Chapter 18
Resolution

Propositional proof systems operate with boolean formulas, the simplest of which are clauses, that is, ORs of literals, where each literal is either a variable x_i or its negation $\neg x_i$. A *truth-assignment* is an assignment of constants 0 and 1 to all the variables. Such an assignment *satisfies* (*falsifies*) a clause if it evaluates at least one (respectively, none) of its literals to 1. A set of clauses, that is, a CNF formula, is *satisfiable* if there is an assignment which satisfies all its clauses. The basic question is: Given an *unsatisfiable* CNF formula F, what is the size of a proof that F is indeed unsatisfiable? The *size* (or *length*) of a proof is the total number of clauses used in it.

A proof of the unsatisfiability of F starts with clauses of F (called *axioms*), at each step applies one of several (fixed in advance) simple rules of inferring new clauses from the already derived ones, and must eventually produce the empty clause $\emptyset$ which, by definition, is satisfied by none of the assignments.

For such a derivation to be a legal proof, the rules must be *sound* in the following sense: if some assignment (of constants to all variables) falsifies the derived clause, then it must falsify at least one of the clauses from which it was derived. Then the fact that $\emptyset$ was derived implies that the CNF F was indeed unsatisfiable: given any assignment $a \in \{0,1\}^n$ we can traverse the proof going from $\emptyset$ to an axiom (a clause of F), and the soundness of the rules will give us a clause of F which is not satisfied by a.

The main goal of proof complexity is to show that some unsatisfiable CNFs require long proofs. A compelling reason to study this problem is its connection with the famous P versus NP question. It has long been known (Cook and Reckhow, 1979) that NP $=$ co-NP iff there is a propositional proof system giving rise to short (polynomial in $|F|$) proofs of unsatisfiability of all unsatisfiable CNFs F; here and throughout, $|F|$ denotes the number of clauses in F.

Thus a natural strategy to approach the NP versus co-NP problem, and hence, also the P versus NP problem is, by analogy with research in circuit complexity, to investigate more and more powerful proof systems and show that some unsatisfiable CNFs require exponentially long proofs. In this and the next chapter we will demonstrate this (currently very active) line of research on some basic proof systems, like resolution and cutting planes proofs.

S. Jukna, *Boolean Function Complexity*, Algorithms and Combinatorics 27,
DOI 10.1007/978-3-642-24508-4_18, © Springer-Verlag Berlin Heidelberg 2012

The areas of circuits complexity and proof complexity look similar, at least syntactically: in proof complexity one also starts from some "simplest" objects (axioms) and applies local operations to obtain a result. But there is a big difference: the number of "objects of interest" differ drastically between the two settings. There are doubly exponentially number of boolean functions of n variables, but only exponentially many CNFs of length n. Thus a counting argument shows that some functions require circuits of exponential size (see Theorem 1.14), but no similar argument can exist to show that some CNFs require exponential size proofs. This is why even *existence* results of hard CNFs for strong proof systems are interesting in this setting as well.

The most basic proof system, called the *Frege system*, puts no restriction on the formulae manipulated by the proof. It has one derivation rule, called the cut rule: from $A \vee C$ and $B \vee \neg C$ one can derive $A \vee B$ in one step. Adding any other sound rule turns out to have little effect on the length of proofs in this system. The major open problem in proof complexity is to find any tautology that has no polynomial-size proof in the Frege system. As lower bounds for Frege are hard to obtain, we turn to subsystems of Frege which are interesting and natural. One of the simplest and most important such subsystems is called *Resolution*. This subsystem is used by most propositional, as well as first order automated theorem provers.

18.1 Resolution Refutation Proofs

The resolution proof system was introduced by Blake (1937) and has been made popular as a theorem-proving technique by Davis and Putnam (1960) and Robinson (1965). Let F be a set of clauses and suppose that F is not satisfiable. A *resolution refutation proof* (or simply, a *resolution proof*) for F is a sequence of clauses $\mathcal{R} = (C_1, \ldots, C_t)$ where $C_t = \emptyset$ is the empty clause and each intermediate clause C_i either belongs to F or is derived from some previous two clauses using the following *resolution rule*:

$$\frac{A \vee x_i \qquad B \vee \neg x_i}{A \vee B} \tag{18.1}$$

meaning that the clause $A \vee B$ can be inferred from two clauses $A \vee x_i$ and $B \vee \neg x_i$.

In this case one also says that the variable x_i was *resolved* to derive the clause $A \vee B$; here A and B are arbitrary ORs of literals. The *size* of such a proof is equal to the total number t of clauses in the derivation. It is often useful to describe a resolution proof as a directed acyclic graph (see Fig. 18.1). If this graph is a tree, then one speaks about a *tree-like* resolution proof. For technical reasons the following "redundant" rule, the *weakening rule*, is also allowed: a clause $A \vee B$ can be inferred from A.

Observe that the resolution rule is sound: if some assignment (of constants to all variables) falsifies the derived clause $A \vee B$, then it must falsify at least one of the

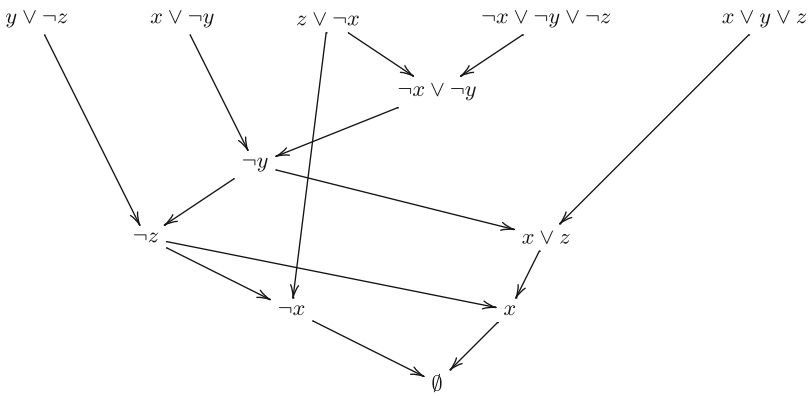

Fig. 18.1 A resolution refutation proof of an unsatisfiable CNF formula F. Leaves (fanin-0 nodes) are clauses of F, and each inner node is a clause obtained from previous ones by the resolution rule. This proof is *not* tree-like

clauses $A \vee x_i$ and $B \vee \neg x_i$ from which it was derived. It is also known (and easy to show, see Exercise 18.1) that Resolution is *complete*: every unsatisfiable set of clauses has a resolution refutation proof.

Interestingly, resolution proofs are related to the model of computation we already considered above—branching programs.

18.2 Resolution and Branching Programs

Let F be an unsatisfiable CNF formula, that is, for every input $a \in \{0,1\}^n$ there is a clause $C \in F$ for which $C(a) = 0$. The *search problem* for F is, given a, to find such a clause; there may be several such clauses—the goal is to find at least one of them. Such a problem can be solved by a branching program with at most $n|F|$ nodes. Namely, given an assignment $a \in \{0,1\}^n$, we can test whether all literals of the first clause in F are falsified by a. If yes, then we reach a leaf labeled by this clause. If not, then test whether all literals of the second clause in F are falsified by a, etc. Note that the resulting branching program is not read-once: if a variable appears in k clauses, then it will be retested k times.

Of course, the search problem for any unsatisfiable CNF formula can be solved by a decision tree, and hence, by a read-once branching program. But the size (total number of nodes) may be then exponential in the number n of variables.

Let $S_R(F)$ be the smallest size of a resolution refutation of F, and $\mathrm{BP}(F)$ the smallest size of (the number of nodes in) a deterministic branching program solving the search problem for F. It is not difficult to show that $S_R(F) \geq \mathrm{BP}(F)$ (see the first part of the proof of Theorem 18.1 below). But the gap between these two measures may be exponential: as mentioned above, any unsatisfiable CNF F has a

trivial branching program of size $n|F|$ whereas, as we will show in the next section, some CNFs require $S_R(F)$ exponential in their variables.

The first exponential lower bounds for resolution proofs were obtained long ago by Tseitin (1970) under an additional restriction that along every path every particular variable x_i can be resolved at most once. He called this model *regular* resolution. It turns out that this model exactly coincides(!) with the familiar model of read-once branching programs.

Let 1-$S_R(F)$ be the smallest size of a regular resolution refutation proof for F, and 1-BP(F) the smallest size of a deterministic read-once branching program solving the search problem for F.

The following theorem was used implicitly by various authors and explicitly noted in Lovász et al. (1995).

Theorem 18.1. *For every unsatisfiable CNF formula F, we have*

$$S_R(F) \geq \text{BP}(F) \quad \textit{and} \quad 1\text{-}S_R(F) = 1\text{-BP}(F).$$

Proof. Resolution proofs $\Rightarrow$ branching programs: To show that $S_R(F) \geq \text{BP}(F)$ and 1-$S_R(F) \geq$ 1-BP(F), let $\mathcal{R}$ be a resolution refutation proof for F. Construct a branching program as follows.

- The nodes of the program are clauses C of $\mathcal{R}$.
- The source node is the last clause in $\mathcal{R}$ (the empty one), the sinks are the initial clauses from F.
- Each non-sink node C has fanout 2 and the two edges directed from C to the two clauses C_0 and C_1 from which this clause is derived by one application of the resolution rule. If the resolved variable of this inference is x_i then the edge going to the clause containing x_i is labeled by the test $x_i = 0$, and the edge going to the clause containing $\neg x_i$ is labeled by the test $x_i = 1$ (see Fig. 18.2).

It is straightforward to verify that *all* clauses on a path determined by an input $a \in \{0,1\}^n$ are falsified by a, and hence, the last clause of F reached by this path is also falsified by a. That is, the obtained branching program solves the search problem, and is read-once if $\mathcal{R}$ was regular.

