DD3501 Current Research in Proof Complexity Nov 3, 2011

Lecture 4

Lecturer: Jakob Nordström *Scribe: Léo Perrin*

The agenda for today's lecture is to start by proving a stronger exponential lower bound on resolution refutation length than in Lecture 3, which we do in Section 1, again using the connection between length and width proven in previous lectures. Then we switch gears a bit and discuss space in resolution in Section 2. It turns out that here, too, the width measure is a very useful tool, as we will see in Section 3 where we prove that space is always an upper bound on length. As part of our continued study of space and the relation between space and width, we conclude in Section 4 by taking a very brief look at so-called pebble games played on DAGs and CNF formulas encoding instances of these games.

1 A Stronger Exponential Lower Bound on Refutation Length

Last time, we managed to prove a lower bound on the length of resolution refutations of CNF formulas encoding the pigeonhole principle. In terms of the size N of the formula, we obtained an $\exp\big(\Omega\big(\sqrt[3]{N}\big)\big)$ bound. The aim today is to prove a stronger lower bound for another family of formulas, defined in terms of graphs, for which any resolution refutations must have length $\exp(\Omega(N))$. This is optimal up to constant factors in the exponent, since any unsatisfiable CNF formula of size N can be refuted in resolution in length $\exp(\mathrm{O}(N))$. We refer to such bounds as being *truly exponential* to distinguish them from bounds $\exp\big(\Omega\big(N^{\delta}\big)\big)$ for $0 < \delta < 1$ which we call just exponential.

We will start by recalling what these formulas look like, as we discussed at the very end of last lecture, and then discuss what graphs are good starting points for constructing such formulas (although we will not exhibit such graphs but just claim that they can be shown to exist, similarly to what we did in last lecture). Then, in a second step we will show that we can obtain from such good graphs unsatisfiable CNF formulas requiring refutations of truly exponential length.

1.1 Tseitin Contradictions and Edge Expander Graphs

Recall from last time that we started with a graph G and let the variables in our formula correspond to edges in this graph. Throughout this lecture, G is assumed to be undirected and connected, and if we want to be really precise we should also specify that G has no self-loops, i.e., that there are no edges (v, v). We also fix some function $f : V(G) \to \{0, 1\}$ giving zero-one labels to all vertices in G. We want the boolean variables corresponding to the edges incident to a vertex v to have truth values such that $f(v) = \bigoplus_{e \ni v} x_e$. In other words, the logical exclusive or (XOR) of the edges must have the same value as $f(v)$. Therefore, f can be viewed as a parity constraint for each vertex, and we can define a set of CNF clauses $PARITY_v$ encoding this constraint as explain in the last lecture.

In this way, for any G and for any f we can define the *Tseitin formula* for G and f as follows.

Definition 1.1 (Tseitin formula). The *Tseitin formula* $Ts(G, f)$ with respect to an undirected graph G and a function $f : V(G) \to \{0, 1\}$ is

$$Ts(G, f) = \bigwedge_{v \in V(G)} PARITY_v \ .$$

Note that so far we have not said anything about satisfiability or unsatisfiability, so $Ts(G, f)$ is just a formula Let us write down some basic properties of this formula.

Proposition 1.2. *If G is a graph with $|V(G)| = n$ and maximum degree d, then $Ts(G, f)$ is a d-CNF formula with at most $\frac{nd}{2}$ variables and at most $n2^{d-1}$ clauses. In particular, if d is constant, then $Ts(G, f)$ has $\Theta(n)$ clauses and variables.*

Proof. Summing over all vertices $v \in V(G)$ and all edges incident to v we get at most nd edges. Each edge $e = (v, w)$ is counted twice (once for v and once for w) so the total number of edges in G, which is also the total number of variables in $Ts(G, f)$, is at most $\frac{nd}{2}$.

We have n constraints $PARITY_v$, and each contraint encodes an XOR or negated XOR (when $f(v) = 1$ or $f(v) = 0$, respectively). As we saw in an example last lecture, an XOR or negated XOR over d variables can be written as a conjunction of 2^{d-1} clauses, each clause ruling out one of the 2^{d-1} assignments to the variables that violate the constraint. Each such clause speaks about at most d edges and so has size at most d. Therefore, $Ts(G, f)$ consists of d-clauses and there are at most $n2^{d-1}$ of them. □

In order to get unsatisfiable formulas, we will focus on functions f such that the total number of 1-labelled vertices is odd.

