Proof Complexity as a Computational Lens 2025/26 Dec 9, 2025

Lecture 13

Lecturer: Jakob Nordström *Scribe: Jakob Nordström, Léo Perrin, Add-Scribe-Here*

In this lecture, we change focus to the complexity measure of space in proof complexity. We will focus on resolution, although the definition we provide will be straightforward to extend to other proof system. First of all though, we will talk about *pebble games*, which turn out to be useful in all kinds of ways when thinking about space complexity.

1 Pebble Games

Pebbling is a tool for studying time-space relationships by means of a game played on directed acyclic graphs (DAGs). This game models computations where the execution is independent of the input and can be performed by straight-line programs. Each such program is encoded as a graph, and a pebble on a vertex in the graph indicates that the corresponding value is currently kept in memory. The goal is to pebble the output vertex of the graph with minimal number of pebbles (amount of memory) and steps (amount of time).

Pebble games were originally devised for studying programming languages and compiler construction, but have found a broad range of applications in computational complexity theory. An excellent survey of pebbling up to ca 1980 is [Pip80], and another in-depth treatment of some pebbling-related questions can be found in [Sav98, Chapter 10]. Some more recent developments are covered in the upcoming survey [Nor25] (which will be finished any decade now).

The *pebbling price* of a DAG G in the black pebble game captures the memory space, or number of registers, required to perform the deterministic computation described by G. It will turn out that an even more interesting game for us will be the more general *black-white pebble game* modelling nondeterministic computation, which was introduced in [CS76]. In what follows, we refer to vertices having indegree 0 as *sources* and vertices having outdegree 0 as *sinks*. We write $pred(v)$ to denote the immediate predecessors of a vertex v, i.e., all vertices u which have an edge to v.

Definition 1.1 (Black-white pebble game). Let G be a directed acyclic graph (DAG) with a unique sink vertex z. The *black-white pebble game* on G is the following one-player game. At any time t, we have a *pebble configuration* $\mathbb{P}_t = (B_t, W_t)$ of black pebbles B_t and white pebbles W_t on the vertices of G, where $B_t, W_t \subseteq V(G)$ and $B_t \cap W_t = \emptyset$. A *(complete) black-white pebbling* of G, or a *black-white pebbling strategy* for G, is a sequence of pebble configurations $\mathcal{P} = \{\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_\tau\}$ such that $\mathbb{P}_0 = (\emptyset, \emptyset)$, $\mathbb{P}_\tau = (\{z\}, \emptyset)$, and for all $t \in [\tau]$ it holds that $\mathbb{P}_t$ is obtained from $\mathbb{P}_{t-1}$ by one of the following rules:

1. **Black pebble placement on v:** A black pebble may be placed on v provided that v is empty and all immediate predecessors of v are covered by pebbles. More formally, letting $B_t = B_{t-1} \cup \{v\}$ and $W_t = W_{t-1}$ is allowed if $v \notin B_{t-1} \cup W_{t-1}$ and $pred(v) \subseteq B_{t-1} \cup W_{t-1}$. (In particular, a black pebble can always be placed on a source vertex s since $pred(s) = \emptyset$.)

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

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

4. **White pebble removal from v:** If all immediate predecessors of a white-pebbled vertex v have pebbles on them, the white pebble on v may be removed. In particular, a white pebble can always be removed from a source vertex. Formally, letting $B_t = B_{t-1}$ and $W_t = W_{t-1} \setminus \{v\}$ is allowed if $v \in W_{t-1}$ and $pred(v) \subseteq B_{t-1} \cup W_{t-1}$.

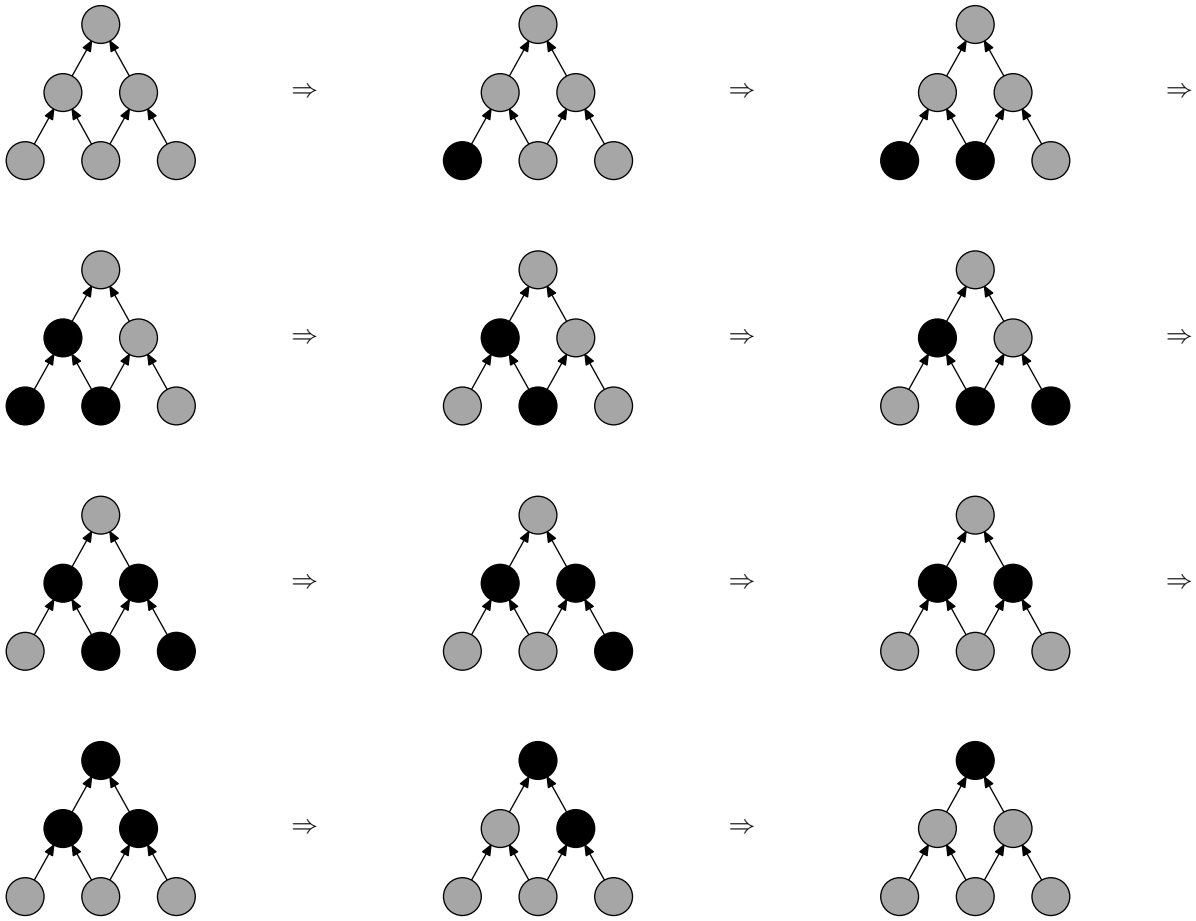


Figure 1: Complete black pebbling $\mathcal{P}_B$ of pyramid graph of height 2.

A *black pebbling* of G is a pebbling using black pebbles only, i.e., having $W_t = \emptyset$ for all t.

Figure 1 depicts the sequence of pebbling moves in an example black pebbling of the so-called *pyramid graph* of height 2, and Figure 2 shows a black-white pebbling of the same graph.

We are interested in measuring the time and space needed to pebble graphs. Time in isolation is not so interesting, since any DAG with $|V(G)| = n$ vertices can be pebbled in time $\mathrm{O}(n)$—just sort the vertices in topological order and then black-pebble the vertices in this order to get a pebbling in linear time and space. However, the minimal space needed can be much less than linear, and if one wants to optimize time and space simultaneously there can be fairly nontrivial relations between the two measures. We will discuss this in more detail in coming lectures.

The next definition makes the concepts of time and space in pebble games more precise and introduces some notation that we will use.