Read-once branching programs $\Rightarrow$ regular resolution proofs: It remains to prove the more interesting direction that 1-$S_R(F) \leq$ 1-BP(F). Let P be a deterministic read-once branching program (1-b.p.) which solves the search problem for F. That is, for every input $a \in \{0,1\}^n$ the (unique) computation path on a leads to a clause $C \in F$ such that $C(a) = 0$. We will associate a clause to every node of P such that P becomes a graph of a resolution refutation for F. A vertex v labeled by a variable will be associated with a clause C_v with the property that

$$C_v(a) = 0 \text{ for every input } a \in \{0,1\}^n \text{ that reaches } v. \tag{18.2}$$

We associate clauses inductively from the sinks backwards. If v is a sink then let C_v be the clause from F labeling this sink in the program P.

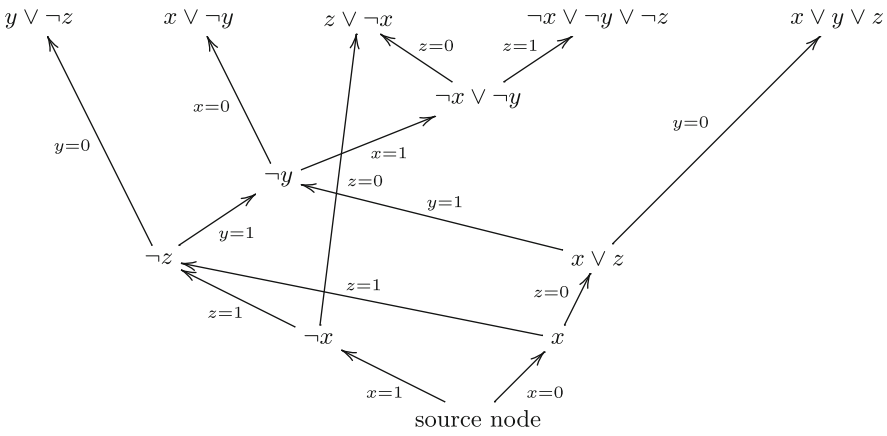

Fig. 18.2 A branching program obtained from the resolution proof given in Fig. 18.1: just reverse the direction of arcs and label them accordingly. The program is not read-once

Now assume that the node v of P corresponds to a variable x_i and has edges (v, u_0) for $x_i = 0$ and (v, u_1) for $x_i = 1$. By induction we may assume that u_0 and u_1 are labeled by clauses C_0 and C_1 satisfying (18.2).

Claim 18.2. *C_0 does not contain $\neg x_i$ and C_1 does not contain x_i.*

Proof. Otherwise, if C_0 contains $\neg x_i$, take an input a with $a_i = 0$ that reaches v. Such an input exists since by the read-once assumption on P, the i-th bit x_i was not asked along any path from the source to v. The input a can reach u_0 and it satisfies C_0, in contradiction to the inductive hypothesis. The proof in the case when C_1 contains x_i is similar. □

We conclude that

(i) Either $C_0 = (x_i \vee A)$ and $C_1 = (\neg x_i \vee B)$,
(ii) Or one of C_0 and C_1 does not contain $x_i, \neg x_i$ at all.

In the first case label v with $C_v = A \vee B$. In the second case label v with the clause that does not contain $x_i, \neg x_i$. (If both clauses do not contain $x_i, \neg x_i$ choose any of them.)

It is easy to see that the inductive hypothesis (18.2) holds for C_v. Indeed, if C_v were satisfied by some (partial) input a reaching the node v then, due to the read-once property, this input could be extended to two inputs a_0 and a_1 by setting the i-th bit to 0 and to 1. But $C_v(a) = 1$ implies that either $A(a) = 1$ or $B(a) = 1$ (or both). Hence, we would have that either $C_0(a_1) = 1$ or $C_1(a_1) = 1$, contradicting the inductive hypothesis (18.2).

Finally, the clause associated with the source node must be the empty clause, just because *every* input reaches it. Thus the obtained labeled digraph represents a regular resolution derivation for F (possibly with some redundant steps that correspond to the second case (ii) in the labeling above. □

We thus have a bridge between resolution refutations and branching programs:

- Every resolution refutation is a restricted branching program.
- Every *regular* resolution refutation is just a *read-once* branching program.
- Every *tree-like* resolution refutation is just a *decision tree*.

The only difference is that now these branching programs solve *search* problems, not just *decision* problems.

Remark 18.3. Note that Claim 18.2 holds for *any* deterministic branching program, not just for read-once programs: it is enough that P is a *minimal* program. Indeed, in this case a node must be reachable by (at least) two inputs a and b such that $a_i = 0$ and $b_i = 1$, for otherwise the test on the i-th bit made at the node v would be redundant. However, the fact that the branching program is read-once was essential to show that the constructed clause C_v satisfies (18.2).

To see this, let $C_0 = (x_i \vee A)$, $C_1 = (\neg x_i \vee B)$ and $C_v = A \vee B$. Suppose that the node v is reached by two inputs a and b such that $a_i = 0$ and $b_i = 1$. Assume that the bit x_i *was* tested along both paths at least once; hence, the paths must diverge after the test on x_i at the node v, that is, a cannot reach C_1, and b cannot reach C_0. Assume now that $A(a) = B(b) = 0$ but $A(b) = 1$ or $B(a) = 1$. Then $C_0(a) = 0$ and $C_1(b) = 0$ but $C_v(a) = 1$ or $C_v(b) = 1$. In the read-once case such a situation cannot occur because then every (single!) computation reaching a node v can be extended in *both* directions.

18.3 Lower Bounds for Tree-Like Resolution

Let F be an unsatisfiable CNF formula. A resolution proof for F is *tree-like* if its underlying graph is a tree. That is, tree-like resolution proof is a special case of regular resolution proofs; the corresponding branching program for the corresponding search problem is then just a decision tree. By the *size* $|T|$ of a tree-like resolution proof T we will mean the number of leaves in the corresponding decision tree. Since the search problem for any unsatisfiable CNF formula can be solved by a decision tree, Theorem 18.1 implies that any such CNF formula *has* a tree-like resolution proof, and hence, also regular resolution proof. The question, however, is: how large tree-like proofs must be?

Lower bounds on the size of tree-like resolution can be proved using the following game-theoretic argument proposed by Pudlák and Impagliazzo (2000). There are two players, Prover and Delayer. The goal of the Prover is to construct a (partial) assignment falsifying at least one clause of F. The goal of Delayer is to delay this happening as long as possible. The game proceeds in rounds. In each round

- Prover suggests a variable x_i to be set in this round, and
- Delayer either chooses a value 0 or 1 for x_i or leaves the choice to the Prover.

- In this last case, Delayer scores one point, but the Prover can then choose the value of x_i.

The game is over when one of the clauses is falsified by the obtained (partial) assignment, that is, when all the literals in the clause are assigned 0.

Pudlák and Impagliazzo observed that, if the Delayer has a strategy which scores r points, then any tree-like resolution refutation proof for F has size at least 2^r. This holds because, given a tree-like derivation, the Prover can use the following strategy: if the Delayer leaves the choice to the Prover, then the Prover chooses an assignment resulting into a *smaller* of the two subtrees.

This result can be easily extended to the case of *asymmetric* games, where the Delayer earns different number of points depending on whether the prover sets $x_i = 0$ or $x_i = 1$. As before, the game proceeds in rounds. In each round

- Prover suggests a variable x_i to be set in this round, and
- Delayer either chooses a value 0 or 1 for x_i or leaves the choice to the Prover
- The number of points earned by the Delayer is
  - 0 if Delayer chooses the value for x_i,
  - $\log_2 a$ if Prover sets x_i to 0, and
  - $\log_2 b$ if Prover sets x_i to 1.

The only requirement is that $1/a + 1/b = 1$; in this case we say that (a, b) is a legal scoring pair, and call this game the (a, b)-game. Hence, the symmetric game is one with $a = b = 2$.

Lemma 18.4. *Let F be an unsatisfiable CNF formula F. If Delayer can earn r points in some asymmetric game on F, then any tree-like resolution refutation proof for F has size at least 2^r.*

Proof. We will prove the lemma in the converse direction: if F has a tree-like resolution refutation proof T with $|T|$ leaves then, in any (a, b) game for F, the Prover has a strategy under which the Delayer can earn at most $\log |T|$ points.

Consider an arbitrary (a, b)-game on F, and let α_i be the partial assignment constructed after i rounds of the game (the i-th prefix of α). By p_i we denote the number of points that Delayer has earned after i rounds, and let T_i be the sub-tree of T which has as its root the node reached in T along the path specified by α_i. Our goal is to prove, by induction on i, that

$$|T_i| \leq \frac{|T|}{2^{p_i}}. \tag{18.3}$$

The desired inequality $p_m \leq \log |T|$ then follows, because T_m consists of just one clause falsified by α.

So it remains to prove the claim (18.3). At the beginning of the game ($i = 0$) we have $p_0 = 0$ and $T_0 = T$. Therefore the claim trivially holds.

For the inductive step, assume that the claim holds after i rounds and Prover asks for the value of the variable x in round $i+1$. The variable Prover asks about is determined by T: it is the variable resolved at the root of the subtree reached after the i-th round.

It the Delayer chooses the value, then $p_{i+1} = p_i$ and (18.3) remains true after the $(i+1)$-th round. Otherwise, let T_i^0 be the 0-subtree of T_i, and T_i^1 the 1-subtree of T_i. Since $1/a + 1/b = 1$, we have that

$$|T_i^0| + |T_i^1| = |T_i| = \frac{|T_i|}{a} + \frac{|T_i|}{b}.$$

If the Delayer defers the choice to the Prover, then the Prover can use the following "take the smaller tree" strategy: set $x = 0$ if $|T_i^0| \le |T_i|/a$, and set $x = 1$ otherwise; in this last case we have that $|T_i^1| \le |T_i|/b$. Thus if Prover's choice is $x = 0$, then we get

$$|T_{i+1}| \le \frac{|T_i|}{a} \le \frac{|T|}{a2^{p_i}} = \frac{|T|}{2^{p_i + \log a}} = \frac{|T|}{2^{p_{i+1}}},$$

as desired. Since the same holds (with a replaced by b) if Prover's choice is $x = 1$, we are done. □

We now apply this lemma to prove that the unsatisfiable CNF corresponding to the pigeonhole principle (and even to its "weak" version) require tree-like resolution refutation proofs of exponential size.

