a bounded width formula has a short refutation, then it has a narrow refutation. Using this relationship they give an algorithm to generate resolution refutations based on the width measure.

Simple pebbling games were initially introduced into the world of resolution in [18] to study size lower bounds in the subsystem of resolution where the proofs are treelike. Later the study of the space measure for resolution (see [12, 1, 7]) suggested the use of a more complex pebbling game.

It was soon clear that games also play an important role also in the study of the width limit for the full resolution system. Atserias and Dalmau [3] have given a finite model-theoretic characterization of the bounded width formulas with narrow refutations, using a pebbling game.

In this paper we carry this idea further. In section 2 we define a new restriction of the resolution system (narrow resolution) for which we give a unified way of proving lower bounds similar to those for bounded width resolution of [9], but without having the restriction that the original formula has bounded width.

We define a "witnessing" pebble game, played between a Prover and a Delayer, and show that proofs in this system correspond to strategies for the Prover. On the other hand, a good strategy for the Delayer corresponds to the formula being extended dynamically satisfiable (EDS), an extension of an idea used in [11, 13] to prove lower bounds on space and on treelike size.

In section 3 we give some sufficient conditions for a formula to be k-EDS. In particular we show that if there is a satisfiable formula G which looks similar to F in that G cannot be distinguished from F with $k + 1$ or fewer pebbles, then F is k-EDS.

We also adapt a criterion of Riis' [19] to show that if F is a translation of a combinatorial principle on some finite structure, and this principle is satisfiable on a larger structure, then F is $\Omega(n)$-EDS. This gives us a useful sufficient condition for proving width lower bounds.

Starting from this witnessing pebble game we explore in two directions. In Section 4 we study some more general "structured" games, and generalize the width concept. In the earlier game the restriction on width corresponded to the Prover only being able to remember the values of a limited number of variables. In the structured game, the number of pebbles still limits how much the Prover can remember, so the games have useful properties in common with bounded width proofs; but each pebble can be labelled with information about several variables, which means the Prover can refute things that would be impossible if he was limited by width.

We show how to recover Resolution refutations from Prover strategies (theorem 19). In particular, since we show that the Prover has a winning strategy for a certain "ordered" structure game with only three pebbles over the Linear Ordering Principle LOP (theorem 26), we can recover a polynomial size refutation of this principle.

The LOP principle was used in [10] to prove the optimality of the width-size tradeoff for Resolution. It is one of the very few examples of CNF formulas having polynomial size resolution refutations but on which the Ben-Sasson Wigderson algorithm takes subexponential time to recover a refutation. Moreover it is also hard for DPLL-based theorem provers. Motivated by the previous result we then look at the the question of automatically generating resolution refutations using strategies for the game.

We devise a new algorithm based on reconstructing strategies for the Prover which allows us to extend the Ben-Sasson and Wigderson algorithm to an algorithm which, with some extra information about ordering of the variables, generates short refutations of LOP in polynomial time. Moreover we show that for bounded width formula our algorithm is at least as good as the Ben-Sasson and Wigderson algorithm.

Finally in section 5 we explore further the idea used in section 3, that if a CNF F is similar (in some finite model theoretic sense) to a satisfiable formula G, then CNF can be thought of as "almost satisfiable" and is hard to refute. While this last result can be seen as a soundness theorem, in this section we also give a kind of "completeness" theorem, and show that if F is hard to refute using a certain game, then F is similar to a satisfiable CNF G.

2 The Witnessing Game

Definition 1 *Fix $k \in \mathbb{N}$. Call a clause narrow if it has width k or less; otherwise it is wide. A width k* narrow resolution refutation *of a CNF F is a sequence of narrow clauses, beginning with the narrow clauses of F and finishing with the empty clause. There are three ways that clauses can be introduced into the sequence:*

1. *From B we can derive Bx (weakening);*
2. *From Bx and $C\bar{x}$ we can derive BC (resolution);*

3. *If $x_1 \dots x_m$ is a (usually wide) clause in F, then from $B\bar{x}_1, \dots, B\bar{x}_m$ we can derive B (resolution by cases).*

Proposition 2 *If F is a r-CNF with a width k narrow resolution refutation, then F has a width $r + k - 2$ "normal" resolution refutation.*

Proof Replace each resolution-by-cases inference with a sequence of resolution steps . □

We introduce a pebble game to accompany this proof system.

Definition 3 *Let F be a CNF. The witnessing pebble game on F is played between a Prover and a Delayer on the set of literals arising from the variables in F. A pebble can never appear on both a literal and its negation. In each turn, one of three things can happen.*

1. *The Prover lifts a pebble from the board; the Delayer makes no response.*
2. *(Querying a variable.) The Prover gives a pebble to the Delayer and names an empty variable x (that is, neither x nor $\bar{x}$ can be pebbled already). The Delayer must put the pebble on either x or $\bar{x}$.*
3. *(Querying a clause.) The Prover gives a pebble to the Delayer and names a clause C from F. The Delayer must place the pebble on one of the literals in C, without contradicting any pebble already on the board. If this is not possible then the Prover wins.*

Normally we will limit the game to some number k of pebbles, and call this the k-pebble witnessing game.

Notice that the Prover can win exactly when the pebbles on the board falsify some clause of F and the Prover has one pebble left over.

Proposition 4 *Let F be a CNF and $k \in \mathbb{N}$.*

1. *If there is a winning strategy for the Prover in the k-pebble witnessing game for F, then there is a narrow resolution refutation of F of width k.*
2. *If there is a narrow resolution refutation of F of width k then there is a winning strategy for the Prover in the $(k+1)$-pebble witnessing game for F.*

3. *If F has a (normal) resolution proof of width k, then there is a winning strategy for the Prover in the $(k+1)$-pebble witnessing game for F.*

4. *If F has a (normal) resolution proof of clause space k, then there is a winning strategy for the Prover in the k-pebble witnessing game for F.*

Proof

1. Consider the Prover's strategy as a tree, with each node labelled with the set of literals falsified under the assignment given by the pebbles currently in play. Then the root will be the empty clause, and the leaves will be (some subset of) the narrow clauses of F. If we read this tree from the leaves down to the root, we get precisely a narrow resolution refutation of F. Removing a pebble corresponds to weakening, the Prover querying a variable corresponds to a resolution step, and the Prover querying a clause corresponds to a resolution-by-cases step.

2. Use the narrow proof as a strategy for the Prover, as above. If $Bx, C\bar{x} \vdash BC$ is a resolution step in the proof, and BC has width k, the corresponding step in the Prover's strategy will require $k+1$ pebbles: the Prover will start with pebbles falsifying BC, and will need one extra pebble to query x (and depending on the answer will the use weakenings to get either Bx of $C\bar{x}$).

3. A special case of 2 above.

4. Define a configuration to be a set of k or fewer clauses, and a resolution proof of clause space k to be sequence $G_0 \ldots G_m$ of configurations such that $G_0 = \emptyset$, $G_m = \{\emptyset\}$ and each G_{i+1} is derived from G_i by axiom download, erasing a clause, or adding a clause by a resolution inference.