Definition 1.2 (Pebbling time and space). The *time* of a pebbling $\mathcal{P} = \{\mathbb{P}_0, \ldots, \mathbb{P}_\tau\}$ is simply $\mathsf{time}(\mathcal{P}) = \tau$ and the *space* is $\mathsf{space}(\mathcal{P}) = \max_{0 \le t \le \tau}\{|B_t \cup W_t|\}$. We say that G can be pebbled in *simultaneous time τ and space s* if there is a complete pebbling $\mathcal{P}$ with $\mathsf{time}(\mathcal{P}) \le \tau$ and $\mathsf{space}(\mathcal{P}) \le s$.

The *black-white pebbling price* (also known as the *pebbling measure* or *pebbling number*) of G, denoted $\mathsf{BW\text{-}Peb}(G)$, is the minimum space of any complete pebbling of G. The *(black) pebbling price* of G, denoted $\mathsf{Peb}(G)$, is the minimum space of any complete black pebbling of G.

For our example pebbling $\mathcal{P}_B$ in Figure 1 we have $\mathsf{time}(\mathcal{P}_B) = 11$ and $\mathsf{space}(\mathcal{P}_B) = 4$, while for the pebbling $\mathcal{P}_{BW}$ in Figure 2 it holds that $\mathsf{time}(\mathcal{P}_{BW}) = 13$ and $\mathsf{space}(\mathcal{P}_{BW}) = 4$. To get a feel for how to

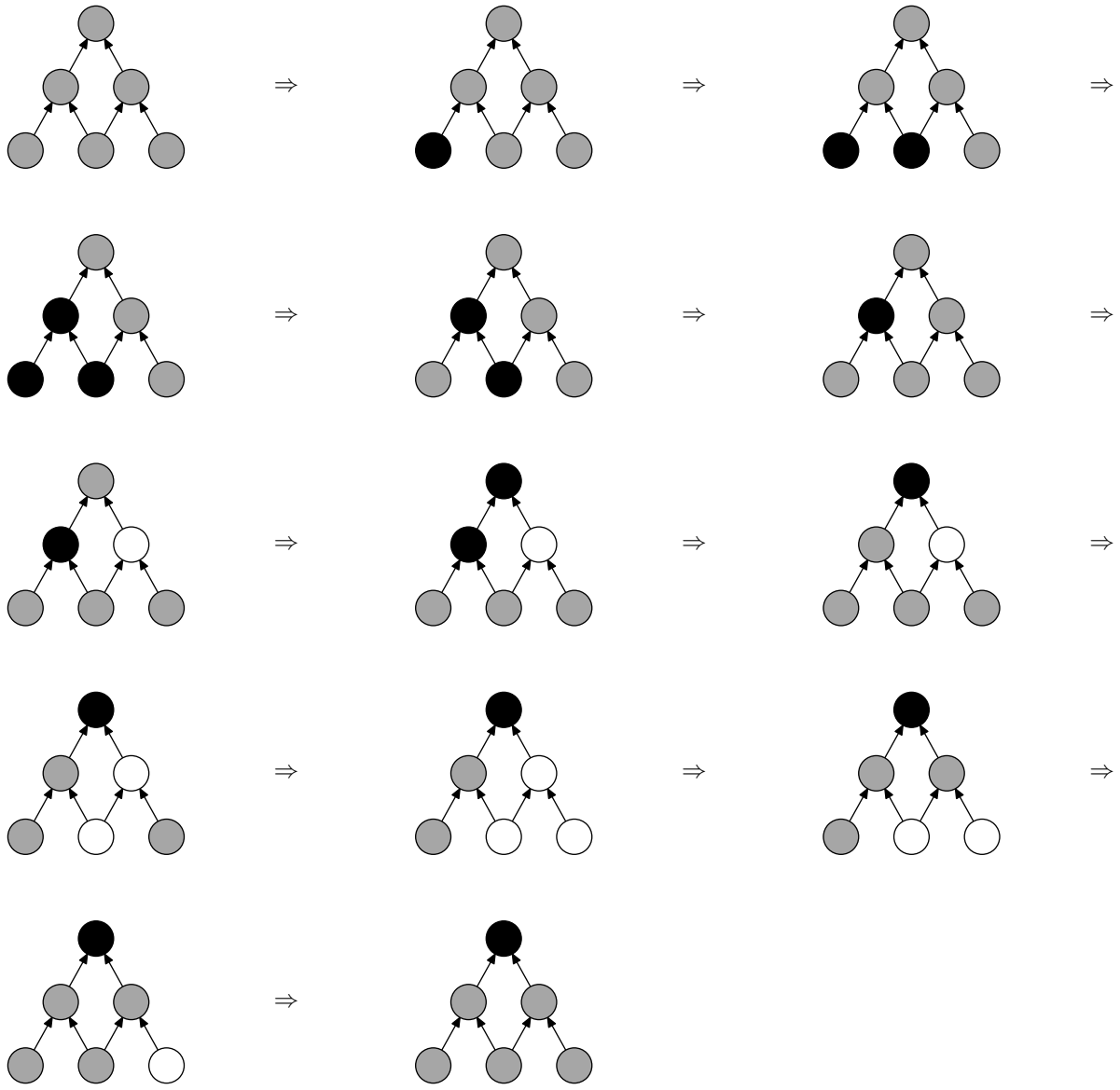


Figure 2: Complete black-white pebbling $\mathcal{P}_{BW}$ of pyramid graph of height 2.

prove pebbling space bounds, and to see why it is not entirely trivial, it might be a good exercise to show that space 4 is optimal for this graph (for black or black-white pebbling).

A classic result in pebbling, and indeed in complexity theory, is that if a graph G as bounded indegree, then it is always possible to save a logarithmic factor over the trivial linear upper bound on space. And you do not need white pebbles for this—there is such a pebbling using black pebbles only.

Theorem 1.3 ([HPV77]). *For any DAG G on n vertices with constant fan-in it holds that* $\mathsf{Peb}(G) = \mathrm{O}(n/\log n)$ *(where the constant hidden in the big-oh notation depends on how large the fan-in is).*

An almost equally classic result is that this is optimal. In the worst case, you can never save more than a logarithmic factor.

Theorem 1.4 ([PTC77, GT78]). *There are explicitly constructible DAGs G_n on n vertices with fan-in 2 such that* $\mathsf{BW\text{-}Peb}(G_n) = \Omega(n/\log n)$.

A nice feature of this result, as stated in the theorem, is that these graphs are *explicit*. That means that we do not need to use, for instance, the probabilistic method to argue that we just know that such graphs

exist but do not know more about them, but that we can in fact give an algorithm that constructs such graphs in an efficient manner (i.e., polynomial time). We will not go into details of the construction here, since it is somewhat elaborate, but the interested reader can refer to [Nor25, Section 7] to see what these graphs look like and how the lower bound is proven.

2 Space Complexity in Resolution

Memory usage is a very important factor for modern state-of-the-art SAT solvers, which makes this a potentially interesting measure to study from an algorithmic point of view. However, space complexity has also been a central part of computational complexity theory since its inception, and so it seems natural to try to study the space measure also in proof complexity from a theoretical point of view, to see if there is a natural way to define space in proof complexity and if interesting results can be proven about such a measure. In today's lecture, we will study space in resolution and prove some upper and lower bounds. When we do this, the width measure pops up again and turns out to be of great importance. Jumping a little bit ahead, we will see that we can get optimal lower bounds on space by proving lower bounds on width and appealing to a general theorem that width is a lower bound for space.

We need to recall the formal definition of space that was mentioned very briefly at the beginning of the course. Before we do so, however, let us first try to get some kind of intuition for what this space measure means.

To get an intuitive sense of what space is, we can think of a resolution proof as being presented on a blackboard. The blackboard represents the information derived so far (analogues to lemmas, propositions, et cetera). We can use such partial result to derive new results. In particular, in a resolution proof we can take two clauses that are written on the blackboard and resolve them to derive a new clause, but we can only resolve clauses that are currently on the board. Since the blackboard has limited size we need to conserve space (a familiar situation during, e.g., a lecture). Therefore, if a clause seems no longer useful at a given point, we can cleverly erase it as it is no longer needed. However, if it turns out later that we were not so clever and the information was needed again after all, then there is nothing to be done about it—we have to rederive this clause from scratch. If it is not on the board, we have forgotten. All we know is what is on the board, and if it is not on the board, then we do not know it. The only exception is that we know the statement of the theorem that we are trying to prove, or rather that we can look it up. For a resolution proof, this means that we have the formula F written on a slip of paper, and we can look at this paper and copy axiom clauses from F to the board. Very loosely speaking, the space of a resolution proof measures the size of a smallest blackboard needed to carry out the proof according to this description.