The *weak pigeonhole principle* asserts that if $m > n$ then m pigeons cannot sit in n holes so that every pigeon is alone in its hole.[1] In terms of 0-1 matrices, this principle asserts that, if $m > n$ then no $m \times n$ 0-1 matrix can simultaneously satisfy the following two conditions:

1. Every row has at least one 1.
2. Every column has at most one 1.

To write this principle as an unsatisfiable CNF formula, we introduce boolean variables $x_{i,j}$ interpreted as:

$$x_{i,j} = 1 \text{ if and only if the } i\text{-th pigeon sits in the } j\text{-th hole.}$$

Let PHP_n^m denote the CNF consisting of the following clauses:

- Pigeon Axioms: each of the m pigeon sits in at least one of n holes:

$$x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n} \text{ for all } i = 1, \ldots, m.$$

[1] The word "weak" is used here to stress that the number m of pigeons may be arbitrarily large. The larger m is, the "more obvious" the principle is, and hence, its proof might be shorter.

- Hole Axioms: no two pigeons sit in one hole:

$$\neg x_{i_1,j} \vee \neg x_{i_2,j} \quad \text{for all } i_1 \neq i_2 \text{ and } j = 1, \ldots, n.$$

Hence, truth assignments in this case are boolean $m \times n$ matrices α. Such a matrix can satisfy all pigeon axioms iff every row has at least one 1, whereas it can satisfy all hole axioms iff every column has at most one 1. Since $m \geq n+1$, no assignment can satisfy pigeon axioms and hole axioms at the same time. So PHP_n^m is indeed an unsatisfiable CNF.

Theorem 18.5. (Dantchev and Riis 2001) *For any $m > n$, any tree-like resolution refutation proof of PHP_n^m has size $n^{\Omega(n)}$.*

Note that the lower bound does not depend on the number m of pigeons—it may be arbitrarily large! A smaller (but also exponential) lower bound of the form $2^{\Omega(n)}$ for an arbitrary number of pigeons was proved earlier by Buss and Pitassi (1998).

Proof. (Due to Beyersdorff et al. 2010) By Lemma 18.4, we only have to define an appropriate scoring pair (a, b) for which the Delayer has a strategy giving her many points in the (a, b) game on PHP_n^m. We first define the Delayer's strategy for an arbitrary (a, b) game, and then choose a and b so that to maximize the total score. By Lemma 18.4, it is enough to show that the Delayer can earn $\Omega(n \log n)$ points.

The goal of the Delayer is to delay an appearance of two 1s in a column and of an all-0 row as long as possible. So if Prover asks for a value of $x_{i,j}$, then the Delayer is only then forced to set it to 0 if the j-th column already has a 1. Otherwise it is beneficial for the Delayer to set $x_{i,j} = 1$ to avoid an all-0 row. But at the same time, it is beneficial for her not to set too many 1s in a row to avoid two 1s in a column. Intuitively, it would be the best for the Delayer to set just one 1 per row. Moreover, she should not wait too long: if the i-th row already has many 0s, she should try to set a 1 in it, for otherwise she could be forced (by many columns already having a 1) to set the remaining variables in this row to 0s.

To formally describe the strategy of the Delayer, let α be a partial assignment to the variables $X = \{x_{i,j} \mid i \in [m], j \in [n]\}$. For pigeon i, let $J_i(\alpha)$ be the set of "excluded free holes" for the pigeon i. These are the holes which are still free (not occupied by any pigeon) but are explicitly excluded for pigeon i by α:

$$J_i(\alpha) := \{j \in [n] \mid \alpha(x_{i,j}) = 0 \text{ and } \alpha(x_{i',j}) \neq 1 \text{ for all } i' \in [m]\}.$$

If Prover asks for a value of $x_{i,j}$, then the Delayer uses the following strategy (see Fig. 18.3):

$$\alpha(x_{i,j}) := \begin{cases} 0 & \text{if either the } i\text{-th row or the } j\text{-th column already has a 1;} \\ 1 & \text{if } |J_i(\alpha)| \geq n/2 \text{ and there is no 1 in the } j\text{-th column yet;} \\ * & \text{otherwise.} \end{cases}$$

Here $*$ means the decision is deferred to the Prover.

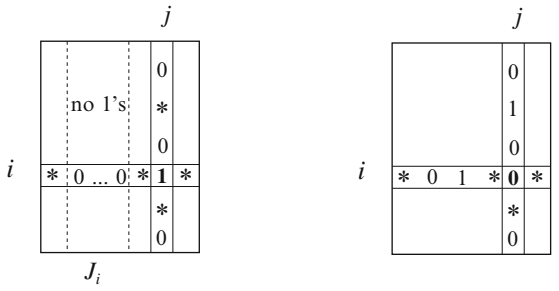

Fig. 18.3 The strategy of the Delayer: she sets $x_{i,j} = 1$ if $|J_i| \geq n/2$ and there is no 1 in the j-th column, and sets $x_{i,j} = 0$ if either the i-th row or the j-th column already has a 1. Otherwise, she defers the decision to the Prover

If Delayer uses this strategy, then none of the hole clauses $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ from PHP_n^m will be falsified in the game. Therefore, a contradiction (a falsified clause) will be a pigeon clause $x_{i,1} \vee \cdots \vee x_{i,n}$. That is, the resulting assignment α sets all n variables in this clause to zero (pigeon i has no hole).

But after the number $|J_i(\alpha)|$ of excluded free holes for pigeon i reaches the threshold $n/2$, Delayer will not leave the choice to Prover. Instead, Delayer will try to place pigeon i into some hole. Since this hasn't happened, the Delayer was forced to set the remaining $n/2$ variables in the i-th row to 0. Since the Delayer is only then forced to set $x_{i,j}$ to 0 when the j-th column already has a 1, there must already be a 1 in each of these $n/2$ columns. Moreover, no two of these 1s can be in one row, since Delayers strategy forbids this: she always sets a "dangerous" variable (with a 1 in the same row or column) to 0. Therefore, at the end of the game at least $n/2$ variables must be set to 1, and no two of these 1s lie in one row or one column. We assume w.l.o.g. that these are the variables x_{i,j_i} for $i = 1, \ldots, n/2$. Let us check how many points Delayer earns in this game. We calculate the points separately for each pigeon $i = 1, \ldots, n/2$.

Case 1: Delayer sets x_{i,j_i} to 1. Then pigeon i was not assigned to a hole yet, and $|J_i(\alpha)| \geq n/2$. Hence, there must be a set J of $|J| \geq n/2$ 0-positions in the i-th row of α. Moreover, all these positions must be already set to 0 by the Prover (not by the Delayer) because none of the columns $j \in J$ can have a 1, by the definition of $J_i(\alpha)$. Thus before Delayer sets $\alpha(x_{i,j_i}) = 1$, she has already earned points for all $|J| \geq n/2$ previous 0-settings by the Prover. That is, in this case Delayer earns at least $(n/2)\log a$ points.

Case 2: Player sets x_{i,j_i} to 1. In this case Delayer earns $(n/2)\log b$ points.

Thus, the Delayer earns either $(n/4)\log b$ or $(n^2/8)\log a$ points. To maximize the score, we set $b = n/\log n$ and

$$a = \frac{b}{b-1} = 1 + \frac{1}{b-1} = \Omega\left(\mathrm{e}^{1/(b-1)}\right) = 2^{\Omega\left(\frac{\log n}{n}\right)}.$$

Since $1/a + 1/b = (b-1)/b + 1/b = 1$, this is a legal scoring, and Delayer earns $\Omega(n \log n)$ points, as desired. □

18.4 Tree-Like Versus Regular Resolution

A partial ordering of A is a binary relation $a \to b$ which is antisymmetric and transitive. That is, $a \to b$ implies $\neg(b \to a)$, and $a \to b$ and $b \to c$ implies $a \to c$. An element $a \in A$ is *minimal* if it has no predecessor, that is, if $\neg(b \to a)$ for all $b \in A$, $b \neq a$. It is clear that in each partial order there must be at least one minimal element. We consider the CNF formula GT_n expressing the negation of this property. For this we take $A = \{1, \ldots, n\}$ and associate a boolean variable x_{ij} to each pair (i, j) of elements. We interpret these variables as $x_{ij} = 1$ if and only if $i \to j$.

The CNF formula GT_n consists of three sets of clauses. The first two sets consist of all clauses $\neg x_{ij} \vee \neg x_{jk} \vee x_{ik}$ and $\neg x_{ij} \vee \neg x_{ji}$ for all distinct i, j, k. These clauses ensure that we have a partial ordering. The third set consists of n clauses

$$C_n(j) = x_{1j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{n,j}, \qquad j = 1, \ldots, n$$

stating that every element j has at least one predecessor (no minimal element). In terms of graphs, the CNF formula GT_n is a negation of the property that if a directed graph is transitive and has no loops and no cycles of size two, then there must be at least one source node, that is, a node of fanin 0.

It was conjectured that GT_n requires resolution refutation proofs of exponential size. And indeed, it was shown by Bonet and Galesi (1999) that *tree-like* refutations for this CNF must be of exponential size. However, Stålmark (1996) showed that this CNF has a small *regular* resolution refutation.