Definition 1.3. We say that $f : V(G) \to \{0, 1\}$ has *odd weight* if $\sum_{v \in V(G)} f(v) \equiv 1 \pmod 2$.

Lemma 1.4. *If G is a connected graph, then $Ts(G, f)$ is unsatisfiable if and only if f has odd weight.*

Proof sketch. One direction of this equivalence is fairly easy to argue. Let G be a graph. Suppose f has odd weight and consider a supposedly satisfying assignment α of $Ts(G, f)$. Since all parity constraints are satisfied, we have $\bigoplus_{e \ni} \alpha(x_e) \equiv f(v) \pmod 2$ for every vertex v. Summing over the whole graph we get

$$\sum_{v \in V(G)} \bigoplus_{e \ni v} \alpha(x_e) \equiv \sum_{v \in V(G)} f(v) \equiv 1 \pmod 2 \tag{1.1}$$

since f has odd weight. On the other hand, as we observed in the proof of Proposition 1.2, when we sum over all vertices in the graph all edges are counted exactly twice, and therefore the sum

$$\sum_{v \in V(G)} \bigoplus_{e \ni v} \alpha(x_e) \equiv 2 \sum_{e \in E(G)} \alpha(x_e) \equiv 0 \pmod 2 \tag{1.2}$$

clearly must be even. Thus there cannot exist any satisfying assignment α.

The other direction requires more complex arguments which will not be detailed here. □

From now on, we will only consider odd-weight functions f, and we will refer to formulas corresponding to such functions as *Tseitin contradictions*.

In order to get hard instances for resolution, we want to define formulas $Ts(G, f)$ for "the right kind" of graphs G. As in the previous lecture, it turns out that *expander graphs* will be useful, although here we will use a slightly different concept of *edge expansion*.

Definition 1.5 (Edge expansion). Let G be an undirected graph and for any subset $S \subseteq V(G)$ of vertices let $E(S, \overline{S})$ be the set of edges connecting S to its complement $\overline{S} = V(G) \setminus S$. Then the *edge expansion* of G, also known as the *isoperimetric number*, is

$$h(G) = \min \left\{ \frac{|E(S, \overline{S})|}{|S|} : S \subset V(G), |S| \leq \left|\overline{S}\right| \right\} .$$

Intuitively, if $h(G)$ is large then there are lots of edges leaving any subset of vertices of G.

Claim 1.6. There are graphs G_n of size $n \to \infty$ of maximal degree 3 and with edge expansion $h(G_n) \geq h^*$ for some absolute constant $h^* > 0$.

When such an inequality holds, we also say that $h(G_n)$ is *bounded away from zero*.

1.2 A Lower Bound on Refutation Length of Tseitin Contradictions

Now that we have all the tools we need at our disposal, let us prove an exponential lower bound. Just as in last lecture, we will show a lower bound on width, and then turn this lower bound on width into a lower bound on length using the result from [BW01].

Lemma 1.7. *If G is an undirected graph with n vertices and f is an odd-weight function, then* $W_{\mathcal{R}}(Ts(G,f) \vdash \bot) \geq h(G) \cdot \frac{n}{3}$.

We will spend the rest of this section proving this lemma. To do so, we again define a measure $\mu : \{\text{clauses}\} \to \mathbb{N}$ of "progress" for resolution refutations of Tseitin contradictions. This μ is different from, but very similar to, the measure used last time, and it should have the same general properties. Namely:

1. $\mu(\text{axiom clause}) \leq 1$, i.e., the measure of any axiom is small (since the axiom clauses are given, just listing such a clause is not making much progress).
2. $\mu(\bot)$ is large (having finished the refutation is a lot of progress).
3. μ only increases gradually, in particular, it can never more than double in a single derivation step (making progress requires hard work).
4. The gradual increase implies that there is a "medium-progress" clause. As last time, we will prove that any such clause must have to be wide.

We define the measure of a clause D as

$$\mu(D) = \min\left\{ \left|V'\right| : \bigwedge_{v \in V'} PARITY_v \vDash D \right\} \quad . \tag{1.3}$$

In words, $\mu(D)$ is the minimal size of a subset of vertices V' such that any truth value assignment satisfying the parity constraint over this set must also satisfy D. We observe that $\mu(D)$ exists and is finite, since if we set $V' = V(G)$, then the set of constraints $\bigwedge_{v \in V'} PARITY_v = Ts(G,f)$ is unsatisfiable and an unsatisfiable formula implies anything, which in particular includes D (All satisfying truth value assignments to $\bigwedge_{v \in V'} PARITY_v$ must satisfy D for the simple reason that there are no such assignments). So $\mu(D) \leq n = |V(G)|$ for any clause D.