Three Definitions of Space

1. With respect to linearly ordered proof DAG (which is the definition we have seen before).
2. Configuration-style (as below).
3. The most space-efficient way of realizing any given proof presented as a DAG

Let us now present the definition that we will mainly use when discussing space in resolution and other proof systems.

Definition 2.1 (Configuration-style resolution derivation). A *configuration-style resolution derivation* is a sequence of sets of clauses, or *clause configurations*, $\{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ such that $\mathbb{C}_0 = \emptyset$ and $\mathbb{C}_t$ follows from $\mathbb{C}_{t-1}$ by one of the following rules:

Download: $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{C\}$ for a clause $C \in F$ (an *axiom*).

Erasure: $\mathbb{C}_t \subseteq \mathbb{C}_{t-1}$.

Inference: $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{C \vee D\}$ for a clause $C \vee D$ inferred by the *resolution rule* from $C \vee x, D \vee \overline{x} \in \mathbb{C}_{t-1}$

A *resolution derivation* $\pi : F \vdash D$ of the clause D from the CNF formula F is a derivation $\{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ such that $D \in \mathbb{C}_\tau$. A *resolution refutation* of F is a derivation $\pi : F \vdash \bot$ of the empty clause $\bot$ from F.

Using this formal definition, we can make the connection to the intuitive description above: the action of writing an axiom on the blackboard corresponds to a *download*, the *erasure* rule is just the formalization of deleting information from the board, and using the the *inference* rule we can derive new partial results from the information we already have on the board. Let us next define the space measure of interest in this lecture, which we should think of as the maximum number of clauses stored in memory while carrying out the proof.

Definition 2.2 (Clause space). The *clause space* of a resolution derivation π is $Sp(\pi) = \max_{\mathbb{C} \in \pi}\{|\mathbb{C}|\}$, i.e., the maximal number of clauses in any configuration. The clause space of deriving D from F in resolution is

$$Sp_{\mathcal{R}}(F \vdash D) = \min_{\pi : F \vdash D}\{Sp(\pi)\}$$

and the clause space of refuting F is the space of deriving the empty clause $\bot$ from F, denoted $Sp_{\mathcal{R}}(F \vdash \bot)$.

We can also define the clause space of refuting F in tree-like resolution, denoted $Sp_{\mathcal{T}}(F \vdash \bot)$, where the minimum is taken over all tree-like resolution refutations.

There are several different ways of measuring space. For resolution, the most studied measure is clause space, and this lecture will focus almost exclusively on this measure. So when we say only "space" in what follows, we mean "clause space."

As we have mentioned before, the length of refuting a formula in general resolution is different from the length of refuting it in (the subsystem of) tree-like resolution. The situation is the same for space. We will focus on the space measure in general, unrestricted resolution, which is the most interesting case (but also the most challenging one). Thus, whenever we drop the subscript from the space measure, the measure is understood to be for general resolution.

Clause space and upper bounds from black pebbling Explain correspondence between resolution space and black pebbling price of proof DAG.

Proposition 2.3. *If a CNF formula F has a refutation in length L, then F can be refuted in clause space* $\mathrm{O}(L/\log L)$.

Is it possible to do better? Proposition 2.3 is the same space saving as for Turing machines in [HPV77]. This was recently improved from $\mathrm{O}(L/\log L)$ to something like $\mathrm{O}(\sqrt{L})$ by Ryan Williams. What about resolution?

Theorem 2.4 ([ET01]). *The space of refuting a CNF formula F is upper-bounded by the number of variables in F by $Sp(F \vdash \bot) \leq |Vars(F)| + 2$.*

Proof. Although we focus on the general case, tree-like resolution is sufficient to establish this upper bound.

In an earlier lecture, we sketched the correspondence between tree-like resolution refutations and decision trees and, in particular, argued that for any CNF formula F over n variables, there is a tree-like refutation π for which the proof DAG G_π is a binary tree of height $h \leq n$. For such a tree, one can prove that the corresponding resolution proof can be carried out in space $h + 2$.

The proof is by induction over the tree height, and the reader can have a look at the example in Figure 3 to see what is happening. The induction hypothesis is that any clause in the proof tree which is at height h above the axiom clauses can be derived in space $h + 2$.

- **Base case $h = 1$:** In this case, the subtree we are considering is simply a sort of triangle consisting of an internal vertex with two predecessor axiom clauses. The clause labelling the internal vertex can be derived in space $3 = h + 2$ from the predecessors by downloading the axioms and resolving. (For instance, $x \vee y$ can be derived in this way from $x \vee z$ and $y \vee \overline{z}$ at the far left bottom end of Figure 3).

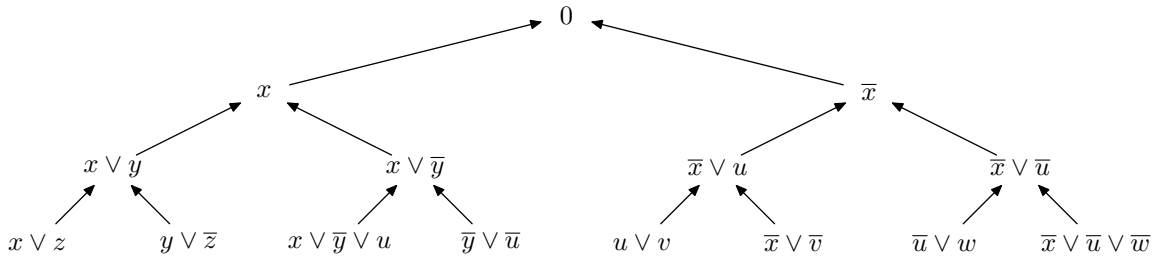


Figure 3: Example tree-like resolution refutation represented as a DAG.

- **Inductive step:** Suppose now that the clause at the root of any height-h subtree can be derived in space $h + 2$. Consider a vertex at height $h + 1$. According to the induction hypothesis, it takes clause space $h + 2$ to derive its left predecessor. Now erase everything except this clause, and then by induction derive also the right predecessor in space $h + 2$ while keeping the left predecessor in memory. Then erase everything except the two predecessors and resolve them to get the clause at height $h+1$. This requires clause space $(h+1)+2 = h+3$. The theorem follows by the induction principle. □

An important concept in proof complexity is that of *minimally unsatisfiable formulas* as defined next.

Definition 2.5. An unsatisfiable CNF formula F is *minimally unsatisfiable* if removing any clause from F makes it satisfiable.

Example 2.6. The formula

$$\begin{aligned} F = \quad & (x \vee z) \wedge (\overline{z} \vee y) \wedge (x \vee \overline{y} \vee u) \wedge (\overline{y} \vee \overline{u}) \\ & \wedge (u \vee v) \wedge (\overline{x} \vee \overline{v}) \wedge (\overline{u} \vee w) \wedge (\overline{x} \vee \overline{u} \vee \overline{w}) \end{aligned}$$

can be verified to be minimally unsatisfiable. Indeed, if we remove the first clause, we can satisfy the rest of the clauses with for instance the assignment $\{x = 0, y = 0, z = 0, u = 0, v = 1, w = 0\}$. If we remove the second clause then the assignment $\{x = 0, y = 0, z = 1, u = 0, v = 1, w = 0\}$ works instead. The rest of the cases are left to the reader as they are a bit tedious.