   For each i, the Prover can force an assignment of pebbles such that all of the clauses in G_i are satisfied. For each axiom download step, he queries the clause being downloaded, so that the Delayer must place a pebble to satisfy that clause. No more than k pebbles are needed to satisfy k clauses, so the Prover can remove a pebble each time a clause is erased. And the Prover does not need to do anything for an inference step.

   Hence either the pebbles will eventually satisfy the empty clause, which is impossible, or at some point the Delayer must be unable to satisfy a clause which the Prover has queried. So the Prover wins. □

We can now adapt the Ben-Sasson Wigderson result that "short proofs are narrow" to talk about games rather than proofs. This allows us to apply it directly to CNFs of unbounded width.

Definition 5 *If F is a CNF and x is a literal, we obtain $F|x$ from F by removing all clauses containing x and removing $\bar{x}$ from any clause in which it appears.*

So from a resolution refutation of F we can obtain a refutation of $F|x$ of equal size or smaller, by substituting in a value of "true" for x and simplifying.

Lemma 6 *If the Prover has a winning strategy for the k-pebble witnessing game on $F|x$, then in the k-pebble witnessing game on F the Prover can force the Delayer to either lose the game or place a pebble on $\bar{x}$.*

Proof Let S be the k-pebble winning strategy for $F|x$. We will make this into a strategy for the game on F as follows. Whenever a clause C is queried in S such that $C \in F|x$ but $C \notin F$, it must be that $C\bar{x} \in F$. So replace this query with a query of $C\bar{x}$. Then either the Delayer must eventually place a pebble on $\bar{x}$, or the play of pebbles must be exactly the same as given in strategy S so that the Delayer must eventually lose. $\square$

Theorem 7 ([9]) *Fix $d, n \in \mathbb{N}$ and let $\beta = (1 - \frac{d}{2n})^{-1}$. Say that a clause is fat if it has width greater than d. Then for any $m \leq n$ and any b, if F is a CNF on m variables and has a (normal) resolution refutation Π containing $< \beta^b$ many fat clauses, then the Prover has a winning strategy in the witnessing pebble game on F, using $d + b + 1$ pebbles.*

Proof

The proof is by induction on m. The base case $m = 0$ is trivial, so suppose $m > 0$.

If $b = 0$, then every clause in Π (and also every clause in F) has width $\leq d$, so by an earlier observation there is a strategy for the Prover using $d + 1$ pebbles.

Otherwise, let Π^* be the set of fat clauses appearing in Π. Then there must be some literal x appearing in at least $\frac{d}{2n}|\Pi^*|$ fat clauses, since otherwise

$$|\Pi^*|\, d \leq |\{(y, C) : \ y \text{ is a literal in } C \in \Pi^* \ \}| < 2m\frac{d}{2n}|\Pi^*|.$$

The first part of the Prover's strategy is to force the Delayer to put a pebble on x. Now $F|\bar{x}$ contains only $m - 1$ variables and has a refutation with fewer than β^b fat clauses, so by the inductive hypothesis the Prover has a strategy for the game on $F|\bar{x}$ using $b+d+1$ pebbles. Hence by the lemma the Prover can force the Delayer to satisfy x.

Setting x to true will make all the clauses in Π containing x vanish, so $F|x$ contains only $m-1$ variables and has a refutation with fewer than $(1-\frac{d}{2n})|\Pi^*| \leq \beta^{b-1}$ fat clauses. Hence the Prover has a winning strategy T for $F|x$ with only $b+d$ pebbles.

The Prover now leaves one pebble on x and uses the remaining $b+d$ pebbles to carry out strategy T on the remaining variables. As in the lemma, he must change T slightly to make it into a strategy for the game on F, by replacing queries to $C \in F|x \setminus F$ with queries to $C\bar{x}$. But the Delayer can never put a pebble on $\bar{x}$, because there is already a pebble on x. So the game plays just like the $F|x$ game with strategy T, and the Prover wins. $\square$

3 Extended dynamic satisfiability

Along similar lines to those used in [3] for resolution width and in [11] for resolution space, we characterize the Delayer's strategy in the witnessing game, in terms of families of partial assignments for the formula.

We also generalize this result by studying sufficient conditions that imply good strategies for the Delayer, along the lines of the approaches in [19] and [16] using first order model theory.

The following definition was introduced in [11].

Definition 8 *A CNF F is k-dynamically satisfiable (k-DS) if there is a class R of partial assignments to the variables of F with the following properties:*

1. *R is closed under subset;*
2. *If $\alpha \in R$, $|\alpha| < k$ and C is any clause of F, then there is an extension $\beta \in R$ of α that satisfies C (in the sense that it makes at least one of the literals in C true, but does not necessarily assign a value to all the literals).*

To characterize good strategies for the delayer in our witnessing game, we alter the definition of dynamic satisfiability by adding a case to deal with queries made to variables:

Definition 9 *A CNF F is k extended-dynamically satisfiable (k-EDS) if there is a class R of partial assignments satisfying the conditions of definition 8, with the extra case:*

3. *If $\alpha \in R$, $|\alpha| < k$ and x is any variable appearing in F, then there is an extension $\beta \in R$ of α that assigns some value to the variable x.*

Lemma 10 *A CNF F is k-extended dynamically satisfiable if and only if the Delayer has a winning strategy for the k-pebble witnessing game on F.*

Proof Suppose F is k-EDS. Then the Delayer can guarantee that after every turn the assignment α given by the k pebbles is in R. Hence by part 3 of the definition of extended dynamic satisfiability, the Delayer is always able to consistently satisfy any clause of F that the Prover queries.

Conversely, suppose that the Delayer has a winning strategy. Let R be the set of all assignments corresponding to the configurations of pebbles that can appear in a game in which the Delayer uses this strategy. Then R witnesses that F is k-EDS. $\square$

Theorem 11 *(A corollary to Ben-Sasson Wigderson) For any $\varepsilon > 0$, if a CNF F has n variables and is $(\sqrt{8}n^{\frac{1+\varepsilon}{2}} + 1)$-EDS, then it has no resolution refutation of size 2^{n^ε}.*

Proof Let $b = d = \sqrt{2}n^{\frac{1+\varepsilon}{2}}$. For large n, $\beta^{\frac{2n}{d}} = (1 - \frac{d}{2n})^{-\frac{2n}{d}} > 2$. So

$$2^{n^\varepsilon} < \beta^{\frac{2n}{d}n^\varepsilon} = \beta^{\frac{2n^{1+\varepsilon}}{\sqrt{2n^{1+\varepsilon}}}} = \beta^b$$

and any proof of size 2^{n^ε} must have fewer than β^b fat clauses. Hence there is a winning strategy for the Prover with $b + d + 1$ pebbles, contradicting dynamic satisfiability. $\square$

The careful reader should have noticed that our extended dynamical satisfiability is a simplification of the characterization of Duplicator strategies in the extended existential game of Asterias and Dalmau (see defintion 3 of [3]).

Notice that while the definition of EDS naturally comes from the definition of the witnessing game of the previous section, it is reasonable that it should also be similar to the criteria in [3], although whereas we are using EDS to prove width lower bounds in a system (narrow resolution) where initial width is unimportant, Atserias and Dalmau give criteria to obtain good width lower bounds for narrow formulas obtained from large initial width formulas by using the standard method of extension variables.