Theorem 18.6. (Stålmark 1996) *The CNF formula GT_n has a regular resolution refutation of polynomial size.*

Proof. We will construct the desired refutation proof recursively. Our initial clauses (axioms) are $A(i, j, k) = \neg x_{ij} \vee \neg x_{jk} \vee x_{ik}$, $B(i, j) = \neg x_{ij} \vee \neg x_{ji}$, and the clauses $C_n(j)$ for all $j = 1, \ldots, n$. We introduce auxiliary clauses

$$C_m(j) = \quad x_{1j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{m,j}$$

for all $m = 2, \ldots, n$, stating that some element $i \in \{1, \ldots, m\}$ is smaller than j. The idea of the proof is to obtain clauses of the form $C_m(j)$ from $m = n$ down to $m = 2$ in the following way:

$$\begin{array}{cccccc} C_n(1) & C_n(2) & \ldots & C_n(n-1) & C_n(n) \\ C_{n-1}(1) & C_{n-1}(2) & \ldots & C_{n-1}(n-1) & \\ & \vdots & & & \\ C_2(1) & C_2(2) & & & \end{array}$$

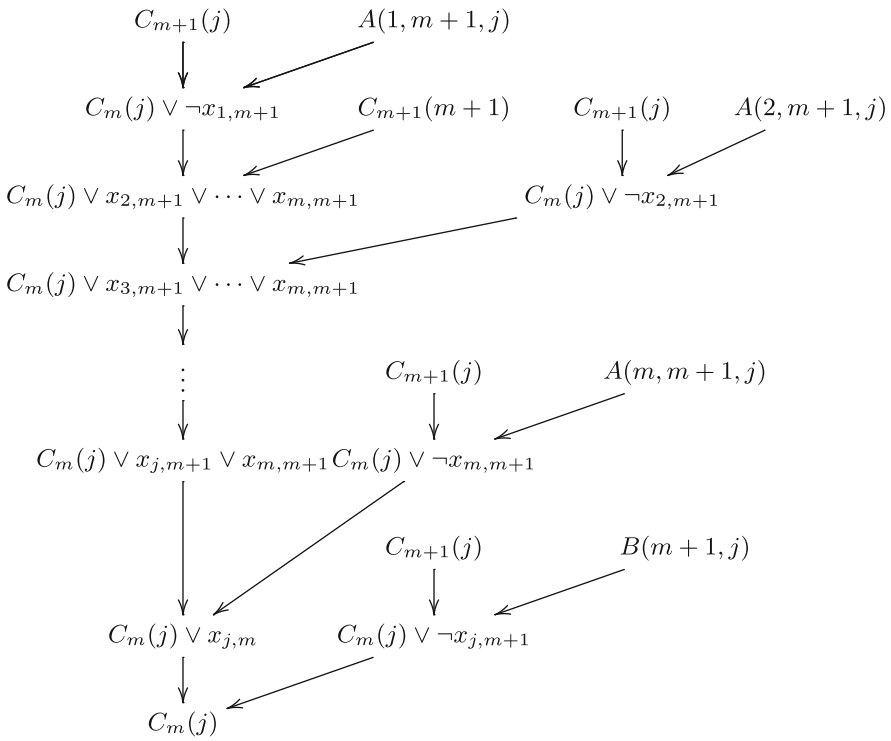

Fig. 18.4 Resolution derivation of $C_m(j)$, for $j \neq m+1$. Note that $x_{j,m+1}$ can not be deleted in the upper part of the derivation but is removed in the last step of the derivation. Note also that the derivation is *not* tree-like: the same clause $C_{m+1}(j)$ is used many times

Note that the first (top) row corresponds to our initial CNF formula GT_n, the second to GT_{n-1}, and so on. For each m, clauses $C_m(1), \ldots, C_m(m)$ are obtained in parallel. Each $C_m(j)$ is obtained using the clauses $C_{m+1}(j)$ and $C_{m+1}(m+1)$ derived in the previous step, and the initial clauses (axioms) $A(1,m+1,j), A(2,m+1,j), \ldots, A(m,m+1,j)$ and $B(m+1,j)$ (see Fig. 18.4). At the end we easily derive the empty clause from $C_2(1), C_2(2)$ and $B(2,1)$. □

Remark 18.7. Alekhnovich et al. (2007) proved that an appropriate modification GT'_n of GT_n requires regular refutations of exponential size, but has (non-regular) resolution refutations of polynomial size. More precisely, if S denotes the smallest size of a non-regular resolution refutation of GT'_n, and R the smallest size of a regular resolution refutation of GT'_n, then $\log R = \Omega(\sqrt[3]{S})$. Using different CNF formulas, Urquhart (2011) obtained even larger gap: $\log R = \Omega(S/\text{polylog}(S))$. Thus, we have the following separations, where "$A \ll B$" stands for "proof system A is exponentially weaker that B":

$$\text{tree-like resolution} \ll \text{regular resolution} \ll \text{general resolution}.$$

18.5 Lower Bounds for General Resolution

General, non-tree-like resolution proofs are much harder to analyze. The first exponential lower bound for the size of such proofs was proved by Haken (1985).

Theorem 18.8. (Haken 1985) *Any resolution refutation proof of* PHP^n_{n-1} *requires size* $2^{\Omega(n)}$.

Proof. (Due to Beame and Pitassi 1996) The proof is by contradiction. We define an appropriate notion of a "fat" clause and show two things:

1. If PHP^n_{n-1} has a short resolution proof, then it is possible to set some variables to constants so that the resulting proof is a refutation of PHP^m_{m-1} for a large enough m, and has no fat clauses.
2. If m is large enough, then every refutation proof for PHP^m_{m-1} must have at least one fat clause.

This implies that PHP^n_{n-1} cannot have short resolution proofs.

In the case of the CNF formula PHP^n_{n-1} truth assignments α are n by $n-1$ boolean matrices. We say that a truth assignment α is *i-critical* if

- The i-th row of α is the only all-0 row, and
- Every column has exactly one 1.

Note that each such assignment α is "barely unsatisfying": it satisfies all hole axioms $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ as well as the axioms of all but the i-th pigeon. That is, the only axiom it falsifies is the pigeon axiom $C_i = x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n-1}$. Thus an i-critical assignment corresponds to an assignment of pigeons to holes such that $n-1$ of the pigeons are mapped to $n-1$ different holes, but the i-th pigeon is mapped to no hole at all. Call an assignment α *critical* if it is i-critical for some $1 \leq i \leq m$.

The properties of critical truth assignments make it convenient to convert each clause C to a positive clause C^+ which is satisfied by precisely the same set of critical assignments as C. More precisely to produce C^+, we replace each negated literal $\neg x_{i,j}$ with the OR of all variables in the j-th column, except the i-th one:

$$X_{i,j} = x_{1,j} \vee \cdots \vee x_{i-1,j} \vee x_{i+1,j} \vee \cdots \vee x_{n,j}.$$

Note that the monotone version C^+ of every hole axiom $C = \neg x_{i_1,j} \vee \neg x_{i_2,j}$ is just the OR of *all* variables in the j-th column, and hence, is satisfied by *any* critical assignment.

Claim 18.9. For every critical truth assignment α, $C^+(\alpha) = C(\alpha)$.

Proof. Suppose there is a critical assignment α such that $C^+(\alpha) \neq C(\alpha)$. This could only happen if C contains a literal $\neg x_{i,j}$ such that $\neg x_{i,j}(\alpha) \neq X_{i,j}(\alpha)$. But this is impossible, since α has precisely one 1 in the j-th column. □

Associate with each clause in a refutation of PHP^n_{n-1} the set

$$\text{Pigeon}(C) = \{i / \text{there is some } i\text{-critical assignment } \alpha \text{ such that } C(\alpha) = 0\}$$

of pigeons that are "bad" for this clause: some critical assignment of these pigeons falsifies C.

The *width*, $w(C)$, of a clause is the number of literals in it.

Claim 18.10. Every resolution refutation of PHP^n_{n-1} must have a clause C such that $w(C^+) \geq n^2/9$.

Proof. Define the *weight* of a clause C as $\mu(C) := |\mathrm{Pigeon}(C)|$. By the definition, each hole axiom has weight 0, each pigeon axiom has weight 1, and the last (empty) clause has weight n since it is falsified by any truth assignment. Moreover, this weight measure is *subadditive*: if a clause C is derived from clauses A and B, then $\mu(C) \leq \mu(A) + \mu(B)$. This holds because every assignment (even a non-critical one) falsifying C must falsify at least one of the clauses A and B. Therefore, if C is the first clause in the proof[2] with $\mu(C) > n/3$, we must have

$$n/3 < \mu(C) \leq 2n/3. \tag{18.4}$$

Fix such a "medium heavy" clause C and let $s = \mu(C)$ be its weight. Since $n/3 < s \leq 2n/3$, it is enough to show that the positive version C^+ of this clause must have $w(C^+) \geq s(n-s)$ distinct variables.

Fix some $i \in \mathrm{Pigeon}(C)$ and let α be an i-critical truth assignment with $C(\alpha) = 0$. For each $j \notin \mathrm{Pigeon}(C)$, define the j-critical assignment α', obtained from α by toggling rows i and j. That is, if α maps the j-th pigeon to the k-th hole, then α' maps the i-th pigeon to this hole (see Fig. 18.5).

Now $C(\alpha') = 1$ since $j \notin \mathrm{Pigeon}(C)$. By Claim 18.9, we have that $C^+(\alpha) = 0$ and $C^+(\alpha') = 1$. Since the assignments α,α' differ only in the variables $x_{i,k}$ and $x_{j,k}$, this can only happen when C^+ contains the variable $x_{i,k}$.