If we restrict F by setting x to true, we get

$$\begin{aligned} F\restriction_x = \quad & (\overline{z} \vee y) \wedge (\overline{y} \vee \overline{u}) \wedge (u \vee v) \\ & \wedge \overline{v} \wedge (\overline{u} \vee w) \wedge (\overline{u} \vee \overline{w}) \end{aligned}$$

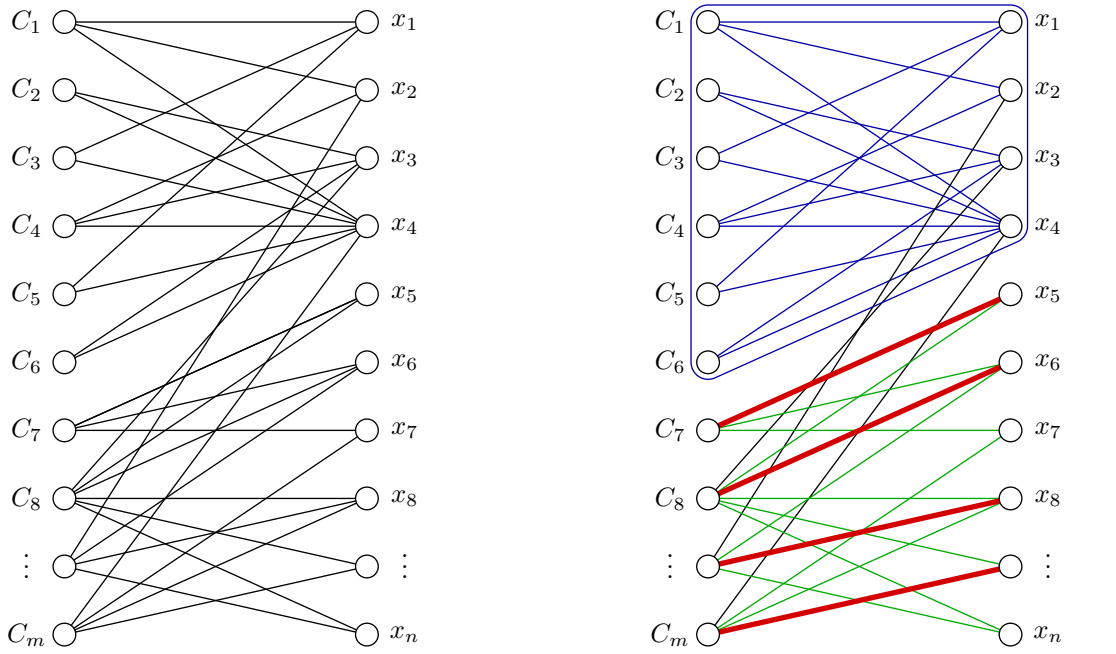
and this formula is *not* minimally unsatisfiable since we can remove the first two clauses and still have an unsatisfiable formula. The easy verification of this fact is left to the reader.

We remark that this example shows that although restrictions preserve other useful properties, for instance the property of being a resolution refutation with weakening (as we have seen before), they do *not* preserve minimal unsatisfiability.

Let us now study the properties of minimally unsatisfiable CNF formulas. We will see that their number of variables is upper-bounded by the number of clauses. In other words, if there are no more clauses than variables, then a formula *cannot* be minimally unsatisfiable. This result is almost folklore and has been (re)proven many times, but it is known as "Tarsi's lemma" from the paper [AL86] .

Theorem 2.7 (Tarsi's lemma). *Any minimally unsatisfiable CNF formula must have strictly more clauses than variables.*

In order to prove this theorem, we will argue in terms of matchings in bipartite graphs, and therefore Hall's theorem, used in the last lecture, will come in handy again. Let us recall what it says.



(a) Bipartite graph corresponding to formula F.

(b) Picture proof that F is not minimally unsatisfiable.

Figure 4: Illustration of argument in proof of Tarsi's lemma.

Theorem 2.8. *Let $G = (U \cup V, E)$ be a bipartite graph. Then there is a matching of U into V if and only if for all subsets $U' \subseteq U$ it holds that $\left|N(U')\right| \geq \left|U'\right|$.*

Proof of Lemma 2.7. We will prove it by contradiction. Suppose there exists a minimally unsatisfiable formula having more clauses than variables.

Let us introduce a bipartite graph G such that left vertices are labelled by clauses $C_i \in F$ and that right vertices are labelled by variables $x_j \in Vars(F)$. There is an edge between C_i and x_j if the variable x_j occurs in the clause C_i. For purposes of illustration, let us assume this graph looks like in Figure 4(a).

By the construction of G, it is easy to see that if there is a matching of U into V, then there is also satisfying assignment (but setting the variables so that they satisfy the clauses with which they are matched). Therefore, as we know that there is no such assignment, we deduce that there is no such matching. This is where Theorem 2.8 enters. As there is no matching, the theorem says that there exists some subset $S \subseteq F$ on the left with $|S| > |N(S)|$.

Fix such an S of maximal size. Suppose that this S is actually F. This would mean that $|F| > |Vars(F)|$ which is exactly what we want. Great! We cannot expect life to be quite this easy, however, and so we also have to consider the possibility that $S \neq F$. Or rather, we will now prove that we do *not* have to consider this possibility, since if $S \neq F$ then we get a contradiction. That clearly proves the theorem. So here goes.

Suppose that when we pick such a set S of maximal size we get a set that is not all of F. Again for purposes of illustration, suppose that S and $N(S)$ are as in the tetragon in Figure 4(b). The set of clauses S must be satisfiable since it is a strict subset of a minimally unsatisfiable formula. Fix some satisfying assignment to S, and observe that we only need to assign values to variables in $N(S)$ since no other variables occur in the clauses in S.

Now consider $F \setminus S$. For any subset $S' \subseteq F \setminus S$ it must hold that $\left|S'\right| \leq \left|N(S') \setminus N(S)\right|$. For if this was not so, we could add S' to S to get an even larger set with $|S \cup S'| > |N(S \cup S')|$. But then Hall's theorem strikes again and says that there is a matching of $F \setminus S$ into $N(S') \setminus N(S)$, as depicted in Figure 4(b). We can use this matching to assign values to the matched variables so that all clauses in $F \setminus S$ are satisfied.

The variables we just assigned are different from those assigned previously for S. As the two domains

are disjoint, we can join these two assignments into one larger assignment satisfying all clauses in F. But if so F is satisfiable, which is a contradiction. Hence it cannot be that $S \neq F$, so as claimed above we do not have to worry about this possibility. □

Tarsi's lemma can be used to show that refutation clause space is also upper-bounded by the number of clauses in a formula.

Theorem 2.9 ([ET01]). $Sp_{\mathcal{R}}(F \vdash \bot) \leq |F| + 1$.

Proof. If F is not minimally unsatisfiable, throw away clauses in some arbitrary order until we get a minimally unsatisfiable subformula $F' \subseteq F$. We know from Theorem 2.7 that $\left|F'\right| > \left|\mathit{Vars}(F')\right|$, and combining this with Theorem 2.4 we get that $\mathit{Sp}(F' \vdash \bot) \leq \left|\mathit{Vars}(F')\right| + 2 \leq \left|F'\right| + 1 \leq |F| + 1$. □

We have proven that

$$\mathit{Sp}(F \vdash \bot) \leq \min\left\{|F| + 1, |\mathit{Vars}(F)| + 2\right\} \tag{2.1}$$

and in particular that the size of a formula (which is at least the number of clauses and at least the number of variables) is an upper bound on the refutation clause space in resolution. This means that the space complexity of a formula F will always be somewhere between constant and linear, and the interesting questions then become:

- Which formulas require linear space? (Are there such formulas?)
- Which formulas can be refuted in, say, just logarithmic space?
- Or even in just constant space?

The next theorem says that the linear upper bound is indeed tight in the worst case, at least up to constant factors. We state it without proof for now.

Theorem 2.10 ([ABRW02, ET01]). *There is a polynomial-size family $\{F_n\}_{n=1}^{\infty}$ of unsatisfiable 3-CNF formulas such that $Sp_{\mathcal{R}}(F \vdash \bot) = \Omega(|F|) = \Omega(|\mathit{Vars}(F)|)$.*

Perhaps even more interesting than exactly what these and other lower bounds on clause space in resolution look like is the history behind how they appeared. The first results were reported independently in [ET01] and [ABRW02] (well, strictly speaking, in the conference versions [Tor99] and [ABRW00]). These papers proved optimal lower bounds on space for pigeonhole principle formulas, which turned out to match the bounds on width. In the same articles, a lower bound was also proven for Tseitin contradictions, which again turned out to also be the same as the width bound. These rather funny coincidences were explicitly pointed out and discussed in the papers. A little bit later,optimal space bounds for random k-CNF formulas were obtained in [BG03] (conference version [BG01]), and again these bounds coincided with the width bounds. Also, at this point there seemed to be a pattern in that the methods to prove space lower bounds were somehow quite similar in flavour to the techniques for proving width lower bounds. Funny indeed. And although there still was no formal link, by now it was an established conjecture that the width and space measures had to be connected in some way, and in particular that it seemed likely that width lower bounds should imply clause space lower bounds.