3.1 A sufficient condition for extended dynamic satisfiability

We will treat CNFs as two sorted structures, with a clause sort and a variable sort and two binary relations "variable x appears positively in clause C" and "variable x appears negatively in clause C". We describe a kind of pebble game, the $(1, k)$-embedding game. The game is played between a Spoiler and a Duplicator on the set of clauses and variables of two CNFs F and G. Each player has k variable pebbles and one clause pebble.

Each turn the Spoiler either plays a clause pebble on one of the clauses of F, in which case the Duplicator must play his corresponding pebble on one of the clauses of G; or the Spoiler plays a variable pebble on one of the variables of either CNF, in which case the Duplicator must place his corresponding pebble on one of the variables of the other CNF.

The aim of the Duplicator is to make sure that the positions of the pebbles give rise to a partial isomorphism. If at any point it is not an isomorphism, then the Spoiler has won.

Theorem 12 *Suppose G is satisfiable, and there is a winning strategy for the Duplicator in this game. Then there is a winning strategy for the Delayer in the k pebble witnessing game on F. Hence F is k-EDS.*

Proof Suppose the Prover queries a variable x in F. To find out how the Delayer responds, let the Spoiler pebble x in the $(1, k)$-embedding game. The Duplicator matches with this by pebbling some variable y in G. The Delayer responds to the Prover with the truth value given to y by our satisfying assignment to G.

If the Prover queries a clause C of F, let the Spoiler pebble that clause in the embedding game. The Duplicator responds with some clause D in G. D has at least one literal made true by our assignment. The Spoiler now pebbles the corresponding variable in G, and the Duplicator matches it with a variable in F. The Delayer then responds to the prover with this variable. $\square$

This theorem is not easy to use, because it is not necessarily easy to prove that two CNFs F and G are in this relationship (although a sufficient condition is for F and G to be indistinguishable in the normal $k + 1$ pebble game of finite model theory).

We give an easier-to-use, but weaker, sufficient condition for extended dynamic satisfiability below, for the case where F and G are propositional translations of some first order principle. The argument is a standard one, see Krajicek [16, 15] or Riis [19] although we do not insist that the structure in which the principle is satisfied is infinite.

Theorem 13 *Let F be a CNF. Let ϕ be a first order quantifier free formula in a relational language L with equality. Let Φ be the formula*

$$\forall x_1 \in [n_1] \ldots \forall x_k \in [n_k] \exists x_{k+1} \in [n_{k+1}] \ldots \exists x_l \in [n_l] \phi(\bar{x})$$

where $n_1, \ldots, n_l \in \mathbb{N}$.

Suppose that Φ is satisfiable with larger bounds, that is, there is an interpretation in $\mathbb{N}$ of the relation symbols from L such that, with this interpretation,

$$\mathbb{N} \models \forall x_1 \in S_1 \ldots \forall x_k \in S_k \exists x_{k+1} \in S_{k+1} \ldots \exists x_l \in S_l \phi(\bar{x})$$

where $S_1, \ldots, S_l$ are (possibly infinite) subsets of $\mathbb{N}$ with $|S_j| \geq n_j$ for each j and $n_i \geq n_j \rightarrow S_i \supseteq S_j$ for each i, j.

Let $\langle \Phi \rangle$ be the formula

$$\bigwedge_{i_1 \in [n_1], \ldots, i_k \in [n_k]} \bigvee_{i_{k+1} \in [n_{k+1}], \ldots, i_l \in [n_l]} \phi(\bar{i}).$$

We treat this is a propositional formula in the usual fashion, thinking of atomic sentences involving a relation symbol as propositional variables, and of atomic sentences involving equality between numbers as the appropriate one of the connectives $\{T, F\}$.

Let $n = \min\{n_1, \ldots, n_l\}$ and let r be the maximum arity of any relation symbol. Then $\langle \Phi \rangle$ is $(\frac{n}{r} - l + 1)$-EDS (l is the number of variables).

It follows that if F is a CNF, and we can fix a one-to-one renaming of the propositional variables of $\langle \Phi \rangle$ with respect to which every clause of F is implied by some clause of $\langle \Phi \rangle$, and every variable in F appears in $\langle \Phi \rangle$, then F is $(\frac{n}{r} - l + 1)$-EDS. (We call this condition "F is covered by Φ".)

Proof

Let Q be the set of partial injections from $\mathbb{N}$ into $\mathbb{N}$ such that for each j, if $x \in dom(f)$ and $x \in [n_j]$ then $f(x) \in S_j$.

For each $f \in Q$, we define a partial assignment α_f to the variables in $\langle \Phi \rangle$ as follows. Suppose P is a relation symbol in L, of some arity s. Suppose $i_1, \ldots, i_s \subseteq dom(f)$. Then α_f assigns $P(\bar{i})$ as true if $P(f(i_1), \ldots, f(i_s))$ is true in our interpretation, and as false if it is false.

Let R be the closure of $\{\alpha_f : f \in Q\}$ under subset. We claim that R witnesses that $\langle\Phi\rangle$ is $(\frac{n}{r} - l + 1)$-EDS.

To show the claim, suppose that $\alpha \in R$ and $|\alpha| \leq \frac{n}{r} - l$. Then $\alpha \subseteq \alpha_f$ for some $f \in Q$ and we may assume $|f| \leq (\frac{n}{r} - l)r$, so $|f| \leq n - l$.

We need to show that we can extend α to satisfy any clause C in $\langle\Phi\rangle$. C has the form

$$\bigvee_{i_{k+1}\in[n_{k+1}],\ldots,i_l\in[n_l]} \phi(\bar{i})$$

for some $i_1, \ldots, i_k$ with each $i_j \in [n_j]$.

Extend f to $f' \in Q$ such that $i_1, \ldots, i_k \subseteq \mathit{dom}(f')$. We can do this because $|f| \leq n - l$ so we have at least l numbers available in $\bigcap_j S_j$ that f does not yet map anything to.

We know that, in our interpretation,

$$\mathbb{N} \models \exists x_{k+1} \in S_{k+1} \ldots \exists x_l \in S_l \phi(f'(i_1), \ldots, f'(i_k), x_{k+1}, \ldots, x_l).$$

Let $x_{k+1}, \ldots, x_l$ be witnesses to these existential quantifiers. Extend f' to $f'' \in Q$ such that $x_{k+1}, \ldots, x_l \subseteq \mathit{ran}(f''|[n])$. We can do this because $|f| \leq n - (l - k)$ so we have $l - k$ numbers available in $[n]$ that f' does not yet map to anything.

Hence for some $i_{k+1} \in [n_{k+1}], \ldots, i_l \in [n_l]$ the assignment $\alpha_{f''}$ satisfies $\phi(\bar{i})$. So let g be any total function in Q extending f''. Then α_g satisfies the clause C.