Running the same argument over all $n - s$ pigeons $j \notin \mathrm{Pigeon}(C)$ (using the same α), it follows that C^+ must contain at least $n - s$ of the variables $x_{i,1}, \ldots, x_{i,n-1}$ corresponding to the i-th pigeon. Repeating the argument for all pigeons $i \in \mathrm{Pigeon}(C)$ shows that C^+ contains at least $s(n-s)$ variables, as claimed. □

We can now finish the proof of Theorem 18.8 as follows. Let $\mathcal{R}$ be a resolution refutation proof of PHP^n_{n-1}. Let a and $b \geq 2$ be positive constants (to be specified

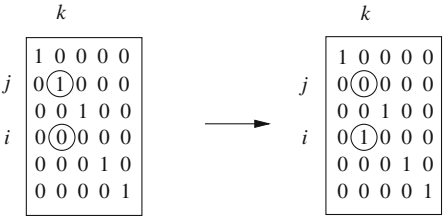

Fig. 18.5 Assignment α' is obtained from α by interchanging the i-th and j-th rows

[2] Recall that a proof is a *sequence* of clauses. Alternatively, one can apply Lemma 1.3.

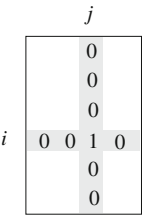

Fig. 18.6 Setting of constants to eliminate clauses containing $x_{i,j}$; non-shaded positions are not set. In this way PHP^n_{n-1} is reduced to PHP^{n-1}_{n-2}

later). For the sake of contradiction, assume that

$$|\mathcal{R}| < \mathrm{e}^{n/a}.$$

Together with $\mathcal{R}$ we consider the set $\mathcal{R}^+ = \{C^+ \mid C \in \mathcal{R}\}$ of positive versions of clauses in $\mathcal{R}$. Say that a clause is *fat* if it contains at least n^2/b variables. Let S be the total number of fat clauses in $\mathcal{R}^+$. Since each fat clause has at least a $1/b$ fraction of all the variables, there must be (by the pigeonhole principle!) a variable $x_{i,j}$ which occurs in at least S/b of fat clauses in $\mathcal{R}^+$.

Set this "popular" variable to 1, and at the same time set to 0 all the variables $x_{i,j'}$ and $x_{i',j}$ for all $j' \neq j, i' \neq i$ (see Fig. 18.6). After this setting, all the clauses containing $x_{i,j}$ will disappear from $\mathcal{R}^+$ (they all get the value 1) and the variables which are set to 0 will disappear from the remaining clauses.

Applying this restriction to the entire proof $\mathcal{R}$ leaves us with a refutation proof $\mathcal{R}_1$ for PHP^{n-1}_{n-2}, where the number of fat clauses in $\mathcal{R}_1^+$ is at most $S(1-1/b)$. Applying this argument iteratively $d = b \ln S < (b/a)n$ times, we are guaranteed to have knocked out all fat clauses, because

$$S(1-1/b)^d < \mathrm{e}^{\ln S - d/b} = 1.$$

Thus we are left with a refutation proof for PHP^m_{m-1}, where

$$m = n - d \geq (1 - b/a)n,$$

and where $w(C^+) < n^2/b$ for *all* its clauses. But Claim 18.10 implies that any refutation proof of PHP^m_{m-1} must contain a clause C for which

$$n^2/b > w(C^+) \geq m^2/9 \geq (1 - b/a)^2 n^2/9.$$

To get the desired contradiction, it is enough to choose the parameters a and b so that $(1-b/a)^2 \geq 9/b$ which, in particular, is the case for $b = 16$ and $a = 4b = 64$. □

The reader may wonder: where in this proof did we used the fact that the clauses in a refutation are derived using only resolution and weakening rules? The same argument seems to work for more general derivations. And this is indeed the case: the only important thing was that the formulas in such a derivation are clauses—this allowed us to kill off a clause by setting just one variable to a constant.

Actually, a closer look at the proof shows that it also works for a more general derivation rule, called *semantic derivation rule*. This rule allows to derive a clause C from clauses $C_1, \ldots, C_k$ if these clauses "semantically imply" C in the following sense: for all $\alpha \in \{0,1\}^n$,

$$C_1(\alpha) = 1, \ldots, C_k(\alpha) = 1 \quad \text{implies} \quad C_j(\alpha) = 1.$$

A *semantic proof* of an unsatisfiable CNF F is a sequence $\mathcal{R} = (C_1, \ldots, C_t)$ of clauses such that $C_t = 0$ is the empty clause and each C_j is either an axiom (belongs to F) or is obtained from k or fewer previous clauses (already derived or belonging to F) by one application of the semantic rule.

The only difference is that now instead of (18.4) we will have (cf. Lemma 1.3):

$$\frac{n}{k+1} < \mu(C) \leq \frac{kn}{k+1},$$

which results in a lower bound $w(C^+) \geq n^2/(k+1)^2$ in Claim 18.10. The rest is the same with constants $b := (k+1)^2$ and $a := (k+1)^3$. The resulting lower bound is then $e^{n/(k+1)^2}$, which is super-polynomial as long as $k \leq \sqrt{n}/\log n$.

18.6 Size Versus Width

We have already seen that "fat" clauses—those whose width (number of literals) exceeds some given threshold value—play a crucial role in trying to show that the size of a resolution proof (= the total number of lines in it) must be large. We are now going to show that this is a general phenomenon, not just an accident: if any resolution proof for an unsatisfiable CNF formula F must contain at least one fat clause, then F cannot have a short resolution proof.

The width of a clause C is just the number of literals in it. If F is a set of clauses then its width $w(F)$ is the maximum width of its clause. Recall that each resolution refutation $\mathcal{R}$ is also a set (more precisely, a sequence) of clauses. Hence, the width of a refutation is also the maximum width of a clause participating in it.

Now let F be an unsatisfiable CNF of n variables. Define its *resolution refutation width* $w_R(F)$ as the minimum width of a resolution refutation of F. The *resolution refutation size* $S_R(F)$ is, as before, the minimum number of clauses in a resolution refutation of F. That is,

$$w_R(F) = \min\{w(\mathcal{R}) : \mathcal{R} \text{ is a resolution refutation proof of } F\}$$

and

$$S_R(F) = \min\{|\mathcal{R}| : \mathcal{R} \text{ is a resolution refutation proof of } F\}.$$

Note that refutation proofs $\mathcal{R}$ achieving $w_R(F)$ and $S_R(F)$ may be different! Let also $S_T(F)$ denote the minimum number of clauses in a *tree-like* resolution refutation of F.

What is the relation between these parameters? If we use all clauses of the CNF F in its refutation, then $w_R(F) \geq w(F)$. But this is not true in general: it may happen that not all clauses of F are used in the refutation of F.

The relation $S_R(F) \leq (2n+1)^{w_R(F)}$ between proof-size and proof-width is easy to see: since we only have $2n$ literals, the number of all possible clauses of width k does not exceed $(2n+1)^k$. Much more interesting is the following *lower* bound on proof-size in terms of proof-width: only CNF formulas having narrow proofs can be proved in a short time!

Theorem 18.11. (Ben-Sasson and Wigderson 2001) *For any unsatisfiable k-CNF formula F of n variables,*

$$\log S_R(F) \geq \frac{(w_R(F)-k)^2}{16n} \tag{18.5}$$

and

$$\log S_T(F) \geq w_R(F) - k. \tag{18.6}$$

For the proof of this theorem we need a concept of a *restriction* of CNFs and of refutation proofs. Let F be some set of clauses (think of F as a CNF or as a refutation proof). Let x be some of its literals. If we set this literal to 0 and to 1, then we obtain two sets of clauses:

- $F_{x=0}$ is F with all clauses containing $\neg x$ removed from F (they get value 1) and literal x removed from all the remaining clauses of F (it gets value 0);
- $F_{x=1}$ is F with all clauses containing x removed from F and literal $\neg x$ removed from all the remaining clauses of F.

Note that, if F was an unsatisfiable CNF, then both CNFs $F_{x=0}$ and $F_{x=1}$ remain unsatisfiable. Moreover, if $\mathcal{R}$ was a resolution refutation proof of F and $a \in \{0,1\}$, then $\mathcal{R}_{x=a}$ is also a resolution refutation proof of $F_{x=a}$. Indeed, if at some step in $\mathcal{R}$ a literal x is resolved using the resolution rule, then this step in $\mathcal{R}_{x=a}$ corresponds to an application of the weakening rule:

$$\frac{A \vee x \quad B \vee \neg x}{A \vee B} \quad \mapsto \quad \frac{A}{A \vee B} \quad \text{or} \quad \frac{B}{A \vee B}.$$

Lemma 18.12. *Let F be an unsatisfiable k-CNF formula. If $w_R(F_{x=1}) \leq w-1$ and $w_R(F_{x=0}) \leq w$, then $w_R(F) \leq \max\{w,k\}$.*

Proof. The idea is to combine refutations for $F_{x=1}$ and for $F_{x=0}$ into one refutation proof for F. First we can deduce $\neg x$ from $F_{x=1}$ using clauses of width at most w. To do this, follow closely the deduction of the empty clause from $F_{x=1}$, which

uses clauses of width at most $w-1$, and add the literal $\neg x$ to every clause in that deduction. Let $\mathcal{R}$ be the resulting deduction of $\neg x$ from $F_{x=1}$. Now, from $\neg x$ and F we can deduce $F_{x=0}$ by using the resolution rule: just resolve $\neg x$ with each clause of F containing x to get $F_{x=0}$. This step does not introduce any clauses of width more than k. Finally, deduce the empty clause from $F_{x=0}$ using clauses of width at most w. □

Now let W be a parameter (to be specified later), and call a clause *fat* if it has width larger than W. Set also

$$a := \left(1 - \frac{W}{2n}\right)^{-1} \geq e^{W/2n}.$$

Lemma 18.13. *If a k-CNF F has a refutation that contains fewer than a^b fat clauses then $w_R(F) \leq W + b + k$.*

Proof. We prove this by induction on b and n. The base case $b = 0$ is trivial, since then we have no fat clauses at all implying that $w_R(F) \leq \max\{W, k\} \leq W + k$.