How do clause space lower bounds go? Consider a configuration-style resolution derivation $\pi = (\mathbb{C}_0 = \emptyset, \mathbb{C}_1, \ldots, \mathbb{C}_L)$ in not too large clause space. For every $\mathbb{C}_i$, construct inductively ρ_i such that

- $\rho_i(\mathbb{C}_i) = \top$,
- $|\mathrm{dom}(\rho_i)| \leq \mathit{Sp}(\mathbb{C}_i)$

Inductive step:

- Axiom download of C: Find free variable in $Vars(C) \setminus \mathrm{dom}(\rho_{i-1})$ to assign so as to satisfy C (uses expansion plus some planning ahead).

- Inference: Nothing needs to be done (by the soundness of the proof system).

- Erasure: Keep one literal per clause in $\mathbb{C}_i$ to satisfy it.

This conjecture turned out to be perfectly correct and was proven true by by Atserias and Dalmau [AD08] (conference version [AD03]). This paper not only solves this question, it also provides very simple and elegant proofs using tools originating in finite model theory. These proofs are what we want to study next.

3 Space and Width

What we want to do now is the following:

1. Give an exact characterization of resolution width in terms of a combinatorial game, where a good winning strategy corresponds to small refutation width.

2. Prove that a small-space refutation can be used to derive a good winning strategy.

Although this outline is (for now) rather vague, it should hopefully be clear that making and argument along these lines will allow us to prove lower bounds on space in terms of lower bounds on width (or upper bounds on width in terms of upper bounds on space, depending on which perspective we prefer). In particular, before we have finished today's lecture, Theorem 2.10 will have followed as a corollary from combining such an argument with lower bounds on width that we have proved previously.

Let us start by trying to give an intuitive description of what the combinatorial game is about. Of course, a formal definition will follow shortly.

The *existential pebble game* is a game between two (perhaps somewhat odd) persons, *Spoiler* and *Duplicator*.[1] These two persons are having a heated debate regarding a given unsatifiable CNF formula F and have different goals. Duplicator claims the formula is satisfiable, but Spoiler wants to prove him wrong (and by assumption Spoiler is right in wanting to do this).

In order for the debate not to spin out of control, strict rules have been set. Spoiler and Duplicator must speak during rounds having a very specific structure. First, Spoiler should pick a variable and ask Duplicator which value it should have. As Duplicator claims there is a satisfying assignment, he should know it and be able to answer. But since the formula is unsatisfiable this cannot really be the case, and if Spoiler just asks about enough variables then the contradiction should become apparent. However, things are not so easy for this poor Spoiler fellow as he is senile—he cannot keep more that p variable assignments in memory simultaneously (where p is a parameter of the game). Before Spoiler asks for the value of the $(p+1)$st variable, therefore, he will forget one of the previous answers. But interestingly, Spoiler is *selectively* senile in that he can make a conscious choice which of the previous answers to forget. All of this is completely transparent to Duplicator, who knows exactly what Spoiler remembers and what he has forgotten. In particular, if Spoiler ever asks again about a variable for which Duplicator already provided and answer but for which Spoiler forgot this answer, Duplicator is free to give a different answer each time which is inconsistent with what he has said previously.

If Duplicator ever answers in such a way that Spoiler has in memory a partial assignment falsifying a clause in F, Duplicator loses and Spoiler wins. The aim of Duplicator is thus to give Spoiler partial assignments that keep the formula from being falsified. Phrased differently, Duplicator wins if he can go on playing the game forever. If Duplicator can do this, a little bit of thought reveals that he must have a strategy that we will denote by $\mathcal{H}$ and that can be described as follows.

[1] These names are used for historical reasons.

Definition 3.1. Duplicator wins the *Boolean existential p-pebble game* over the CNF formula F if there is a nonempty family $\mathcal{H}$ of partial truth value assignments that do not falsify any clause in F and for which the following holds:

1. If $\alpha \in \mathcal{H}$ then $|\alpha| \leq p$ (where $|\alpha|$ denotes the number of variables in the current partial truth value assignment, which should be at most p).

2. If $\alpha \in \mathcal{H}$ and $\beta \subseteq \alpha$ then $\beta \in \mathcal{H}$. (When Spoiler forgets, Duplicator just keeps track of the partial assignment to the variables still in memory.)

3. If $\alpha \in \mathcal{H}$, $|\alpha| < p$ and $x \in Vars(F)$ then there exists a $\beta \in \mathcal{H}$ such that $\alpha \subseteq \beta$ and x is in the domain of β. (This tells Duplicator how to answer to a query about x.)

$\mathcal{H}$ is called a *winning strategy* for Duplicator. If there is no such strategy then Spoiler wins the game.

If there is a winning strategy for Duplicator, then it is not hard to see that there is a *deterministic* winning strategy that for each $\alpha \in \mathcal{H}$ and each move of Spoiler defines one unique move β for Duplicator. This in turn means that if Duplicator does *not* win, then there is a deterministic winning strategy for Spoiler as explained next.

Proposition 3.2. *If Duplicator has no winning strategy, then there is a deterministic winning strategy (in the form of a partial function from partial truth value assignments to variable queries/deletions) for Spoiler.*

Proof. The reason for this proposition to be true is that since the number of possible deterministic strategies for Duplicator is finite, then Spoiler can build a strategy by "brute force" evaluating all possible responses to sequences of queries and deletions. (Note that we are not at all interested in questions about efficiency here, we just care about the existence of deterministic strategies.)

Arguing somewhat more carefully, we can consider Duplicator's strategy to be simply a large table. If the partial assignment in Spoiler's memory is α and variable x is being queried, then Duplicator looks at the corresponding row in the table to see if he should answer 0 or 1. If F has n variables, then the number of subsets of variables Spoiler can have in memory is $\sum_{i=0}^{p} \binom{n}{i}$ and since each subset of variables of size m can be assigned in 2^m ways, we get a total of at most $\sum_{i=0}^{p} 2^i \binom{n}{i}$ partial assignments. The total number of rows needed in Duplicator's strategy table, therefore, is at most $R = n \sum_{i=0}^{p} 2^i \binom{n}{i}$. Each row contains 0 or 1, so if we write $\mathcal{S}$ to denote the set of all possible strategies for a deterministic Duplicator in the p-pebble game, we get that $|\mathcal{S}| \leq 2^R < \infty$ is certainly finite.

The assumption is that none of these strategies is winning. Therefore, shifting to Spoiler's perspective, for each strategy $S \in \mathcal{S}$, each $\alpha \in S$, and each x not assigned by α, there is a shortest sequence of queries and deletions beginning with x that leads to a $\beta \in S$ that cannot be extended to any variable y without falsifying some $C \in F$. Denote this length $l(S, \alpha, x)$. Start with the empty assignment $\emptyset$ and let Spoiler choose x minimizing $\max_{S \in S}\{l(S, \emptyset, x)\}$. Record the possible answers $\alpha_1 = \{x = 0\}, \alpha_2 = \{x = 1\}$. Note that for both α_1 and α_2 we now have shorter sequences $\min_{y \neq x}\{\max_{S \in \mathcal{S}}\{l(S, \alpha_i, y)\}\} < \max_{S \in \mathcal{S}}\{l(S, \emptyset, x)\}$ by assumption. Inductively, suppose we need to fix Spoiler's response to α. Look at all possible moves and pick one minimizing the maximum distance to a contradiction. Since these sequences keep getting shorter, this construction takes a finite number of steps and yields a winning strategy. □

Somewhat amazingly, this existential p-pebble game *exactly characterizes* resolution width, as stated formally in the following theorem.

Theorem 3.3 ([AD08]). *Let F be an unsatisfiable CNF formula and let $p > W(F)$. Then F has a resolution refutation $\pi : F \vdash \bot$ of width $W(\pi) \leq p$ if and only if Spoiler wins the existential $(p + 1)$-pebble game on F.*

Proof. Let us do the directions of the if and only if statement on at a time.