Let us verify that μ as defined in (1.3) has all the properties requested above.

1. If $C \in Ts(G,f)$ then $\mu(C) = 1$ since $C \in PARITY_v$ for some v.
2. $\mu(\bot) = n$ since $\bigwedge_{v \in V'} PARITY_v$ is satifiable for all $V' \subsetneqq V(G)$. To see this suppose that $v \in V(G) \setminus V'$ and consider f' such that $f'(w) = f(w)$ for all $w \in V(G) \setminus \{v\}$ but $f'(v) = 1 - f(v)$. Then by Lemma 1.4 the formula $Ts(G,f')$ is satisfiable since f' must have even weight, and since flipping $f'(v)$ to $1 - f(v)$ does not change $\bigwedge_{v \in V'} PARITY_v$ because $v \notin V'$ we have that $\bigwedge_{v \in V'} PARITY_v \subseteq Ts(G,f')$ is also satisfiable.
3. Suppose that $D \vee D'$ is derived by resolving $D \vee x$ and $D' \vee \overline{x}$. Fix V_1 and V_2 of minimal size such that $\bigwedge_{v \in V_1} PARITY_v \vDash D \vee x$ and $\bigwedge_{v \in V_2} PARITY_v \vDash D' \vee \overline{x}$. Then any assignment satisfying $\bigwedge_{v \in V_1 \cup V_2} PARITY_v$ must satisfy both of these clauses and hence also their resolvent $D \vee D'$. Thus, it must hold that

$$\mu(D \vee D') \leq |V_1 \cup V_2| \leq |V_1| + |V_2| = \mu(D \vee x) + \mu(D' \vee \overline{x}) \quad . \tag{1.4}$$

   A fancy way of rephrasing (1.4) in words is to say that μ is a *subadditive measure* with respect to resolution steps.

4. Because axioms have measure 1, contradiction has measure n, and the measure is subadditive, in any resolution refutation $\pi : Ts(G, f) \vdash \bot$ there must exist at least one clause D such that $\frac{n}{3} \leq \mu(D) \leq \frac{2n}{3}$.

Fix such a clause $D \in \pi$ and also some minimal V^* such that

$$\bigwedge_{v \in V^*} PARITY_v \vDash D \ . \tag{1.5}$$

By construction, $\frac{n}{3} \leq |V^*| \leq \frac{2n}{3}$ as the measure of D is this large. Note that this implies the same bounds for the size of the complement $\overline{V^*}$ of V^* in V, namely that $\frac{n}{3} \leq \left|\overline{V^*}\right| \leq \frac{2n}{3}$. Hence,

$$\left|E\left(V^*, \overline{V^*}\right)\right| \geq h(G) \cdot \min\left\{|V^*|, |\overline{V^*}|\right\} \geq h(G) \cdot \frac{n}{3} \ . \tag{1.6}$$

We claim that for all of the at least $h(G) \cdot \frac{n}{3}$ edges $e \in E\left(V^*, \overline{V^*}\right)$, the variables x_e must occur in D, from which Lemma 1.7 follows.

We establish the claim by contradiction. Consider a hypothetical edge $e^* \in E\left(V^*, \overline{V^*}\right)$ such that $x_{e^*} \notin Vars(D)$. Suppose $e^* = (v^*, w^*)$ for $v^* \in V^*$. Since V^* has minimal size among all vertex sets satisfying (1.5), if we look at the smaller set $V^* \setminus \{v^*\}$ there exists some assignment α such that $\alpha\left(\bigwedge_{v \in V^* \setminus \{v^*\}} PARITY_v\right) = 1$ but $\alpha(D) = 0$. Because of (1.5) it must hold that $\alpha(PARITY_{v^*}) = 0$. Let α^* be the same assignment as α except that we flip in on x_{e^*} so that $\alpha^*(x_{e^*}) = 1 - \alpha(x_{e^*})$. Then $\alpha^*(PARITY_{v^*}) = 1$ since the parity locally around v^* has been flipped, and we still have $\alpha^*\left(\bigwedge_{v \in V^* \setminus \{v^*\}} PARITY_v\right) = 1$ since x_{e^*} is not incident to $V^* \setminus \{v^*\}$ (the other endpoint w^* of e^* is in $\overline{V^*}$). Thus, for all of V^* we have $\alpha^*\left(\bigwedge_{v \in V^*} PARITY_v\right) = 1$. But since flipping a variable x_{e^*} that does not appear in D clearly does not change the truth value of D, we have $\alpha^*(D) = 0$. But this contradicts (1.5). The claim is proven.