Now assume that the claim holds for all smaller values of n and b. Take a resolution refutation $\mathcal{R}$ of F using $< a^b$ fat clauses. Since there are at most $2n$ literals and any fat clause contains at least W of them, an average literal must occur in at least a $W/2n$ fraction of fat clauses. Choose a literal x that occurs most frequently in fat clauses and set it to 1. This way we kill off (evaluate to 1) all clauses containing x. The obtained refutation $\mathcal{R}_{x=1}$ of $F_{x=1}$ has fewer than $a^b\left(1 - \frac{W}{2n}\right) = a^{b-1}$ fat clauses. By induction on b we have $w_R(F_{x=1}) \leq W + (b-1) + k$. On the other hand, since $F_{x=0}$ has one variable fewer, induction on n yields $w_R(F_{x=0}) \leq W + b + k$. The desired upper bound $w_R(F) \leq W + b + k$ now follows from Lemma 18.12. □

Proof of Theorem 18.11. Choose b so that $a^b = S_R(F)$. Then

$$b = \frac{\log S_R(F)}{\log a} \leq \frac{2n \log S_R(F)}{W \log(e)} \leq \frac{4n \log S_R(F)}{W}$$

and, by Lemma 18.13,

$$w_R(F) \leq W + \frac{4n \log S_R(F)}{W} + k.$$

Choosing $W := 2\sqrt{n \log S_R(F)}$ to minimize the right-hand side yields the desired upper bound $w_R(F) \leq 4\sqrt{n \log S_R(F)} + k$. This finishes the proof of (18.5). We leave the proof of (18.6) as an exercise; hint: as the literal x to be set take the *last* literal which is resolved to get the empty clause. □

Remark 18.14. That Theorem 18.11 cannot be substantially improved was shown by Bonet and Galesi (1999): there are unsatisfiable k-CNF formulas F (k being a constant) such that $S_R(F) \leq n^{\mathcal{O}(1)}$ but $w_R(F) = \Omega(\sqrt{n})$.

A general frame to prove that the proof-width $w_R(F)$, and hence, the proof-size $S_R(F)$ must be large is as follows.

1. Take an arbitrary resolution refutation proof $\mathcal{R}$ for F.
2. Define some measure $\mu(C)$ of "weight" of its clauses $C \in \mathcal{R}$ such that
   a. The weight of each axiom is small;
   b. The last (empty) clause $\emptyset$ has large weight, and
   c. The measure is subadditive: $\mu(C) \leq \mu(A) + \mu(B)$ if C is a resolvent of A and B.
3. Use the subadditivity of μ to find a clause $C \in \mathcal{R}$ of "intermediate" (large, but not too large) measure $\mu(C)$.
4. Show that any clause of intermediate μ-measure must have many literals.

To achieve these goals one usually takes $\mu(C)$ to be the smallest number of axioms in a "witness" for C. A set $\mathcal{A}$ of axioms is a *witness* for C if every assignment satisfying all axioms in $\mathcal{A}$ satisfies the clause C as well. Then one argues as follows. The minimality of $\mathcal{A}$ implies that, for any axiom $A \in \mathcal{A}$, there must exist an assignment α such that $C(\alpha) = 0$ but $B(\alpha) = 1$ for all $B \in \mathcal{A}$, $B \neq A$. Now suppose that flipping the i-th bit of α gives us an assignment α' satisfying all axioms in $\mathcal{A}$. Since $\mathcal{A}$ is a witness for C, we have that $C(\alpha') = 1$. But the assignments α and α' only differ in the i-th position, implying that the i-th variable x_i or its negation must be present in C. Note that this was the way we argued in the proof of Haken's theorem for PHP_n^{n+1}.

In the next sections we show how this idea works in other situations.

18.7 Tseitin Formulas

In this section we discuss a large class of unsatisfiable CNFs whose resolution refutation proofs have large width. These CNFs formalize the basic property of graphs: in every graph, the number of vertices of *odd* degree must be *even*. This is a direct consequence of Euler's theorem stating that the sum of degrees in any graph is two times the number of edges, and hence, is even.

Let $G = (V, E)$ be a connected graph, and $f : V \to \{0, 1\}$ an assignment of bits 0 and 1 to its vertices. Let $d(v)$ denote the degree of a vertex $v \in V$. Associate with each edge $e \in E$ a boolean variable x_e. For each vertex $v \in V$, let A_v be a CNF formula with $2^{d(v)-1}$ clauses expressing the equality

$$L_v : \quad \bigoplus_{e : v \in e} x_e = f(v). \tag{18.7}$$

For example, the equality $x \oplus y \oplus z = 0$ is expressed by a CNF

$$(\neg x \vee \neg y \vee \neg z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z).$$

The *Tseitin formula*, $\tau(G, f)$, is the AND of all these CNF formulas A_v, $v \in V$. Tseitin (1970) used such formulas to prove the first exponential lower bound on the size of regular resolution.

Remark 18.15. If k is the maximal degree of G, then $\tau(G, f)$ is a k-CNF formula with at most $n2^{k-1}$ clauses and $nk/2$ variables. Thus, if the degree k is *constant*, then $\tau(G, f)$ is a k-CNF formula with $\mathcal{O}(n)$ clauses and $\mathcal{O}(n)$ variables.

The meaning of Tseitin's formulas is the following. The function f "charges" some of the vertices, that is, gives them value 1. Each assignment α of constants 0 and 1 to the variables x_e defines a subgraph G_α of G. Such an assignment α satisfies $\tau(G, f)$ if and only if exactly the charged vertices have odd degrees in the subgraph G_α.

It is not difficult to show that if we charge an *odd* number of vertices, that is, if $\bigoplus_{v \in V} f(v) = 1$, then $\tau(G, f)$ is not satisfiable. Indeed, otherwise the graph G would have a subgraph in which an odd number of vertices (the charged ones) have odd degree, contradicting the Euler theorem. Interestingly, the converse also holds.

Lemma 18.16. (Tseitin 1970) *For a connected graph $G = (V, E)$, the CNF $\tau(G, f)$ is satisfiable if and only if an even number of vertices are charged by f.*

Proof. Assume first that f charges an *odd* number of vertices. We have used Euler's theorem to show that then $\tau(G, f)$ is unsatisfiable. This can also be shown directly. Observe that each variable x_e with $e = \{u, v\}$ appears in exactly two equations L_u and L_v. Hence, if we sum (modulo 2) all equations in (18.7), the left hand side will be equal to 0, whereas the right hand side will be 1, a contradiction. Hence, in this case the system (18.7) is not satisfiable.

Now assume that f charges an *even* number of vertices. We have to show that then $\tau(G, f)$ is satisfiable. For this, we make the following simple observation. Now start with an all-0 assignment α. If α satisfies all equalities L_v, we are done. If α does not satisfy all equalities, then the number of unsatisfied equalities must be even (the number of vertices v with $f(v) = 1$ must be even). We take any two vertices u, v with unsatisfied equalities L_v, L_u and change all bits of α corresponding to edges on a path from u to v (such a path must exist since G is connected). The obtained assignment α' will already satisfy L_v and L_u. Moreover, by our observation,[3] we have that $L_w(\alpha') = L_w(\alpha)$ for all vertices $w \notin \{u, v\}$. Hence, the number of unsatisfied equalities decreases by two. Proceeding in this way we will eventually reach an assignment satisfying all equalities. □

We now give a general lower bound of the resolution refutation width of unsatisfiable Tseitin formulas $\tau(G, f)$ in terms of one combinatorial characteristic of the underlying graphs $G = (V, E)$. For a subset $S \subseteq V$ of vertices, let $e(S, V \setminus S)$ denote the number of crossing edges with one endpoint lying in S and the other in

[3] Let $\alpha \in \{0, 1\}^E$ be an assignment, and e_1, e_2 two edges with a common endpoint v. Let α' be obtained by flipping the values of both variables x_{e_1} and x_{e_2}. Then $L_v(\alpha') = L_v(\alpha)$.

$V \setminus S$. Define the *edge expansion*, $\mathrm{ex}(G)$, of G as the minimum of $e(S, V \setminus S)$ over all subsets S with $n/3 \le |S| \le 2n/3$; here $n = |V|$ is the total number of vertices in G.

Theorem 18.17. (Ben-Sasson and Wigderson 2001) *Let $G = (V, E)$ be a connected graph, and $f : V \to \{0, 1\}$ satisfy $\bigoplus_{v \in V} f(v) = 1$. Then*

$$w_R(\tau(G, f)) \ge \mathrm{ex}(G).$$

Proof. Fix an arbitrary resolution refutation proof $\mathcal{R}$ for $\tau(G, f)$. Recall that axioms of this proof are CNF formulas A_v corresponding to equalities (18.7). For a subset $S \subseteq V$ of vertices, let A_S be the AND of all clauses in the sets A_v, $v \in S$. Define the measure $\mu : \mathcal{R} \to \mathbb{N}$ on clauses by:

$$\mu(C) := \min\{|S| : A_S \text{ implies } C\}.$$

If C is one of the axioms, then clearly $\mu(C) = 1$. Furthermore, μ is subadditive: $\mu(C) \le \mu(A) + \mu(B)$ if C is a resolvent of A and B.

Claim 18.18. $\mu(\emptyset) = n$.

Proof. By the definition of μ, $\mu(\emptyset)$ is exactly the smallest number $|S|$ of vertices such that A_S is unsatisfiable. So it is enough to show that A_S is satisfiable for each subset $S \subseteq V$ of size $|S| < |V|$. To show this, take any vertex $v \in V \setminus S$. Consider the function $f' : V \to \{0, 1\}$ such that $f'(v) = 1 - f(v)$ and $f'(u) = f(u)$ for all $u \ne v$. Since $\bigoplus_{v \in V} f'(v) = 0$, Lemma 18.16 implies that $\tau(G, f')$ is satisfiable. Since $v \notin S$, the CNF A_S is a part of the formula $\tau(G, f')$, and hence, is satisfiable as well. Hence, $\mu(\emptyset) = n$. □

By the subadditivity of μ, there must exist a clause $C \in \mathcal{R}$ such that $n/3 \le \mu(C) \le 2n/3$, an "intermediate clause" (see Lemma 1.3). Let $S \subseteq V$ be a minimal set for which A_S implies C; hence $n/3 \le |S| \le 2n/3$.