($\Rightarrow$) First, let us prove that a narrow resolution refutation yields a winning strategy for Spoiler. Consider the DAG G_π associated to the resolution proof $\pi : F \vdash \bot$. Spoiler starts at the vertex for $\bot$ and inductively queries the variable resolved upon to to get there. Spoiler then moves to the assumption clause D falsified by Duplicator's answer and forgets all variables not in D. This is inductively repeated for each new clause. As Spoiler walks through the DAG, he maintains the invariant that he is always standing on a clause falsified by the variables he has kept in memory. Spoiler will eventually reach a falsified axiom having remembered no more than $W(\pi) + 1$ variables simultaneously. The reason we have $W(\pi)$ in the bound is that this is the largest size of any clause $D \in \pi$, and the extra $+1$ is needed for the variable resolved over to derive D.

($\Leftarrow$) Let us now prove that the existence of a winning strategy for Spoiler yields a narrow proof. We will build a DAG G_π corresponding to a proof π of the formula using Spoiler's strategy. Here is how.

Start with the $\bot$ vertex. Supposing that x is the first variable queried, make vertices labelled by the clauses x and $\overline{x}$ with edges to $\bot$. Inductively, let ρ_v be the unique minimal partial truth value assignment falsifying the clause D_v labelling v. If the move on ρ_v is the deletion of y, make new vertex $D_v \setminus \{y, \overline{y}\}$ with an edge to D_v, corresponding to a weakening step. Otherwise, if y is queried, make new vertices $D \vee y$ and $D \vee \overline{y}$ with edges to D, corresponding to a resolution step.

Since Spoiler will win in a finite number of steps regardless of how Duplicator answers, the graph G_π is finite. Moreover, it is easy to see that locally all steps in G_π are legal resolution derivation steps. The question is from which initial clauses this derivation starts, i.e., which clauses are labelling source vertices in G_π. By construction the source vertices of the DAG are winning positions for Spoiler, and since Spoiler wins when the current assignment falsifies an axiom clauses all sources will be labelled by (weakenings of) axioms of F. Therefore, G_π is the DAG of a correct resolution refutation $\pi : F \vdash \bot$, possibly using weakening. If we go over π in a postprocessing step and remove all weakening steps, we get a derivation in width at most p since if $|\rho_v| = p + 1$ then the next move for Spoiler must be a deletion. □

The lower bound on space in terms of width that we will prove next follows from the fact that Spoiler can use *proofs in small space* to construct winning strategies with few pebbles.

Lemma 3.4 ([AD08]). *Let F be an unsatisfiable CNF formula of width $W(F) = w$ with refutation space $Sp(F \vdash \bot) = s$. Then Spoiler wins the existential $(s + w - 2)$-pebble game on F.*

Proof. Take a resolution refutation $\pi = \big(\mathbb{C}_0 = \emptyset, \mathbb{C}_1, \ldots, \mathbb{C}_\tau = \{\bot\}\big)$ of F in clause space s. Spoiler can construct a winning strategy by inductively defining a partial truth value assignment ρ_t such that ρ_t satisfies $\mathbb{C}_t$ by setting (at most) one literal per clause to true.

Suppose inductively that we have constructed such assignments up to time t and consider the three possible derivation steps to get from $\mathbb{C}_t$ to $\mathbb{C}_{t+1}$.

Suppose first that π does an *axiom download* at time t. We observe that we can assume that axiom downloads occur only for configurations $\mathbb{C}_t$ of size $|\mathbb{C}_t| \leq s - 2$. This is without loss of generality since at the time of a download, one also needs one free memory slot for the resolvent. Otherwise, the next step would have to be an erasure and we could reverse the order of these two derivation steps. When π downloads $C \in F$, Spoiler queries Duplicator about all variables in C until Duplicators gives an answer that satisfies C (which he must to avoid losing). Once Spoiler obtains an answer satisfying a literal a in C, he forgets the rest of the variables queried for C and keeps this one satisfying literal to get $\rho_{t+1} = \rho_t \cup \{a\}$. This sequence of moves in the pebble game uses at most $(s - 2) + w$ pebbles. To see this, note that ρ_t has size atmost $|\mathbb{C}_t| \leq s - 2$ as explained above, and at most $W(F) = w$ extra pebbles are needed to find a literal satisfying C.

When a clause is *erased*, Spoiler deletes the corresponding literal satisfying the clause from ρ_t if necessary (i.e if $|\rho_t| = |\mathbb{C}_t|$).

At last, for *inference* steps, Spoiler sets $\rho_t = \rho_{t-1}$ since, by induction, ρ_{t-1} must satisfy the resolvent. Indeed, suppose a clause $D \vee D' \in \mathbb{C}_{t+1}$ is derived by resolving two clauses $D \vee x, D' \vee \overline{x} \in \mathbb{C}_t$ over x. Then $\rho_t(D \vee x) = \rho_t(D \vee \overline{x}) = 1$ by our inductive hypothesis, and at least of of these clauses is satisfied

by some literal over a variable other than x. This same literal appears in $D \vee D'$, and so ρ_t must fix this resolvent to true.

As we can see, ρ_τ cannot satisfy $\mathbb{C}_\tau$ since it contains $\bot$, so whatever Duplicator does he must fail to satisfy an axiom downloaded at some time prior to time τ. Therefore, Spoiler has a winning strategy with a number of pebbles smaller than or equal to $(s-2)+w$. □

Theorem 3.5 ([AD08]). *For any unsatisfiable CNF formula F it holds that*

$$Sp(F \vdash \bot) - 3 \geq W(F \vdash \bot) - W(F)\,.$$

Proof. This theorem is a direct consequence of Theorem 3.3 and Lemma 3.4. Indeed, from the theorem we know that if Spoiler wins the existential $(p+1)$-pebble game on F, then $W(F \vdash \bot) \leq p$. The lemma then says that if $W(F) = w$ and $Sp(F \vdash \bot) = s$, then Spoiler wins the existential $(s+w-2)$-pebble game on F. □

Remark 3.6. One way of rephrasing what Theorem 3.5 means in words is as follows. Any resolution refutation of a nontrivial formula needs to do at least one resolution step, and that requires clause space 3 (to keep the two clauses resolved as well as the resolvent in memory). Thus, the left-hand side $Sp(F \vdash \bot) - 3$ of the inequality in Theorem 3.5 measures the amount of space needed over the bare minimum necessary. Similarly, if F is a minimally unsatisfiable formula it is clear that any resolution refutation needs to have width $W(F)$ since all clauses are needed to refute the formula. Hence, the right-hand side of the inequality $W(F \vdash \bot) - W(F)$ is the width needed on top of the obvious minimal requirement. What Theorem 3.5 says is that the minimum extra space needed is lower-bounded not asymptotically, but *exactly*, by the minimum extra width needed.

Using this theorem, we can rederive all optimal (i.e, linear) lower bounds on space we know. To do so, all we have to do is to use lower bounds on width from, e.g., [BW01] and then the lower bound on space in terms of width in Theorem 3.5. In particular, in this way we can obtain Theorem 2.10 (for instance, by using Tseitin contradictions).

In view of that we have just proven that width is a lower bound on clause space, and that as we saw above these two measures seem to coincide for a number of formula families, it is very natural to ask whether this is always the case. That is, more formally we have the following question:

> *Is it true for an unsatisfiable k-CNF formula that the width of refuting it and the clause space of refuting it always coincide asymptotically?*

We will return to this question, and answer it, later during the course.

An interesting corollary of Theorem 3.5 is that if a k-CNF formula is easy with respect to clause space, it is also easy with respect to length.

Corollary 3.7. *If a k-CNF formula F over n variables is refutable in constant space, then F is also refutable in length polynomial in n.*

Proof. The proof of this corollary is very direct. A constant upper bound on clause space implies a constant upper bound on width. As the number of clauses of constant width is polynomial, we obtain an upper bound on the length by simply counting. □

Space is still not a very well understood complexity measure, not even for such a (seemingly) simple proof system and resolution, and there are a number of easy to state but (apparently) hard to resolve open problems regarding space. Let us very quickly give a few examples.

The first question is whether Corollary 3.7 can be strengthened by relaxing the necessary upper bound on clause space to logarithmiv.

Open Problem 1. *Is it true that a* logarithmic *upper bound on clause space implies a polynomial upper bound on length?*

The second question is whether something can be said about the length of a refutation in constant clause space.