Given any propositional variable in $\langle\Phi\rangle$, we can extend α in a similar way so that it assigns a value to the variable, again using that $|f| \leq n - l$. $\Box$

Corollary 14 *LOP_n is $(\frac{n}{2} - 3)$-EDS.*

Proof LOP_n is covered by the formula $\forall x, y, z \in [n] \exists w \in [n], (\neg P(x, y) \vee \neg P(y, x)) \wedge (P(x, y) \vee P(y, x)) \wedge (P(x, y) \wedge P(y, z) \rightarrow P(x, z)) \wedge P(w, x)$, and this is satisfiable if we let the quantifiers range over all of $\mathbb{N}$ and interpret P as any total ordering with no least element. $\Box$

4 Pebbling games and subsystems of resolution

We represent (usually partial) assignments by the following notation: $[x_1 \mapsto 1, \ldots, x_n \mapsto 0]$. Given an assignment α, we denote by $\overline{\alpha}$ the negation of α, that is, if α is the assignment $[x \mapsto 1, y \mapsto 0, z \mapsto 1]$, then $\overline{\alpha}$ is the clause $(\neg x \vee y \vee \neg z)$. Given a set S of assignments, we denote by $\overline{S}$ the set of clauses obtained by the negation of all assignments in S.

Definition 15 (Structure) *Let F be a CNF formula. For each clause C of F, let S_C be a set of partial assignments to the variables of C, such that*

1. *each assignment in S_C satifies C*
2. *C implies the disjunction of the assignments (in other words, $C \cup \overline{S_C}$ is a contradictory set of clauses).*

We call $\mathcal{S} = \bigcup_{C \in F} S_C$ a structure *for F.*

The *Structured Witnessing Game* $\mathcal{SWG}(F, \mathcal{S}_F)$, over a CNF F and with a structure $\mathcal{S}_F$, is a two player (Prover and Delayer) game defined as follows.

At each round the Prover either

- puts a pebble on some clause C of F. Then the Delayer answers by choosing one assignment $\alpha \in S_C$, and labelling the pebble with it; the Delayer is not allowed to choose an assignment inconsistent with an assignment already in play
- or removes a pebble, together with with its label.

The game ends when the Delayer is unable to choose an assignment consistently. This only happens when the the assignments labelling all the pebbles in play together falsify some initial clause of F.

We say that a formula F is $(k, \mathcal{S})$-easy if the Prover has a winning strategy for the game $\mathcal{SWG}(F, \mathcal{S})$ using at most k pebbles simultaneously. Otherwise the Delayer has a winning strategy, and we say that F is $(k, \mathcal{S})$-hard.

We will look at three structures in this paper.

- In the *unary* structure $\mathcal{U}_F$, for each $C \in F$,

$$\mathcal{U}_C = \bigcup_{l \in C} [l \mapsto 1]$$

  In other words, assignments in $\mathcal{U}_C$ satisfy exactly one literal in C. Note that the game for this structure is similar to the witnessing game described above, except that here the Prover is not allowed to query the value of individual variables.

- In the *ordered* structure $\mathcal{O}_F$, for each clause $C \in F$ we first fix a total order $\prec$ on the variables of C (extended to literals by ignoring negations), then define

$$\mathcal{O}_C = \bigcup_{l \in C} ([l \mapsto 1] \cup \bigcup_{r \in C, r \prec l} [r \mapsto 0])$$

- In the *full structure* $\mathcal{F}_F$, for each clause C we let S_C be the set of all possible assignments to all of the variables in C.

In lemma 24 we will prove that the PHP_n is $(n/2-1, \mathcal{F})$-hard. On the other hand in the next subsection we will prove that LOP_n formulas are $(3, \mathcal{F})$-easy, and that this gives us an upper bound on the size of refutations of LOP_n.

Clearly the more complex the structure is, the more information the Prover can get using fewer pebbles, and the easier it will be to force the Delayer into a contradiction. This is captured by the following lemma.

Definition 16 *Let $\mathcal{S}_F$ and $\mathcal{T}_F$ be two structures for F. We write $\mathcal{S}_F \leq \mathcal{T}_F$ if for all $C \in F$, it holds that: (1) for all $\alpha \in S_C$ there is a $\beta \in T_C$ such that $\beta \subseteq \alpha$; (2) for all $\beta \in T_C$ there is $\alpha \in S_C$ such that $\beta \subseteq \alpha$.*

Lemma 17 *If F is $(k, \mathcal{S}_F)$-hard and $\mathcal{S}_F \leq \mathcal{T}_F$, then F is $(k, \mathcal{T}_F)$-hard.* $\square$

4.1 Feasible Structured Games and Short Refutations

Definition 18 (Feasible Structure) *Let F be a CNF formula. We say that a structure $\mathcal{S}_F = \bigcup_{C \in F} S_C$ is* feasible *if there are two polynomials p and q, such that for all $C \in F$:*

- *$|S_C| \leq p(|F|)$;*
- *there is a Resolution refutation of $C \cup \overline{S}_C$ of size bounded by $q(|F|)$.*

From a winning strategy for the Prover in a feasible structured game, we can construct a resolution refutation of F.

Theorem 19 *Let F be CNF over n variables and a let $\mathcal{S}_F$ be a feasible structure for F. If F is $(k, \mathcal{S}_F)$-easy, then there is a resolution refutation of F of size bounded by $O(m^k|\mathcal{S}_F|^k)q(|F|)$.*

Proof Assume F has n variables and m clauses. F is $(k, \mathcal{S}_F)$-easy, so the Prover has a winning strategy for the structured game on F using at most k pebbles.

A position in the game is formally a tuple $(C_1, \alpha_1), \ldots, (C_r, \alpha_r)$ where each C_i is a clause from F on which a pebble is in play and $\alpha_i \in S_{C_i}$ is the partial assignment with which the Delayer has labelled that pebble. So there are $\sum_{i=1}^{k} \binom{m}{i} |S_F|^i \leq O(m^k|S_F|^k)$.

Our first step in constructing the refutation is to write down the Prover's strategy as a directed acyclic graph. The nodes in the graph will be positions in the game. The edges will be defined as follows:

- if the position w is reached from position v after the Prover pointed at a clause C and the Delayer answering with assignment α, then we put a directed edge from v to w and label it with (C, α) (or just with C, if the Delayer is unable to play)
- if position w is reached from position v by removing the clause C and its label, we put a directed edge from v to w and label it with C.

There are two other special kind of nodes in the graph: a single source node without any label, and sink nodes, one for each clause in F.

The game starts with no pebbles in play, this is the empty position at the single source node of our graph. When the Delayer has labelled the pebbles in a way that falsifies some clause C of F, then we have an edge going into a the sink node corresponding to C. The Prover's strategy guarantees that he will win, so every path from the source must lead to a sink. Hence our graph is a dag.

Given the strategy of the Prover, first we build the associated dag. Notice that its size is $\leq O(m^k |S_F|^k)$. Recall from the start of this section the notation $\bar{\alpha}$ for an assignment α.

To make the graph into a resolution refutation, we first reverse the directions of all the arrows. We associate with each node corresponding to a position $(C_1, \alpha_1), \ldots, (C_r, \alpha_r)$ the clause $\overline{\alpha_1} \vee \ldots \vee \overline{\alpha_r}$.

The single sink node will be labelled with the empty clause. Notice that if a node corresponding to a position $(C_1, \alpha_1), \ldots, (C_r, \alpha_r)$ has an edge labelled (C) coming to it from a source node corresponding to the clause C, then $\overline{\alpha_1} \vee \ldots \vee \overline{\alpha_r}$ is derivable by one weakening from C.