Using this lemma, we can now prove the following theorem which states the so eagerly awaited lower bound.

Theorem 1.8 ([Urq87]). *Let $\{G_n\}_{n=1}^{\infty}$ be a family of graphs of size $\Theta(n)$ with constant indegree and edge expansion bounded away from zero (which exist); let $f_n : V(G_n) \to \{0, 1\}$ be odd-weight functions; and let $F_n = Ts(G_n, f_n)$ denote the corresponding family of Tseitin contradictions. Then it holds that the formulas F_n have size $\Theta(n)$, bounded width, and resolution refutation length*

$$L_{\mathcal{R}}(Ts(G_n, f_n) \vdash \bot) = \exp(\Omega(n)) \ .$$

Proof. We know

$$L_{\mathcal{R}}(F \vdash \bot) \geq \exp\left(\Omega\left(\frac{(W(F \vdash \bot) - W(F))^2}{n}\right)\right) \tag{1.7}$$

from [BW01], where n is the number of variables in the formula. Plugging in that F_n has $\Theta(n)$ variables and width $W(F_n) = \mathrm{O}(1)$ by Proposition 1.2, and that $W(F_n \vdash \bot) \geq h(G) \cdot \frac{n}{3} = \Omega(n)$ by Lemma 1.7, we get the truly exponential bound claimed. □

And now for something completly different...

2 Space Complexity in Resolution

Memory usage is a very important factor for modern state-of-the-art SAT solvers. Therefore, it is also of interest to study this issue 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 the rest of 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 given in the very first lecture 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.

We now make these notions more formal.

Definition 2.1. A *resolution derivation* is a sequence of sets of clauses, or *clause configurations*, $\{\mathbb{C}_0, \dots, \mathbb{C}_\tau\}$ 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: $\mathbb{C}_t = \mathbb{C}_{t-1} \setminus \{C\}$ for a clause $C \in \mathbb{C}_{t-1}$.

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 $\{\mathbb{C}_0, \dots, \mathbb{C}_\tau\}$ 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*[1] 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

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

[1] Recall from the first lecture that there are also other ways of measuring space. For resolution, the most studied measure is clause space, however, and this lecture will focus almost exclusively on this measure. So when we say only "space" in what follows, we mean "clause space."

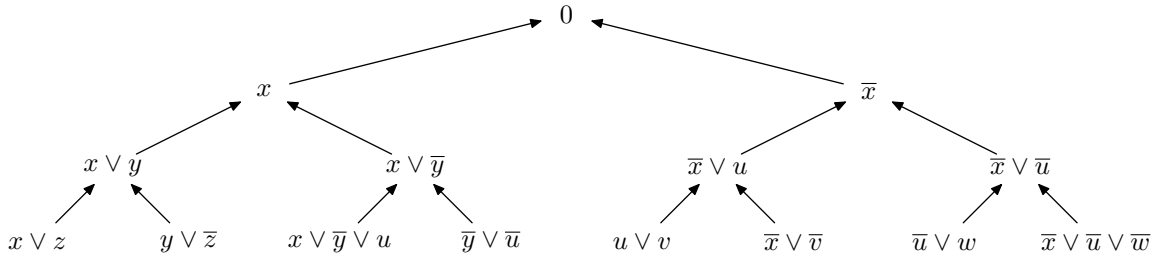


Figure 1: Example tree-like resolution refutation represented as a DAG.

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.

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.

Theorem 2.3 ([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 lecture 2, we sketched the correspondence between tree-like resolution refutations and decision trees[2] 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 1 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 1).

- **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.

[2] Working out the formal details of this correspondence is one of the problems in the first problem set.

Definition 2.4. An unsatisfiable CNF formula F is *minimally unsatisfiable* if removing any clause from F makes it satisfiable.

Example 2.5. 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 2.6 (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.