To finish the proof of the theorem, it is enough to show that $x_e \in C$ for every crossing edge $e = \{u, v\}$ with $u \in S$ and $v \in V \setminus S$. For the sake of contradiction, assume that $x_e \notin C$. By the minimality of S, there exists an assignment α which satisfies all axioms in A_S except those in A_u, and falsifies C. The assignment α', obtained from α by flipping the bit x_e, satisfies all axioms in A_S (because $v \notin S$), and hence, must satisfy the clause C. This is a contradiction because $C(\alpha) = 0$ and the new assignment α' still agrees with α for all variables of C. □

Theorem 18.17 gives us a whole row of unsatisfiable k-CNF formulas $F = \tau(G, f)$ of n variables such that $k = \mathcal{O}(1)$, $|F| = \mathcal{O}(n)$ and $w_R(F) = \Omega(n)$. Together with Theorem 18.11, these CNF formulas require resolution proofs of size $2^{\Omega(n)}$. For this, it is enough that the underlying graph G has constant degree k and still has large edge extension $\mathrm{ex}(G)$. The existence of such graphs can be shown by simple probabilistic arguments. There are even *explicit* graphs with these properties. Such are, for example, Ramanujan graphs considered in Sect. 5.8. By the Expander Mixing Lemma (see Appendix A) these graphs have $\mathrm{ex}(G) = \Omega(n)$.

18.8 Expanders Force Large Width

We have shown that resolution refutation proofs for Tseitin CNF formulas $\tau(G, f)$ require large width as long as the underlying graph G has good expanding properties. It turns out that a similar fact also holds for *any* unsatisfiable CNF as long as it has good expansion properties in the following sense.

Look at a CNF formula F as a *set* of its clauses. Hence, $|F|$ denotes the number of clauses in F, and $G \subseteq F$ means that the CNF G contains only clauses of F. Let $\text{var}(F)$ denote the number of variables in F.

We say a CNF formula F is (r, c)-*expanding* if

$$\text{var}(G) \geq (1 + c)|G| \text{ for every subset } G \subseteq F \text{ of its } |G| \leq r \text{ clauses.}$$

We can associate with F a bipartite graph, where nodes on the left part are clauses of F, nodes on the right part are variables, and a clause C is joined to a variable x iff x or $\neg x$ belongs to C. Then F is (r, c)-expanding iff every subset of $s \leq r$ nodes on the left part have at least $(1 + c)s$ neighbors on the right part.

Theorem 18.19. (Ben-Sasson and Wigderson 2001) *Let F be an unsatisfiable CNF formula. If F is (r, c)-expanding, then $w_R(F) \geq cr/2$.*

We first prove three claims relating the number of clauses with the number of variables in unsatisfiable CNF formulas.

Claim 18.20. If $|G| \leq \text{var}(G)$ for every $G \subseteq F$, then F is satisfiable.

Proof. We will use the well-known Hall's Marriage Theorem. It states that a family of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ has a system of distinct representatives (that is, a sequence $x_1, \ldots, x_m$ of m distinct elements such that $x_i \in S_i$) iff the union of any number $1 \leq k \leq m$ of members of $\mathcal{S}$ has at least k elements.

Now assume that $|G| \leq \text{var}(G)$ for all $G \subseteq F$. Then, by Hall's theorem, we can find for each clause C of F a variable $x_C \in \text{var}(C)$ such that x_C or its negation appears in C, and for distinct clauses these variables are also distinct. We can therefore set these variables to 0 or 1 independently to make all clauses true. Hence, F is satisfiable. □

Say that an unsatisfiable CNF formula is *minimally* unsatisfiable if removing any clause from it makes the remaining CNF satisfiable. The following claim is also known as Tarsi's Lemma.

Claim 18.21. If F is minimally unsatisfiable, then $|F| > \text{var}(F)$.

Proof. Since F is unsatisfiable, Claim 18.20 implies that there must be a subset of clauses $G \subseteq F$ such that $|G| > \text{var}(G)$. Let $G \subseteq F$ be a *maximal* subset of clauses with this property. If $G = F$ then we are done, so assume that $G \subset F$ and we will derive a contradiction.

Take an arbitrary sub-formula $H \subseteq F \setminus G$, and let $\mathrm{Vars}(H)$ be the set of its variables. Due to maximality of G, $\mathrm{Vars}(H) \setminus \mathrm{Vars}(G)$ must have at least $|H|$ variables, for otherwise we would have that $\mathrm{var}(G \cup H) < |G \cup H|$, a contradiction with the maximality of G.

Thus the CNF formula $F \setminus G$ satisfies the condition of Claim 18.20, and hence, can be satisfied by only setting constants to variables in $\mathrm{Vars}(F) \setminus \mathrm{Vars}(G)$. Since F is minimally unsatisfiable, the CNF formula G must be satisfiable using only the variables in $\mathrm{Vars}(G)$. Altogether this gives us a truth assignment satisfying the entire formula F, a contradiction. □

As before, we say that a CNF formula F *implies* a clause A if any assignment satisfying F also satisfies A. We also say that F *minimally implies* A if the CNF formula F implies A but none of its proper subformulas (obtained by removing any clause) does this.

Claim 18.22. If F minimally implies a clause A, then $|A| > \mathrm{var}(F) - |F|$.

Proof. Let $\mathrm{Vars}(F) = \{x_1, \ldots, x_n\}$ and assume that $\mathrm{Vars}(A) = \{x_1, \ldots, x_k\}$. Take a (unique) assignment $\alpha \in \{0,1\}^k$ for which $A(\alpha) = 0$. Since F implies A, restricting F to α must yield an unsatisfiable formula F_α on variables $x_{k+1}, \ldots, x_n$. The formula F_α must also be minimally unsatisfiable because F minimally implied A. By Claim 18.21, F_α must have more than $n-k$ clauses. Hence, $|F| \geq |F_\alpha| > n - k = \mathrm{var}(F) - |A|$, as desired. □

We now turn to the actual proof of the theorem.

Proof of Theorem 18.19. Let F be an (r,c)-expanding unsatisfiable CNF formula, and let $\mathcal{R}$ be any resolution refutation proof of F. We can assume that both numbers r and c are positive (otherwise there is nothing to prove). With each clause C in $\mathcal{R}$ associate the number

$$\mu(A) = \min\{|G| : G \subseteq F \text{ and } G \text{ implies } A\}.$$

It is clear that $\mu(A) \leq 1$ for all clauses A of F. Furthermore, μ is subadditive: $\mu(C) \leq \mu(A) + \mu(B)$ if C is a resolvent of A and B. Finally, the expansion property of F implies that $\mu(0) > r$. Indeed, by the definition, $\mu(0)$ is the smallest size $|G|$ of an unsatisfiable subformula $G \subseteq F$, and Claim 18.21 yields $|G| > \mathrm{var}(G)$. Had we $\mu(0) \leq r$, then we would also have $|G| \leq r$ and the expansion property of F would imply $\mathrm{var}(G) \geq (1+c)|G|$, a contradiction.

Hence, the subadditivity of μ implies that the refutation $\mathcal{R}$ of F must contain a clause C such that $r/2 \leq \mu(C) < r$ (cf. Lemma 1.3). Fix some $G \subseteq F$ minimally implying C; hence, $r/2 \leq |G| = \mu(C) < r$. By the expansion of F, $\mathrm{var}(G) \geq (1+c)|G|$. Together with Claim 18.22 this implies $|C| > \mathrm{var}(G) - |G| \geq c|G| \geq cr/2$, as desired. □

18.9 Matching Principles for Graphs

We already know (see Theorem 18.8) that the pigeonhole principle PHP_n^m requires resolution proof of exponential size, as long as the number m of pigeons is $m = n + 1$, where n is the number of holes. However, the larger m is, the more true the pigeonhole principle itself is, and it could be that PHP_n^m with larger number m pigeons could be refuted by much shorter resolution refutation proof. We now will use expander graphs to prove that PHP_n^m has no resolution proofs of polynomial size even if we have up to $m = n^{2-o(1)}$ pigeons.

Given a bipartite $m \times n$ graph $G = ([m], [n], E)$, we may consider the CNF formula $\mathrm{PHP}(G)$ which is an AND of the following set of axioms:

- Pigeon Axioms: $C_i = \bigvee_{(i,j)\in E} x_{i,j}$ for $i = 1, \ldots, m$.
- Hole Axioms: $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ for $i_1 \neq i_2 \in [m]$ and $j \in [n]$.

That is, the graph dictates what holes are offered to each pigeon, whereas hole axioms forbid (as in the case of PHP_n^m) that two pigeons sit in one hole.

Observe that, if $m > n$ and if the graph G has no isolated vertices, then the CNF formula $\mathrm{PHP}(G)$ is unsatisfiable. Indeed, every truth assignment α defines a subgraph G_α of G. Now, if α satisfies all hole axioms then G_α must be a (possibly empty) matching. But we have $m > n$ vertices of the left side. Hence, at least one of these vertices $i \in [m]$ must remain unmatched in G_α, implying that $C_i(\alpha) = 0$.

Observe also that $\mathrm{PHP}_n^m = \mathrm{PHP}(K_{m,n})$ where $K_{m,n}$ is a complete bipartite $m \times n$ graph. Moreover, if G' is a subgraph of G, then every resolution refutation for $\mathrm{PHP}(G)$ can be turned to a resolution refutation of $\mathrm{PHP}(G')$ just by setting to 0 all variables corresponding to edges of G that are not present in G'. Thus to prove a lower bound of the resolution complexity of $\mathrm{PHP}(G)$ it is enough to prove such a bound for *any* subgraph of G.