This will give us something that is almost a refutation of F. For the internal nodes, suppose at position $(C_1, \alpha_1), \ldots, (C_r, \alpha_r)$ in our original strategy, the Prover played a pebble on the clause B, and $S_B = \{\beta_1, \ldots, \beta_m\}$ was the set of possible replies the Delayer could have made. Let $\Gamma = \overline{\alpha_1} \vee \ldots \vee \overline{\alpha_r}$. In our almost-refutation, this corresponds to a node labelled with the disjunction Γ, with edges coming into it from m nodes labelled $\Gamma \vee \overline{\beta_1}, \ldots, \Gamma \vee \overline{\beta_m}$. By the definition of a feasible structure, there is a resolution refutation of $B \wedge \overline{\beta_1} \wedge \ldots \wedge \overline{\beta_m}$ of size bounded by $q(|F|)$. By adding the disjunction Γ to each clause in this refutation, we get a resolution proof of Γ from F together with the clauses $\Gamma \vee \overline{\beta_1}, \ldots, \Gamma \vee \overline{\beta_m}$. We put this proof into our graph in place of the edges coming into the node Γ.

The case when the Prover deletes a pebble follows by weakening and we leave it to the reader.

This completes the construction, and the size of the refutation at the end is $\leq O(m^k |S_F|^k) q(|F|)$. $\square$

Lemma 20 *The unary and the ordered structures are feasible.*

Proof
Let F be a CNF. From the definitions of our structures we have that for all clauses C in F, $|\mathcal{U}_C| = |\mathcal{O}_C| = |C| \leq w(F)$. Let $C = (l_1 \vee \ldots \vee l_k)$ be a clause in F. In the case of the unary structure there is a straightforward treelike resolution refutation of size k of $C \wedge \overline{\mathcal{U}}_C$, since $\overline{\mathcal{U}_C}$ is $\bigwedge_{i=1,\ldots,k} \neg l_i$.

In the case of the ordered structure we have a daglike resolution refutations of $C \wedge \overline{\mathcal{O}_C}$ as follows. Assume w.l.o.g. that the order of the literals in C is $l_1 \prec \ldots \prec l_k$. The proof proceeds in k stages $i = 1, \ldots, k$. At stage i we produce a daglike proof of the clause $(l_i \vee \ldots \vee l_k)$ and of the clause $\neg l_i$. Hence after the k-th stage ends we immediately get the empty clause. Stage 1 is immediate since $(l_1 \vee \ldots \vee l_k)$ is in C and $\neg l_1$ is a clause in $\overline{\mathcal{O}_C}$. Assume stage i is completed and we have already proofs of $(l_i \vee \ldots \vee l_k)$ and $\neg l_1, \ldots \neg l_i$. The clause $(l_{i+1} \vee \ldots \vee l_k)$ is obtained by resolving $(l_i \vee \ldots \vee l_k)$ with $\neg l_i$. The clause $\neg l_{i+1}$ is obtained by resolving the clause $(l_1 \vee \ldots \vee l_i \vee \neg l_{i+1})$ of $\overline{\mathcal{O}_C}$ in sequence with $\neg l_1, \ldots \neg l_i$. It is easy to see that this size of this proof is bounded by $O(k^2)$ and hence is polynomial in n. □

4.2 Structured Games as Subsystems of Resolution

In [13] it is proved that k-dynamic satisfiability is a sufficient condition for a *CNF* F to require exponential size treelike resolution refutations. In the next theorem we prove that it completely characterizes Delayer strategies for the unary structure.

Theorem 21 *F is k-dynamic satisfiable iff it is $(\mathcal{U}, k)$-hard.*

Proof Suppose F is k-DS. Let R be the family of partial assignments witnessing this. We use R to give a strategy for the Delayer in the unary game. The Delayer will maintain the invariant that in every configuration $(C_1, [l_1 \mapsto 1]), \ldots, (C_r, [l_r \mapsto 1])$ that appears in play (where the l_i are literals), the assignment $\alpha = [l_1 \mapsto 1, \ldots, l_r \mapsto 1]$ is in R. When the Prover presents a new clause $C \in F$, since $\alpha \in R$ and $|\alpha| < k$, there is a $\beta \in R$ extending α such that β satisfies C. Take one literal l in C set to 1 by β and consider the assignment $[l_1 \mapsto 1, \ldots, l_r \mapsto 1, l \mapsto 1] \subseteq \beta$. This is in R because R is closed under inclusion. So the Delayer can reply to C with $[l \mapsto 1]$.

Conversely, suppose that F is $(\mathcal{U}, k)$ hard. Then there is a winning strategy for the Delayer in the unary game. Let R be the set of partial assignments $[l_1 \mapsto 1, \ldots, l_r \mapsto 1]$ such that the configuration $(C_1, [l_1 \mapsto$

1]), ..., $(C_r, [l_r \mapsto 1])$ appears in some game played with this strategy. Then R witnesses that F is k-DS. $\square$

Extended dynamic satisfiability corresponds to the witnessing game, in which the Prover is allowed to query variables. That is not allowed in our structured games, but we do have the following relationship.

Theorem 22 *If F is $(\mathcal{F}, k)$-hard (that is, hard for the full game) then F is k-EDS.*

Proof Suppose the Delayer has a winning strategy for the k pebble full game on F. Let R' be the family of all partial assignments of the form $\bigcup_{i=1}^{r} \alpha_i$ where $(C_1, \alpha_1), \ldots, (C_r, \alpha_r)$ is a configuration appearing in some game played according this strategy. Let R be the closure of R' under inclusion.

Then R witnesses that F is k-EDS. For suppose $\alpha \in R$, $|\alpha| < k$. Then we may assume that α arises from a configuration in which $< k$ pebbles are in play. So if C is any clause in F, by playing a pebble on C we can find β in R extending α and satisfying C (and in fact β assigns a value to all variables in C). Also if x is any variable, choose any clause C in which x appears. Then we can extend α to an assignment assigning a value all variables in C, in particular to x. $\square$

Now we will consider strategies for the Prover and Delayer for two standard families of CNFs, the pigeonhole principle PHP^m_n:

$$\bigwedge_{i\in[m]} \bigvee_{j\in[n]} p_{ij} \ \wedge \bigwedge_{i,i'\in[m], j\in[n], i\neq i'} (\neg p_{ij} \vee \neg p_{i'j})$$

and the linear ordering principle LOP_n, that expresses (the negation of) that every linear ordering of n elements has a least element:

$$\bigwedge_{i,j,k\in[n]} (\neg x_{i,j} \vee \neg x_{j,k} \vee x_{i,k}) \wedge \bigwedge_{i,j\in[n]} (x_{i,j} \vee x_{j,i}) \\ \wedge \bigwedge_{i,j\in[n]} (\neg x_{i,j} \vee \neg x_{j,i}) \wedge \bigwedge_{i\in[n]} \bigvee_{j\in[n], i\neq j} x_{j,i}.$$

From definition 16 and lemma 17, it is straighforward to observe that:

Lemma 23 *$\mathcal{F} \leq \mathcal{O} \leq \mathcal{U}$.*

It is quite easy to prove that the PHP^m_n is hard for all these games:

Lemma 24 *PHP^m_n is $(\mathcal{F}, n/2-1)$-hard.*

Proof A critical truth assignment is a partial assignment which for each hole $j \in [m]$ only sets one $p_{i,j}$ to 1. The strategy of the Delayer is to answer the Prover in such a way that her labels are always consistent with some critical truth assignment. Now by the form of the clauses of PHP, a set of at most $n/2-1$ clauses mentions at most $n-2$ pigeons, and if a critical truth assignment assigns $< n-2$ pigeons we can always extend it to one that assigns one more pigeon. This shows that PHP^m_n is $(\mathcal{F}, n/2)$-hard. $\Box$

On the other hand, using the fact that LOP_n is $(\frac{n}{2}-3)$-dynamically satisfiable (see [11] or corollary 14), we have:

Corollary 25 *LOP_n is $(\mathcal{U}, \frac{n}{2}-3)$-hard.*

The main result of this subsection follows from the next theorem in which we prove that the linear ordering principle LOP_n is $(\mathcal{O}, 3)$-easy.

Theorem 26 *LOP_n is $(\mathcal{O}, 3)$-easy.*

Proof Consider the following order on all variables that in particular defines an order on each clause: $x_{i,j} \prec x_{h,k}$ iff either $j < k$ or $j = k$ and $i < h$.

We describe the strategy of the Prover by stages: we prove that at each stage the Prover, using only three pebbles, either wins or will force the Delayer to answer to a clause of the form $\bigvee_{j\in[n], j\neq r} x_{j,r}$ with an assignment assigning strictly more literals to 0 than the previous stage. Hence, if he does not win sooner, after at most n stages he will force a clause of this form to be falsified.

Assume w.l.o.g. that at the begining of a stage the Prover pebbles the clause $C_1 = \bigvee_{j\in[n], j\neq 1} x_{j,1}$. Let $\alpha \in \mathcal{O}_{C_1}$ be the assignment chosen by the Delayer. α will be of the form

$$[x_{2,1} \mapsto 0, \ldots, x_{j-1,1} \mapsto 0, x_{j,1} \mapsto 1]$$

for some $j \in [n], j \neq 1$. The Prover then pebbles the clause $C_j = \bigvee_{k\in[n], k\neq j} x_{k,j}$ Let β be the assignment chosen by the Delayer. β is of the form

$$[x_{1,j} \mapsto 0, \ldots, x_{k-1,j} \mapsto 0, x_{k,j} \mapsto 1]$$

for some $k \in [n], k \neq j$.

Now if $k < j$, then $\alpha(x_{k,1}) \mapsto 0$. But then all literals in the clause $(\neg x_{k,j} \vee \neg x_{j,1} \vee x_{k,1})$ are false, and the Prover can pebble this clause and win.

Assume then that $k > j$. The Prover then pebbles the clause $\neg x_{k,j} \vee \neg x_{j,k}$. Since $\beta(x_{k,j}) = 1$, The Delayer must answer with the assignment γ setting $x_{j,k}$ to 0 (and obviously $x_{k,j}$ to 1). At this point the Prover removes the pebbles from C_1 and C_j and places a pebble on C_k. The Delayer must answer with an assignment δ of the form

$$[x_{1,k} \mapsto 0, \ldots, x_{l-1,k} \mapsto 0, x_{l,k} \mapsto 1]$$

where clearly $l > j$, to not contradict the assignment γ. $\square$

Also any formula with a bounded width resolution refutation is easy for the *full* game, since by proposition 4, lemma 10 and theorem 22, if F has width k refutations in resolution then F is $(\mathcal{F}, k+1)$-easy.

4.3 Automatic Generation of Refutations

Lemma 24 shows that the *ordered structure*, via theorem 19 and lemma 20, gives rise to a subsystem of daglike resolution (corresponding to a strategy for the Prover with $O(1)$ pebbles):

- powerful enough to obtain polynomial size refutations of important families of contradictions like LOP_n;
- where PHP (and other classes of formulas we omit in this version) are hard to refute and the hardness proof is relatively easy.

Since LOP_n is an example of formulas known to be hard for many automatic theorem provers (see [5]), the previous properties suggest that it's worth investigating algorithms for generating winning strategies for the Prover, as this gives a way of generating resolution refutations.

We present such an algorithm below. It is analagous to the Ben-Sasson Wigderson algorithm based on width. In our case, rather than limiting the width, we limit the number of pebbles used in the strategy.

We first describe a subroutine. The input is a formula F, and the description of the structure S_F (this may be given in a simple way, e.g. as an ordering if we are dealing with the ordered game), and a number k of pebbles. The output is a winning strategy for the Prover using k pebbles, if one exists, otherwise "NONE".

The subroutine first builds up a dag A as follows: The nodes of A are all of the positions possible in the game, ie. all the possible ways of labelling k or fewer pebbles plus a source node. There are $\sum_{i=1}^{k} i! m^i |S_F|^i \leq O(m^k |S_F|^k)$, where m is the number of clauses in F. At the start of the algorithm all the positions that are self contradictory or falsify clauses of F are *marked*, and the graph has no edges. While the source node has not been marked the algorithm does the following: it checks each not marked node X in A. If by removing a pebble the Prover can move from X to a node Y already marked in A, then the algorithm marks X and labels it with the pebble to be removed and adds an edge from X to Y. If there is a clause C that is not pebbled at X, and is such that whatever label the Delayer could choose to give to C, it would lead to a position corresponding to a node already marked in A, then the algorithm marks X, labels it with C, and adds edges going to all the nodes corresponding to the Delayer's possible answers.

The subroutine stops when the source node has been marked, or when there are no more edges to be added to the graph. If the source node has been marked, the subroutine outputs only the marked subgraph of A, which is a winning strategy for the Prover. Otherwise it outputs NONE.

The proof search algorithm works by calling this subroutine for increasing values of k, until the subroutine outputs a strategy (which it will do eventually, when the Prover is able to query all clauses at once).

The running time of the algorithm is $O(m^{r+1}|S_F|^{r+1})$ where r is the minimal number of pebbles used by the Prover to win $\mathcal{SWG}(\mathcal{S}_{\mathcal{F}}, F)$.

For feasible structures, using theorem 19, this algorithm allows us to generate daglike resolution refutations of size polynomial in the size of the formula.

For the case of ordered structures we could add a preprocessing phase to try all possible orders. Although in the worst case this can increase the running time to exponential we notice that for the case of LOP_n, *any* order of $[n]$ naturally gives rise to an ordering with respect to which the formula is easy.

Moroever we observe also that although the full structure is not feasible in general, it become so in the case of formulas with $O(1)$ initial width. Now by proposition 4, lemma 10 and theorem 22, if F has width k refutations in resolution then F is $(\mathcal{F}, k+1)$-easy, so if there is a constant width formula with a narrow refutation then our algorithm (looking for strategies for the full game) works at least as well on it as the Ben-Sasson and Wigderson algorithm.

5 Satisfiability vs Hardness

In Section 3 we showed how the existence of a satisfiable formula G similar to F gives a good strategy for the Delayer in a certain game, which shows that F has no narrow resolution refutation.