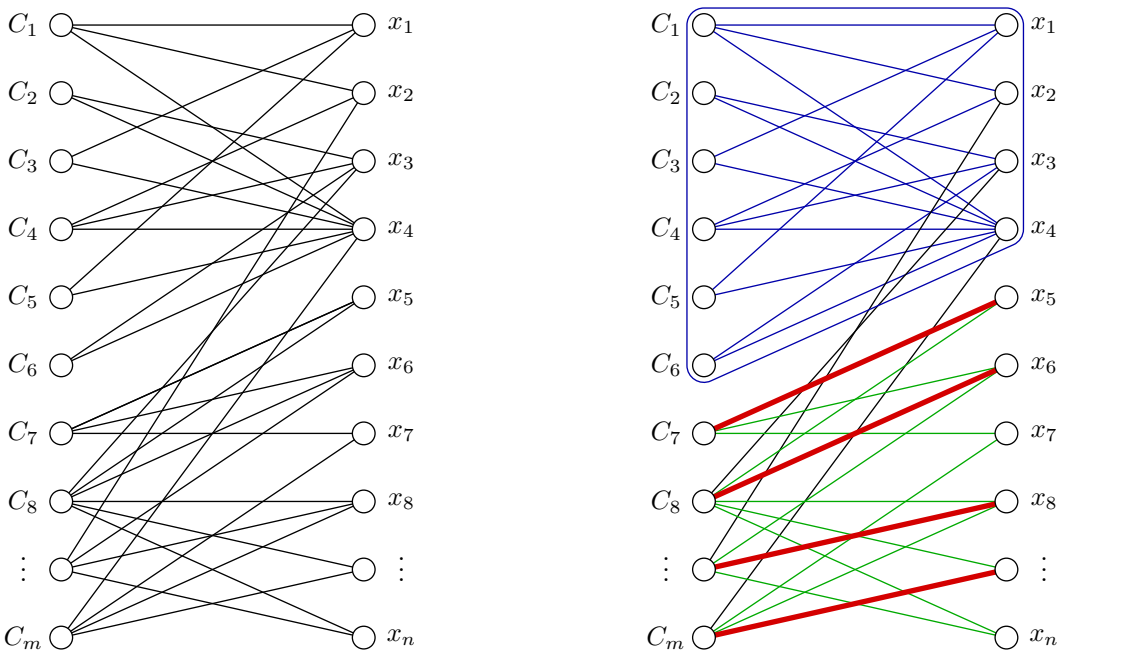
Theorem 2.7. *Let $G = (U \cup V, E)$ be a bipartite graph. Then there is a matching of U into V if and only if for all subsets $U' \subseteq U$ it holds that $\left|N(U')\right| \geq \left|U'\right|$.*

Proof of Lemma 2.6. 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 2(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 2.7 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.



(a) Bipartite graph corresponding to formula F.

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

Figure 2: Illustration of argument in proof of Tarsi's lemma.

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 2(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 2(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 2.8 ([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 2.6 that $\left|F'\right| > \left|Vars(F')\right|$, and combining this with Theorem 2.3 we get that $Sp(F' \vdash \bot) \leq \left|Vars(F')\right| + 2 \leq \left|F'\right| + 1 \leq |F| + 1$. □

We have proven that

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

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 2.9 ([ABRW00, Tor99]). *There is a polynomial-size family $\{F_n\}_{n=1}^{\infty}$ of unsatisfiable 3-CNF formulas such that $Sp_{\mathcal{R}}(F \vdash \bot) = \Omega(|F|) = \Omega(|Vars(F)|)$.*

(The journal version of [ABRW00] is [ABRW02], and [Tor99] later appeared as a part of [ET01].)

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 [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 (cf. Theorem 1.8) which also 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, [BG01] (journal version in [BG03]) proved optimal space bounds for random k-CNF formulas, 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 width actually provides a lower bound on clause space.

This conjecture turned out to be perfectly correct and was proven true in the beautiful paper [AD03] by Atserias and Dalmau (the corresponding journal article took a while before it appeared as [AD08]). 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 2.9 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*.[3] 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.

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 $x \in \mathit{Vars}(F)$ 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"

[3] These names are used for historical reasons.

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]). *The CNF formula 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 $\pi = \{\mathbb{C}_0 = \emptyset, \mathbb{C}_1, \ldots, \mathbb{C}_\tau = \{\bot\}\}$ 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 2.9 (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 will conclude this lecture by starting to discuss the tools used to answer this question, and will continue on the same topic next time.[4]

4 Pebble Games and 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 very different games played on DAGs. It the pebble games we are interested in now, we should think of a DAG G as modelling some kind of computation, and we use pebbling to determine the time and space needed to carry out this computation. More precisely, the time needed for calculation is the number of pebbling moves and the space needed is the number of pebbles.