This opens plenty of possibilities to prove large lower bounds for PHP_n^m: just show that there *exists* a graph G (a subgraph of $K_{m,n}$) such that $\mathrm{PHP}(G)$ requires long resolution refutation proofs. By Theorems 18.11 and 18.19, this can be done by showing that the CNF formula $F = \mathrm{PHP}(G)$ has large expansion. This, in turn, can be achieved if the underlying graph G itself has good expansion properties.

A bipartite graph is an (r, c)-*expander* if every set of $k \leq r$ vertices on the left part has at least $(1 + c)k$ neighbors on the right part. It can be easily shown (Exercise 18.4) that if G is an (r, c)-expander then the CNF formula $\mathrm{PHP}(G)$ is (r, c)-expanding.

Using a probabilistic argument it can be shown that (r, c)-expanders with $c > 0$, $r = \Omega(n)$ and *constant* left-degree exist (Exercise 18.5). Hence, the CNF formula $F = \mathrm{PHP}(G)$ has $N = \mathcal{O}(m)$ variables and each its clause has constant width. Theorem 18.19 implies that $w_R(F) = \Omega(n)$. So by Theorem 18.11, every resolution refutation for F, and hence, for PHP_n^m must have size exponential in $w_R(F)^2/N = \Omega(n^2/m)$.

This gives super-polynomial lower bound on the size of resolution refutations of PHP_n^m for up to $m \ll n^2/\log n$ pigeons. In Sect. 18.3 we have proved that, no matter

how large the number $m > n$ of pigeons is, any *tree-like* resolution refutation proof of PHP_n^m must have size $n^{\Omega(n)}$. But all attempts to overcome the "n^2 barrier" for the number of pigeons m in the case of general (not just tree-like) resolution proofs failed for many years. This was one of the most famous open problems concerning resolution proofs.

The "n^2 barrier" was finally broken by Raz (2001). He proved that, for any number $m > n$ of pigeons, the CNF PHP_n^m requires general (non-tree-like) resolution proofs of exponential size. Shortly after, Razborov (2003) found a simpler proof.

Exercises

18.1. Show that Resolution is complete: every unsatisfiable CNF formula F has a resolution refutation proof.

Hint: Show that the search problem for F can be solved by a decision tree, and use Theorem 18.1.

18.2. Show that Theorem 18.5 remains true if instead of CNF formula PHP_n^m we take its *functional* version by adding new axioms $\neg x_{i,j_1} \vee \neg x_{i,j_2}$ for all $j_1 \neq j_2$ and $i = 1, \ldots, m$. These axioms claim that no pigeon can sit in two holes.

18.3. Let F be a CNF formula and x a literal. Show that F is unsatisfiable if and only if both CNFs $F_{x=1}$ and $F_{x=0}$ are unsatisfiable.

18.4. Let G be a bipartite (r, c)-expander graph. Show that then the induced CNF formula $\text{PHP}(G)$ is (r, c)-expanding.

18.5. Show that for every constant $d \geq 5$, there exist bipartite $n \times n$ graphs of left degree d that are (r, c)-expanders for $r = n/d$ and $c = d/4 - 1$.

Hint: Construct a random graph with parts L and R, $|L| = |R| = n$, by choosing d neighbors for each vertex in L. For $S \subseteq L$ and $T \subseteq R$, let $E_{S,T}$ be the event that all neighbors of S lie within T. Argue that, $\text{Prob}[E_{S,T}] = (|T|/n)^{d|S|}$. Let E be the event that the graph is not the desired expander, i.e., that all neighbors of some subset $S \subseteq L$ of size $|S| \leq n/d$ lie within some subset $T \subseteq R$ of size $|T| < (d/4)|S|$. Use the union bound for probabilities and the estimate $\binom{n}{k} \leq (en/k)^k$ to show that $\text{Prob}[E] \leq \sum_{i=1}^{n/d} \left(\frac{e}{4}\right)^{id/2}$. Use our assumption $d \geq 5$ together with the fact that $\sum_{i=0}^{\infty} x^i = 1/(1-x)$ for any real number x with $|x| < 1$ to conclude that $\text{Prob}[E]$ is strictly smaller than 1.

18.6. Given an unsatisfiable set F of clauses, define its *boundary* ∂F to be the set of variables appearing in exactly one clause of F. Let also

$$s(F) = \min\{|G| : G \subseteq F \text{ and } G \text{ is unsatisfiable}\}.$$

Define the *expansion* of F by

$$e(F) = \max_{s \leq s(F)} \min\{|\partial G| : G \subseteq F,\ s/2 \leq |G| < s\}.$$

Prove that, for every unsatisfiable CNF F, $w_R(F) \geq e(F)$.

Hint: Take a resolution refutation proof $\mathcal{R}$ for F. Define the *witness* of a clause C in the proof to be the set $G \subseteq F$ of all those clauses in F that are used by the proof to derive C. Show that the clause C can have at most $|\partial G|$ literals (if a literal appears in an axiom $A \in F$, then the only way it can be removed from clauses derived using A is if the literal is resolved with its negation). Then, define $\mu(C)$ to be the number $|G|$ of clauses in the witness G of C in the proof. Show that: $\mu(\emptyset) \geq s(F)$, and $\mu(C) = 1$ for any clause C in F, and $\mu(C) \leq \mu(A) + \mu(B)$ if C is a resolvent of A and B.

18.7. (2-satisfiable CNFs) A CNF formula F is *k-satisfiable* if any subset of its k clauses is satisfiable. Prove the Lieberher-Specker result for 2-satisfiable CNF formulas: if F is a 2-satisfiable CNF formula then at least γ-fraction of its clauses are simultaneously satisfiable, where $\gamma = (\sqrt{5} - 1)/2 > 0.618$.

Hint: Define the probability of a literal y to be satisfied to be: a ($a > 1/2$) if y occurs in a unary clause, and $1/2$ otherwise. Observe that then the probability that a clause C is satisfied is a if C is a unary clause, and at least $1 - a^2$ otherwise (at worst, a clause will be a disjunction of two literals whose negations appear as unary clauses); verify that $a = 1 - a^2$ for $a = \gamma$.

18.8. (3-satisfiable CNFs) Given a 3-satisfiable CNF formula F of n variables, define a random assignment $\alpha = (\alpha_1, \ldots, \alpha_n) \in \{0, 1\}^n$ by the following rule:

$$\text{Prob}[\alpha_i = 1] = \begin{cases} 2/3 \text{ if } F \text{ contains a unary clause } (x_i); \\ 1/3 \text{ if } F \text{ contains a unary clause } (\neg x_i); \\ 1/2 \text{ otherwise.} \end{cases}$$

1. Why is this definition consistent?

   Hint: 3-satisfiability.
2. Show that $\text{Prob}[y(\alpha) = 1] \geq 1/3$ for each literal $y \in \{x_i, \neg x_i\}$, which appears in the formula F (independent of whether this literal forms a unary clause or not).
3. Show that the expected number of clauses of F satisfied by α is at least a $2/3$ fraction of all clauses.

   Hint: Show that each clause if satisfied by α with probability at least $2/3$. The only nontrivial case is when the clause has exactly two literals. Treat this case by keeping in mind that our formula is 3-satisfiable, and hence, cannot have three clauses of the form $(y \vee z)$, $(\neg y)$ and $(\neg z)$.

18.9. (Due to Hirsch 2000) Suppose we have a CNF formula F of n variables with *is* satisfiable. Our goal is to find a satisfying assignment. Consider the following randomized algorithm: pick an initial assignment $\alpha \in \{0, 1\}^n$ uniformly at random, and flip its bits one by one trying to satisfy all clauses. At each step, the decision on what bit of a current assignment α to flip is also random one. The algorithm first constructs a set $I \subseteq [n]$ of bits such that flipping any bit $i \in I$ increases the number of satisfied clauses. Then it chooses one of these bits at random, and flips it. If $I = \emptyset$, then the algorithm chooses one bit at random from the set of bits that do not lead to the decrease of the number of satisfied clauses. If all variables lead to such a decrease, it chooses at random a bit from $[n]$. The algorithm works in iterations, one iteration being a random choice of an initial assignment α. We are

interested in how many iterations are needed to find a satisfying assignment with a constant probability.

Consider the CNF formula F which is an AND of two CNFs G and H. The first CNF G consists of $n+1$ clauses:

$$\neg x_1 \vee x_2, \quad \neg x_2 \vee x_3, \quad \ldots, \quad \neg x_n \vee x_1 \quad \text{and} \quad \neg x_1 \vee \neg x_2.$$

The first n clauses express that in every satisfying assignment for G the values of all its bit must be equal. The last clause of G ensures that all these values must be equal to 0. Hence, $\alpha = \mathbf{0}$ is the only assignment satisfying all the $n+1$ clauses of G. The second CNF H consists of all $n\binom{n-1}{2}$ clauses of the form $\neg x_i \vee x_j \vee x_k$ with $i \neq j \neq k$. Hence, $\alpha = \mathbf{0}$ is the unique satisfying assignment for the entire CNF $F = G \wedge H$. The clauses in H are intended for "misleading" the algorithm. Prove that, regardless of how long one iteration tends, at least $2^{\Omega(n)}$ iterations are necessary for the CNF formula F.

Hint: Show that, if c is a sufficiently large constant, then assignments α with $t := n/3 + c$ or more ones form an "insurmountable ring" around the (unique) satisfying assignment $\mathbf{0}$. Namely, if the algorithm encounters an assignment with this number of ones, then it chooses a wrong bit for flipping. That is, on such assignments α the algorithms flips some 0-bit to 1-bit, and hence, goes away from the satisfying assignment $\mathbf{0}$. When showing this, it is only important that $(k-1)(n-k-1) > \binom{n-k}{2} + 2$ holds for all $k \geq t$.