This suggests an attractive idea, that we should look for a sort of soundness and completeness theorem for polynomial size resolution. It would have the following form: a CNF F has no small resolution refutation if and only if F is "almost" satisfiable, in that it is hard to distinguish from a satisfiable formula G.

In this section we give two results in this direction. We first show that if F is hard to distinguish from G in the sense that it is hard to prove in resolution that F and G are different, then F is hard to refute. We then show a converse, although this talks about games rather than proofs: if there is a good strategy for the Delayer in the full (structured) pebble game on F, then there exists a satisfiable CNF G that looks similar to F, in a certain sense.

Definition 27 *Let F and G be CNFs, considered as two-sorted structures. We define a new CNF, $ISO(F, G)$, that expresses the statement "F is isomorphic to G". This is intended to be used when F is not isomorphic to G, as a generalization of the pigeonhole principle. Let $Var(F)$ and $Cl(F)$ be the sets of variables and clauses in a CNF.*

We take the conjunction of

1. $\bigvee_{D \in Cl(G)} \sigma_{CD}$ *for each* $C \in Cl(F)$; $\bigvee_{C \in Cl(F)} \sigma_{CD}$ *for each* $D \in Cl(G)$;
2. $\neg\sigma_{CD} \vee \neg\sigma_{CD'}$ *and* $\neg\sigma_{CD} \vee \neg\sigma_{C'D}$ *for all* $C \neq C' \in Cl(F)$, $D \neq D' \in Cl(G)$;
3. $\bigvee_{y \in Var(G)} \sigma_{xy}$ *for each* $x \in Var(F)$; $\bigvee_{x \in Var(F)} \sigma_{xy}$ *for each* $y \in Var(G)$;
4. $\neg\sigma_{xy} \vee \neg\sigma_{xy'}$ *and* $\neg\sigma_{xy} \vee \neg\sigma x'y$ *for all* $x \neq x' \in Var(F)$, $y \neq y' \in Var(G)$;
5. *If x appears positively in C in F, but y does not appear positively in D in G, then we include the clause $\neg\sigma_{xy} \vee \neg\sigma_{CD}$; similarly for appearing negatively or not appearing at all.*

So 1 and 2 say that σ is a bijection on clauses, 3 and 4 say it is a bijection on variables, and 5 says it preserves the structure. The total number of clauses in $ISO(F, G)$ is $O(|F|^2|G|^2)$ (where $|F|$ is the number of clauses plus the number of variables).

Theorem 28 *Suppose that G is satisfiable and F has a small resolution refutation. Then $ISO(F, G)$ has a small resolution refutation.*

Proof Let α be a satisfying assignment to G. α partitions $Var(G)$ into a set A of true variables and a set B of false variables. For $x \in Var(F)$, let X be the clause $\bigvee_{y \in A} \sigma_{xy}$ and let $\bar{X}$ be the clause $\bigvee_{z \in B} \sigma_{xz}$. Let F^* be F with every variable x replaced by X and every negated variable $\neg x$ replaced by $\bar{X}$.

We will show how to derive F^* from $ISO(F,G)$. Then we can use the refutation of F to give a refutation of F^*: from two clauses $C \cup X$ and $D \cup \bar{X}$ we can derive $C \cup D$ by many resolutions with clauses of the form $\neg\sigma_{xy} \vee \neg\sigma_{xy'}$ from $ISO(F,G)$. In particular, for each $z \in B$ we can derive $C \cup \neg\sigma_{xz}$, then resolve all these with $D \cup \bar{X}$.

So let C be any clause of F. For each clause of D, there are two cases: either some $y \in A$ appears positively in D, or some $z \in B$ appears negatively in D.

In the first case, in $ISO(F,G)$ we have the clause $\bigvee_{x \in Var(F)} \sigma_{xy}$ and since y appears positively in D, for each x that does not appear positively in C we have the clause $\neg\sigma_{xy} \vee \neg\sigma_{CD}$. Resolving these together, we get $\sigma_{CD} \rightarrow \bigvee\{\sigma_{xy} : x \text{ appears positively in } C\}$. Then by several weakenings we get $\sigma_{CD} \rightarrow \bigvee\{\sigma_{xu} : x \text{ appears positively in } C \text{ and } u \in A\}$.

The second case is similar, and gives us

$$\sigma_{CD} \rightarrow \bigvee\{\sigma_{xv} : x \text{ appears negatively in } C \text{ and } v \in B\}.$$

If we obtain one of the above clauses for every clause D of G, we can resolve with $\bigvee_{D \in Cl(G)} \sigma_{CD}$ to get

$$\bigvee\{\sigma_{xu} : x \text{ appears positively in } C \text{ and } u \in A\} \\ \vee \bigvee\{\sigma_{xv} : x \text{ appears negatively in } C \text{ and } v \in B\}$$

which is exactly the required translation of C. □

To use this to get lower bounds for F, we need lower bounds for $ISO(F,G)$. This is a generalization of the pigeonhole principle, so the many existing tools for showing lower bounds for PHP may be useful here. The most tractable case should be when F and G both arise as translations of some first order combinatorial principle, but on structures of slightly different sizes, so we can use the hardness of PHP directly. Krajicek gives an argument like this is in [16], relating the proof complexity of the weak pigeonhole principle and the Ramsey theorem. Potentially this will give a sufficient criterion for a formula to have no small daglike refutation, similar to Riis' criterion for treelike refutations [19].

So far we have given "soundness" theorems, that if F is almost a satisfiable CNF then F is hard to refute (or that it is hard for the Prover to win some game). Now we give a "completeness" theorem.

We think of CNFs as two sorted structures with a clause sort and a variable sort. There are two binary relations, "x appears positively in C" and "x appears negatively in C."

We define the sense in which our constructed CNF G will be similar to F. Informally, we have "local" embeddings of the clauses of F into the clauses of G. (If we had a global embedding, then the satisfying assignment for G would immediately give a satisfying assignment for F.)

Definition 29 *Let F and G be CNFs. We say that F is strongly k-clause embeddable in G if there is a family H of partial isomorphisms from F to G with the following properties:*

1. *Suppose $f \in H$ maps clauses $C_1, \ldots, C_k$ and no other clauses. Then the variables mapped by f are precisely the variables appearing in $C_1, \ldots, C_k$. In other words, f is first of all a partial isomorphism on clauses, but brings with it a partial isomorphism on the variables appearing in those clauses.*

2. *If $f \in H$ maps fewer than k clauses, and C is any other clause in F, then there is $g \in H$ that extends f and maps C somewhere.*

Proposition 30 *If F is strongly k-clause embeddable in G and G is satisfiable, then the Delayer wins the k-pebble full game on F.*

Proof The family of embeddings from F into G gives rise to a family of partial assignments to F which can be used as a strategy for the Delayer in the full game (compare lemma 10). $\square$

Theorem 31 *Suppose the Delayer wins the k-pebble full game on F. Then there is a satisfiable finite CNF G such that F is strongly k-clause embeddable in G.*

Proof We will build G out of the Delayer's strategy for the game. Think of this strategy as a dag consisting of positions in the game. Each position P is a tuple $(C_1, \alpha_1), \ldots, (C_r, \alpha_r)$ of at most k clauses from F together with assignments to all the variables appearing in each clause, that are consistent and satisfy each clause. Notice that only a finite number of different positions appear in the strategy.