Open Problem 2. *Is it true that a resolution refutation in constant space can also be carried out in polynomial length, or can at least be modified to a refutation in simultaneous polynomial length and constant space?*

We remark that Corollary 3.7 does not give any information regarding this question. When we take a small-space refutation and run it through Theorem 3.5, what pops out is a potentially completely different refutation, and we do not know anything about this refutation except that it is narrow, and therefore also has to be short. It would be interesting to get a more constructive proof of Corollary 3.7 that can provide some understanding as to why constant space implies polynomial length.

Open Problem 3. *Is there a direct proof showing that constant clause space implies polynomial length, and* explaining *why this is true?*

In a sense, the problem is that the upper bound on width in terms of space by Atserias and Dalmau is fairly non-constructive. It just says that small space implies the existence of a narrow-width refutation, but does not tell us anything about how to *construct* such a narrow refutation if we are given a small-space refutation. Hence the final open question that we want to mention in this context.

Open Problem 4. *Is there a constructive, explicit version of Theorem 3.5 that can tell us how to convert space-efficient refutations to narrow refutations?*

A question which is *not* open however is the following:

Is it true that the space-efficient refutation can itself also be narrow?

The answer to this question is "no." Interestingly, this was proven in [Ben02] even before the paper [AD03] appeared. We turn to this topic next.

4 Pebbling Contradictions

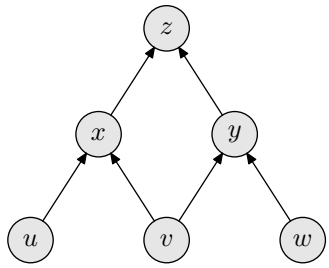
We want to find CNF formulas exhibiting a trade-off between clause space and width in the sense that they are easy for both width and space, but that one cannot minimize both of these measures simultaneously. It turns out that such formulas can be constructed by using *pebble games*, but these games are not the existential pebble games discussed above but instead the pebble games discussed at the beginning of this lecture.

So what does this game have to do with proof complexity in resolution? Not a lot, it might seem, but it turns out that one can define CNF formulas encoding specific instances of pebble games on graphs, and such formulas have interesting properties.

In the last couple of decades, there has been renewed interest in pebbling in the context of proof complexity. The way pebbling results have been used in proof complexity has mainly been by studying so-called *pebbling contradictions* (also known as *pebbling tautologies*[2] or *pebbling formulas*). These are CNF formulas encoding the pebble game played on a DAG G by postulating the sources to be true and the sink to be false, and specifying that truth propagates through the graph according to the pebbling rules. The idea to use such formulas seems to have appeared for the first time in Kozen [Koz77], and they were also studied in [RM99, BEGJ00] before being defined in full generality by Ben-Sasson and Wigderson in [BW01].

Definition 4.1 (Pebbling contradiction). Suppose that G is a DAG with sources S and a unique sink z. Identify every vertex $v \in V(G)$ with a propositional logic variable v. The *pebbling contradiction* over G, denoted Peb_G, is the conjunction of the following clauses:

[2]Recall that we warned already during the first lecture for this sometimes confusing convention in proof complexity to consider contradictory CNF formulas to be tautologies (since they are used to encode negations of tautological statements).



(a) Pyramid graph Π_2 of height 2.

$$
\begin{aligned}
& u \\
\wedge\ & v \\
\wedge\ & w \\
\wedge\ & (\overline{u} \vee \overline{v} \vee x) \\
\wedge\ & (\overline{v} \vee \overline{w} \vee y) \\
\wedge\ & (\overline{x} \vee \overline{y} \vee z) \\
\wedge\ & \overline{z}
\end{aligned}
$$

(b) Pebbling contradiction Peb_{Π_2}.

Figure 5: Pebbling contradiction for the pyramid graph Π_2.

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

If G has n vertices and maximal indegree ℓ, the formula Peb_G is a minimally unsatisfiable $(1+\ell)$-CNF formula with $n+1$ clauses over n variables. We will almost exclusively be interested in dags with bounded indegree $\ell = O(1)$, usually $\ell = 2$. We note that DAGs with fan-in 2 and a single sink have sometimes been referred to as *circuits* in the proof complexity literature, although we will not use that terminology in this course. For an example of a pebbling contradiction, see the CNF formula in Figure 5(b) defined in terms of the graph in Figure 5(a).

Clearly, such formulas have to be unsatisfiable since all sources are postulated to be true and truth propagates from predecessors to successors but the sink is required to be false.

Though this might not a priori be obvious at all, there are close relations between resolution refutations of pebbling contradictions and pebblings of graphs. It is not so hard to see that a black pebbling of G can be turned into a resolution refutation of Peb_G in length and space corresponding to the pebbling time and space. What is more interesting, though, is the result in the other direction that any resolution refutation of Peb_G can be translated into black-white pebbling of G in the same way.

5 Reductions Between Pebbling and Resolution

The formal statement of how any black pebbling of a graph can be simulated by a resolution refutation of the corresponding pebbling contradiction is as follows.

Observation 5.1 ([BIW04]). *If G is a DAG with constant fan-in and a single sink, then from any complete black-only pebbling $\mathcal{P}$ of G one can extract a resolution refutation $\pi : Peb_G \vdash \bot$ of the pebbling contradiction over G such that $L(\pi) = O(\mathsf{time}(\mathcal{P}))$, $W(\pi) = O(1)$, and $TotSp(\pi) = O(\mathsf{space}(\mathcal{P}))$.*

Proof of Observation 5.1. Suppose that G is a DAG with indegree ℓ and unique sink z. Let $n = |V(G)|$. Sort the vertices of G in topological order. We will show how to derive the unit clause v for all vertices $v \in V(G)$ in this topological order in length $n(1+\ell)$ and width $1+\ell$. Once we derive z, we then resolve with $\overline{z}$ to get the empty clause. Since ℓ is assumed to be constant, this is sufficient to prove the observation.

The argument is by induction. If s is a source vertex, then the unit clause s is an axiom of Peb_G and there is nothing to prove. If u is a non-source, say with immediate predecessors $pred(u) = \{v_1, \ldots, v_{\ell'}\}$ for $\ell' \leq \ell$, then by induction we have already derived all the unit clauses v_i for $i = 1, \ldots, \ell'$. Download the axiom $\overline{v}_1 \vee \cdots \vee \overline{v}_{\ell'} \vee u$ and resolve in ℓ' steps with v_i, $i = 1, \ldots, \ell'$, to derive u. This is a total of at most $1+\ell$ steps per vertex, and the widest clauses that appear in the refutation are the axiom clauses. □

The other direction is much less obvious, but essentially says that any resolution refutation of a pebbling contradiction can be simulated by a black-white pebbling of the underlying graph.

Theorem 5.2 ([Ben09]). *If G is a DAG with constant fan-in and a single sink, then from any resolution refutation $\pi : Peb_G \vdash \bot$ of the pebbling contradiction over G one can extract a complete black-white pebbling $\mathcal{P}_\pi$ of G such that* $\mathsf{time}(\mathcal{P}_\pi) = \mathrm{O}(L(\pi))$ *and* $\mathsf{space}(\mathcal{P}_\pi) = \mathrm{O}(VarSp(\pi))$. *In more detail, the pebbling time will be at most linear in the number of axiom downloads in π.*

Proof sketch. This might also become part of a problem set, but let us drop some heavy hints.

The general idea is to let positive literals correspond to black pebbles and negative literals to white pebbles. Using this correspondence, one gets a pebble configuration from every clause configuration. Then one needs to glue these pebble configurations together in such a way that one obtains a correct pebbling for which the bound stated above holds.

There are two helpful technical observations that one needs to prove. Namely, without loss of generality one can make the following assumptions:

- Any clause appearing in any configuration of the proof (other than the final empty clause) is resolved over at least once before being erased.
- When a clause is erased from a configuration after having been used in a resolution inference for the last time, it is erased immediately after this final resolution inference step.