Before going any further, let us agree on some terminology. DAGs consist of vertices with directed edges between them. Vertices with no incoming edges are sources and vertices with no outgoing edges are sinks. The (immediate) predecessors of a vertex v are all vertices u having edges to v. In what follows G is assumed to have one unique sink, which we denote z.

In the *black-white pebble game*, as defined by [CS76], we start with all vertices in G empty and want to finished with the whole graph empty again except for z which should contain a black pebble. The rules of the pebble game are the following:

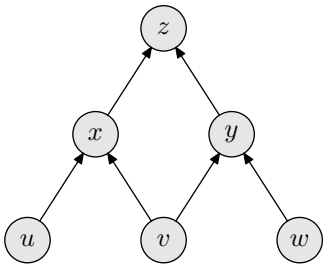
1. One can place a black pebble on an empty vertex v if all its predecessors have pebbles on them.
2. It is always possible to remove a black pebble from any vertex.
3. It is always possible to place a white pebble on any empty vertex.
4. If all the predecessors of a white pebble on v have pebbles, it is possible to remove the white pebble from v.

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 speficic instances of pebble games on graphs, and such formulas have interesting properties.

If G is a DAG with sources S and a unique sink z we associate a variable v with every vertex $v \in V(G)$. The *pebbling contradiction* over G, denoted Peb_G, consists of the following clauses over these variables:

- For all source vertices $s \in S$, we have a unit clause (i.e., a clause of size 1) s—such clauses are called *source axioms*.
- For all non-source vertices v with immediate predecessors $u_1, \ldots, u_\ell$, we have a clause $\overline{u}_1 \vee \cdots \vee \overline{u}_\ell \vee v$—a pebbling axiom.

[4] Or that was the plan, but it did not quite work out. Instead, there is now a separate write-up "Notes on pebble games and pebbling contradictions" on the course webpage in which you can read all about this.



(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 3: Pebbling contradiction for the pyramid graph Π_2.

- Finally, for the sink z we have the unit clause $\overline{z}$—the *sink axiom*.

See Figure 3 for an example of this. 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.

At the very end of the lecture, we argued that although this is a priori not abvious at all, there is a close relation between resolution refutations of pebbling contradictions and black-white pebblings of graphs. Namely, from any resolution refutation of a pebbling contradiction Peb_G, one can extract a black-white pebbling of the graph G, and the time and space of this pebbling is at least as good as the length and space of the resolution refutation. This was proven by Ben-Sasson in [Ben02] (but as usual a better reference is the full-length journal version [Ben09]). A somewhat informal statement of this result is as follows.[5]

Theorem 4.1 ([Ben09]). *Any resolution refutation π of a pebbling contradiction Peb_G over G can be translated into black-white pebbling $\mathcal{P}$ of G such that the time of the pebbling $\mathcal{P}$ is upper-bounded by the length of the refutation π and the space of $\mathcal{P}$ is upper-bounded by the total space of π.*

We ended the lecture by giving a "proof by example" of Theorem 4.1.[6] We did not even attempt to make any formal proof that this always works out as nicely as in the example we saw, but we briefly described the general outline of how to show such a thing.

The idea is to let positive literals in a clause configuration correspond to black pebbles and negative literals correspond to white pebbles. Then for each clause configuration in a refutation we get sets of black and white pebbles on the graph, and the number of pebbles is clearly upper-bounded by the total space of the clause configuration. One now needs to argue that as the resolution refutation proceeds, the sets of black and white pebbles on the graph appear and disappear in accordance with the rules of the pebble game. This is not quite true as stated, but if one just works out some necessary technicalities[7] then everything pans out and Theorem 4.1 follows.

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.

[5] Again we refer to the separate write-up "Notes on pebble games and pebbling contradictions" on the course webpage for more details.

[6] This example illustration can be studied in the lecture slides posted on the course webpage.

[7] Doing this is not entirely trivial, however, and this will be one of the problems in the first problem set.

[ABRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002. Preliminary version appeared 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 appeared in *CCC '03*.

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

[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 appeared 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, 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 appeared in *CCC '01*.

[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 appeared 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.

[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, 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].

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

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