Let G' be the CNF whose clauses are all the pairs (C, α) that appear in the Delayer's strategy. We give variables to the clauses as follows: if $C \in F$ contains variables $x_1, \ldots, x_m$, then (C, α) contains variables $(x_1, C, \alpha), \ldots, (x_m, C, \alpha)$ and (x_i, C, α) appears positively or negatively in (C, α) just as x_i appears positively or negatively in C. That is, G' consists of clauses named by pairs (C, α) and variables named by triples (x, C, α); each clause is isomorphic to some clause from F, and no two clauses share any variables.

Now define an equivalence relation $\sim$ on the variables of G' as the transitive closure of the relation: (x, C, α) is related to (y, D, β) if and only if x and y are the same variable (in F) and (C, α) and (D, β) appear together in some position in the Delayer's strategy.

Let G be the CNF obtained from G' by renaming every variable (x, C, α) with its $\sim$-equivalence class $[(x, C, \alpha)]$.

Claim 1 G is satisfiable.

Consider the assignment $A : [(x, C, \alpha)] \mapsto \alpha(x)$. This is well-defined, because if x appears in clauses C and D in F, and if (C, α) and (D, β) appear together in some position, then α and β must assign the same value to x (or the Prover would win at that point).

Furthermore this assignment satisfies every clause in G, because (C, α) only appears in the Delayer's strategy if α satisfies C (or again the Prover would win).

Note also that the existence of this assignment means that G contains no repeated clauses. A repeated clause would appear if G' contained distinct clauses (C, α) and (C, β) such that for every variable $x \in C$, $(x, C, \alpha) \sim (x, C, \beta)$, so that after the renaming these two clauses would have the same literals. This cannot happen, because if $\alpha \neq \beta$ then there will be some x such that $\alpha(x) \neq \beta(x)$ and the assignment A will not be well-defined.

Claim 2 F is strongly k-clause embeddable in G.

Let $P = \langle (C_1, \alpha_1), \ldots, (C_r, \alpha_r) \rangle$ be any position in the Delayer's strategy, and define a partial mapping f_P from F to G as

$$\begin{aligned} f_P : C \mapsto & (C_i, \alpha_i) \text{ if } C \text{ appears as some } C_i \\ & \text{unmapped otherwise} \\ x \mapsto & [(x, C_i, \alpha_i)] \text{ if } x \text{ appears in some } C_i \\ & \text{unmapped otherwise} \end{aligned}$$

This mapping is injective on variables because two variables in G' are only $\sim$-equivalent if they arise from the same variable in F; and if x occurs more than once in a position then by the definition of $\sim$ those occurrences are equivalent. It is injective on clauses because each clause only appears once in any position.

Suppose x appears positively in C_i in F. Then (x, C_i, α_i) appears positively in (C_i, α_i) in G', so $[(x, C_i, \alpha_i)]$ appears positively in (C_i, α_i) in G.

Suppose conversely that x does not appear positively in C_i. Then in G', no variable arising from x appears positively in any clause arising from C_i. $[(x, C_i, \alpha_i)]$ only consists of variables in G' arising from x, hence $[(x, C_i, \alpha_i)]$ does not appear positively in (C_i, α_i) in G.

The same is true for appearing negatively. Hence f_P is a partial isomorphism, and the family $H := \{f_P : P \text{ is a position in the Delayer's strategy}\}$ witnesses that F is strongly k-clause embeddable in G. $\square$

References

[1] M. Alekhnovich, E. Ben-Sasson, A. Razborov, A. Wigderson. Space complexity in propositional calculus. *SIAM J. Comput.* 31(4) pp. 1184–1211, 2002.

[2] M. Alekhnovich, A.A. Razborov. Resolution is not automatizable unless $W[P]$ is not tractable. *42nd IEEE Symposium on Foundations of Computer Science, FOCS 2001*, pp. 210–219.

[3] A. Atserias, v. Dalmau. A combinatorial characterization of Resolution Width *18th IEEE Conference on Computational Complexity* (CCC), pp. 239-247, 2003.

[4] P. Beame, R.M. Karp, T. Pitassi, M.E. Saks. On the Complexity of Unsatisfiability Proofs for Random k-CNF Formulas. *SIAM J. Comput.* 31(4) pp. 1048–1075, 2002.

[5] P. Beame, H. Kautz. A Sabharwal Understanding the power of clause learning *Proceedings IJCAI* pp. 1194–1201, 2003

[6] P. Beame, T. Pitassi. Simplified and Improved Resolution Lower Bounds. *37th IEEE Symposium on Foundations of Computer Science, FOCS 1996*, pp. 274–282.

[7] E. Ben-Sasson, N. Galesi. Space Complexity of Random Formulae in Resolution. *16th IEEE Annual Conference on Computational Complexity, CCC 2001*, pp. 42–51.

[8] E. Ben-Sasson, R. Impagliazzo, A. Wigderson. Near optimal separation of treelike and general Resolution. *Electronic Colloquium on Computational Complexity (ECCC) TR00-005, 2000.* To appear in *Combinatorica.*

[9] E. Ben-Sasson, A. Wigderson. Short Proofs Are Narrow—Resolution Made Simple. *J. ACM* 48(2) pp. 149–168, 2001.

[10] M.L. Bonet, N. Galesi. Optimality of Size-Width Tradeoffs for Resolution. *Computational Complexity*, Vol 10(4) 2001. pp. 261-276.

[11] J.L. Esteban, N. Galesi, J. Messner. On the Complexity of Resolution with Bounded Conjunctions. *Theoretical Computer Science* 321(2-3) pp. 347–370, 2004.

[12] J.L. Esteban, J. Torán. Space bounds for Resolution. *Inform. and Comput.* 171 (1) pp. 84–97, 2001.

[13] N. Galesi, N. Thapen. The Complexity of treelike Systems over λ local formuale *Proceedings of IEEE COnference on Computational Complexity* 2004.

[14] A. Haken. The Intractability of Resolution. *Theoret. Comp. Sci.* 39, pp. 297–308, 1985.

[15] J. Krajíček. Bounded arithmetic, propositional logic, and complexity theory, Encyclopedia of Mathematics and Its Applications, Vol. **60**, *Cambridge University Press,*(1995),

[16] J. Krajíček. On the weak pigeonhole principle. *Fund. Math.* 170(1-3) pp. 123–140, 2001.

J. Symbolic Logic 59(1) pp. 73–86, 1994.

[17] P. Pudlak. Proofs as Games. *American Math. Monthly*, Vol. 2000-2001, pp.541-550

[18] P. Pudlák, R. Impagliazzo. A lower bound for DLL algorithms for k-SAT". *Conference Proceeding of Symposium on Distributed Algorithms* (2000), pp. 128-136.

[19] S. Riis. A complexity gap for tree-resolution. *Computational Complexity* 10(3), pp. 179-209, 2001.

[20] A. Urquhart. Hard examples for Resolution. *J. ACM* 34(1) pp. 209-219, 1987.