The rest is just figuring out how to put the pieces together. Note that the hard part is to show that you really get a correct pebbling strategy for the graph, and this needs to be argued carefully. □

6 A Trade-off Between Width and Clause Space in Resolution

When we showed that clause space is an upper bound on width in resolution, we observed that the proof of this fact it [AD08] worked by taking a space-efficient resolution refutation π and and transforming it into a potentially completely different narrow resolution refutation π' (namely, by first using π to derive a good strategy for the Spoiler in the combinatorial game characterizing width, and then using the Spoiler strategy to obtain a refutation π' in small width). It is natural to ask whether such a transformation is really necessary. Maybe one can prove that a space-efficient refutation is also narrow simply by virtue of having small space? Or maybe a refutation in small space can at least be massaged into one that has small width also without blowing up the space?

It turns out that this question had been answered even before we knew there was this reason to ask it, namely in the conference paper [Ben02] preceding [AD03]. (These papers later appeared as journal versions [AD08] and [Ben09] in reverse chronological order.) The answer is that in general it is impossible to optimize space and width simultaneously in any meaningful way.

Theorem 6.1 ([Ben09]). *There is a family of k-CNF formulas F_n of size $\Theta(n)$ such that $Sp(F_n \vdash \bot) = \mathrm{O}(1)$ and $W(F_n \vdash \bot) = \mathrm{O}(1)$, but for any resolution refutation $\pi_n : F_n \vdash \bot$ it holds that $Sp(\pi) \cdot W(\pi) = \Omega(n/\log n)$.*

What this theorem says is that although the formulas F_n can be refuted in essentially minimal clause space and essentially minimal width, when we optimize one of these measures the other has to blow up to almost worst possible. (Recall that the worst-case upper bound is linear in n, and the lower bound we get here is linear except for a $\log n$ factor.)

We will spend the rest of this section proving this theorem. Let us start by an easy observation.

Observation 6.2. *For any resolution refutation π it holds that $Sp(\pi) \cdot W(\pi) \geq TotSp(\pi)$.*

Proof. The refutation π never has more that $Sp(\pi)$ clauses in memory, and each clause has size at most $W(\pi)$. Thus the total number of literals in memory at any point during π, i.e., the total space, is at most $Sp(\pi) \cdot W(\pi)$. □

Now we can present the formulas we want to use to prove Theorem 6.1. Namely, we take the graphs G_n from Theorem 1.4, which are very hard with respect to pebbling space, and look at pebbling contradictions over these graphs. Using Theorem 5.2, we can conclude that for these formulas it must hold that $TotSp(Peb_{G_n} \vdash \bot) \geq VarSp(Peb_{G_n} \vdash \bot) = \Omega(n/\log n)$.

All that remains now is to prove that these formulas are easy with respect to both width and clause space. But this is in fact the case for any pebbling contradiction. We already observed this for width in Observation 5.1, and the formal statement for clause space follows below.

Lemma 6.3 ([Ben09]). *For any DAG G of size $\Theta(n)$ there is a refutation $\pi : Peb_G \vdash \bot$ with $L(\pi) = \mathrm{O}(n)$ and $Sp(\pi) = \mathrm{O}(1)$.*

Before proving Lemma 6.3, let us just note that if we can do so then we are done also with Theorem 6.1. We already argued above that $TotSp(Peb_{G_n} \vdash \bot) \geq VarSp(Peb_{G_n} \vdash \bot) = \Omega(n/\log n)$. This means that although we can refute the formulas Peb_{G_n} in width $W(Peb_{G_n} \vdash \bot) = \mathrm{O}(1)$ by Observation 5.1 and also in clause space $Sp(Peb_{G_n} \vdash \bot) = \mathrm{O}(1)$ by Lemma 6.3, any refutation π optimizing one of the measures must have the other measure being large.

Proof of Lemma 6.3. Suppose that G is a DAG with unique sink z and number of vertices $n = |V(G)|$. Sort the vertices of G in *reverse* topological order. We will consider the vertices in this order $v_1 = z, v_2, \ldots, v_{n-1}, v_n$ and derive in constant clause space (in fact, minimal clause space 3) for each v_i a clause $D_i = \bigvee_{w \in W_i} \overline{w}$ such that $W_i \subseteq \{v_{i+1}, \ldots, v_n\}$. When we reach v_n, this means we have derived the empty clause.

Start by downloading the sink axiom $\overline{z}$ as well as the pebbling axiom $\bigvee_{w \in pred(z)} \overline{w} \vee z$ for the sink, and resolve these two clauses to get $D_1 = \bigvee_{w \in pred(z)} \overline{w}$. This clause clearly satisfies the invariant specified above for $W_1 = pred(z)$.

When we get to the vertex v_i, the clause $D_{i-1} = \bigvee_{w \in W_{i-1}} \overline{w}$ must contain the literal $\overline{v}_i$ since v_i is the predecessor of some $v_{i'}$ for $i' < i$, and the literal $\overline{v}_i$ was added to the clause when the pebbling axiom for $v_{i'}$ was downloaded. Now download the pebbling axiom $\bigvee_{w \in pred(z)} \overline{w} \vee v_i$ for v_i and resolve with D_{i-1} to get a new clause $D_i = \bigvee_{w \in W_i} \overline{w}$ for $W_i = (W_{i-1} \cup pred(v_i)) \setminus \{v_i\}$, and then erase everything except D_i from memory. Since the resolution step removed the literal $\overline{v_i}$ and all new literals added must be for vertices before v_i in the topological ordering, the invariant is maintained. Since the derivation works by always keeping one clause in memory and downloading axioms and resolving with this clause, the space of the derivation is 3. We make exactly one resolution inference (plus two erasures) per vertex, so the length of the derivation is linear in the number of vertices (independent of the fan-in of the graph). The lemma now follows by the induction principle. □

References

[ABRW00] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC '00)*, pages 358–367, May 2000.

[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, April 2002. Preliminary version in *STOC '00*.

[AD03] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. In *Proceedings of the 18th IEEE Annual Conference on Computational Complexity (CCC '03)*, pages 239–247, July 2003.

[AD08] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, May 2008. Preliminary version in *CCC '03*.

[AL86] Ron Aharoni and Nathan Linial. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *Journal of Combinatorial Theory Series A*, 43(2):196–204, 1986.

[BEGJ00] María Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM Journal on Computing*, 30(5):1462–1484, 2000. Preliminary version in *FOCS '98*.

[Ben02] Eli Ben-Sasson. Size space tradeoffs for resolution. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pages 457–464, May 2002.

[Ben09] Eli Ben-Sasson. Size-space tradeoffs for resolution. *SIAM Journal on Computing*, 38(6):2511–2525, May 2009. Preliminary version in *STOC '02*.

[BG01] Eli Ben-Sasson and Nicola Galesi. Space complexity of random formulae in resolution. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity (CCC '01)*, pages 42–51, June 2001.

[BG03] Eli Ben-Sasson and Nicola Galesi. Space complexity of random formulae in resolution. *Random Structures and Algorithms*, 23(1):92–109, August 2003. Preliminary version in *CCC '01*.

[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.

[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*.

[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*.

[ET99] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. In *Proceedings of the 16th International Symposium on Theoretical Aspects of Computer Science (STACS '99)*, volume 1563 of *Lecture Notes in Computer Science*, pages 551–560. Springer, March 1999.

[ET01] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001. Based on the conference papers in *STACS '99* [ET99] and *CSL '99* [Tor99].

[GT78] John R. Gilbert and Robert Endre Tarjan. Variations of a pebble game on graphs. Technical Report STAN-CS-78-661, Stanford University, 1978. Available at http://infolab.stanford.edu/TR/CS-TR-78-661.html.

[HPV77] John Hopcroft, Wolfgang Paul, and Leslie Valiant. On time versus space. *Journal of the ACM*, 24(2):332–337, April 1977. Preliminary version in *FOCS '75*.

[Koz77] Dexter Kozen. Lower bounds for natural proof systems. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science (FOCS '77)*, pages 254–266, 1977.

[Nor25] Jakob Nordström. New wine into old wineskins: A survey of some pebbling classics with supplemental results. Manuscript in preparation. Current draft version available at https://jakobnordstrom.se/publications/, 2025.

[Pip80] Nicholas Pippenger. Pebbling. Technical Report RC8258, IBM Watson Research Center, 1980. In *Proceedings of the 5th IBM Symposium on Mathematical Foundations of Computer Science*.

[PTC77] Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977.

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

[Sav98] John E. Savage. *Models of Computation: Exploring the Power of Computing*. Addison-Wesley, 1998. Available at http://www.modelsofcomputation.org.

[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, September 1999.