The Lovász Local Lemma and Satisfiability*

Heidi Gebauer, Robin A. Moser, Dominik Scheder, and Emo Welzl

Institute of Theoretical Computer Science, ETH Zürich
CH-8092 Zürich, Switzerland

Abstract. We consider boolean formulas in conjunctive normal form (CNF). If all clauses are large, it needs many clauses to obtain an unsatisfiable formula; moreover, these clauses have to interleave. We review quantitative results for the amount of interleaving required, many of which rely on the Lovász Local Lemma, a probabilistic lemma with many applications in combinatorics.

In positive terms, we are interested in simple combinatorial conditions which guarantee for a CNF formula to be satisfiable. The criteria obtained are nontrivial in the sense that even though they are easy to check, it is by far not obvious how to compute a satisfying assignment efficiently in case the conditions are fulfilled; until recently, it was not known how to do so. It is also remarkable that while deciding satisfiability is trivial for formulas that satisfy the conditions, a slightest relaxation of the conditions leads us into the territory of NP-completeness.

Several open problems remain, some of which we mention in the concluding section.

1 Introduction

SAT, the problem of deciding whether a boolean formula in conjunctive normal form (CNF) is satisfiable by a truth assignment, is the classical NP-complete problem. Such a *CNF formula* is obtained as a conjunction of clauses, where a *clause* is the disjunction of literals, with a *literal* either a boolean variable or its negation; we require that variables in a clause do not repeat (neither with the same nor complementary signs). A CNF formula is *satisfiable* if there is a true-false assignment to the variables so that every clause has at least one literal that evaluates to true. Consider e.g.

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4}),$$

a 3-CNF formula with 5 clauses over the variables $\{x_1, x_2, x_3, x_4\}$ (for k a nonnegative integer, a *k-CNF formula* is a CNF formula where every clause contains *exactly* k literals). This formula is satisfiable, e.g. by the assignment $(x_1, x_2, x_3, x_4) \mapsto (\text{true}, \text{true}, \text{false}, \text{true})$. But, actually, it can be recognised as satisfiable even without any close inspection, simply because 5, the number of clauses, is less than 2^3. This is because a simple probabilistic argument for instance demonstrates that

* Research is supported by SNF Grant 200021-118001/1.

S. Albers, H. Alt, and S. Näher (Eds.): Festschrift Mehlhorn, LNCS 5760, pp. 30–54, 2009.

it needs at least 2^k clauses to construct an unsatisfiable k-CNF formula.

For, suppose that some k-CNF formula has fewer than 2^k clauses, then an assignment sampled uniformly at random violates each clause with probability 2^{-k} and, by linearity of expectation, the expected total number of violated clauses is then smaller than 1, implying that some of the assignments have to satisfy the whole formula. While this result may reveal some of the beauty of probabilistic reasoning (cf. [1]), it is not very striking in its own. But let us discover that it can be extended to yield something much more powerful.

The statement becomes miraculous as soon as we observe that the constraint on the formula size need not only be satisfied globally but even *locally*. What do we mean by global and local? Suppose you have a formula of arbitrary size. Now pick any of its clauses, say C. We will say that the *neighbourhood of* C, denoted by $\Gamma(C)$, is the set of clauses that share variables with C. These are, in a sense, those clauses that relate to C, since, if we have C violated by some given assignment and change some values of variables within it to remedy that problem, then the clauses in $\Gamma(C)$ are exactly the ones we might harm. Now our intuition suggests the following: if we can change values in a clause C without causing too much damage in its surroundings and if this local property holds everywhere, then most probably we can find a globally satisfying assignment by just moving around violation issues until they disappear. And this intuition proves to be absolutely correct. In order to construct an unsatisfiable k-CNF formula, not only do we need at least 2^k clauses in total, but those clauses need to be, at least somewhere in the formula, concentrated densely around some clause. For one can prove the following:

If every clause in a k-CNF formula, $k \geq 1$, has a neighbourhood of size at most $2^k/e - 1$, then the whole formula admits a satisfying assignment.

This statement is known as the *Lovász Local Lemma* from 1975 ([2], cf. [1]), formulated in terms of satisfiability. Before we present two proofs in the next section, let us discuss other variants of the theme

"In an unsatisfiable CNF formula clauses have to interleave –
the larger the clauses, the more interleaving is required."

First, it is clear that clauses sharing variables of the same sign will not get us in major trouble in a search for a satisfying assignment. To reflect this, we define the *conflict-neighbourhood of a clause* C in a CNF formula as the set of clauses which share variables with C, at least one with opposite sign. The so-called *lopsided Local Lemma* shows that the above mentioned condition for neighbourhoods holds actually for conflict-neigbourhoods. As an aside we cannot resist mentioning the fact that if each pair of clauses in a CNF formula either has no conflict or a conflict along at least two variables, then this formula is satisfiable, unless it contains the empty clause. For a reader familiar with resolution, the mystery can be resolved instantaneously: Try resolution!

Second, it is easily seen that a clause with a large neighbourhood requires some variable to occur often in a formula. To make this precise, we call the

number of occurrences of a variable x (with either sign) in a CNF formula the *degree of* x. Then we have:

If every variable in a k-CNF formula, $k \geq 1$, has degree at most $2^k/(ek)$, then the formula is satisfiable.

Is $2^k/(ek)$ tight? While we do not believe this to be true, one can show that it cannot be increased by more than a constant factor. This holds also for the previously mentioned bounds for the (conflict-)neighbourhood size, but while this is certified by the simple example of a k-CNF formula that contains all possible 2^k clauses over a given set of k variables, the degree bound requires a more elaborate construction and therefore this had been open for some time.

Third, what can be said if we constrain the quality of interleaving rather than the quantity? For this we consider *linear*[1] *CNF formulas*, i.e. CNF formulas where any two clauses share at most one variable. Here is an example of a linear 2-CNF formula:

$$(\overline{y_1} \vee \overline{y_2}) \wedge (y_1 \vee x) \wedge (y_2 \vee x) \ \wedge \ (z_1 \vee \overline{x}) \wedge (z_2 \vee \overline{x}) \wedge (\overline{z_1} \vee \overline{z_2}) \,.$$

Since the first half of the formula forces x to be true in a satisfying assignment and the second forces it to be false, the formula is not satisfiable; it is the smallest unsatisfiable linear 2-CNF formula. Unsatisfiable linear k-CNF formulas can be constructed for all k, although their size needs to grow faster than 2^k, again a fact whose proof falls back on the Local Lemma.

Any linear k-CNF formula with at most $4^k/(4e^2k^3)$ clauses is satisfiable.

We will see that the bound in the condition is tight up to a polynomial factor. Via a probabilistic argument $8k^3 4^k$ clauses can be shown to suffice for unsatisfiability; the best explicit construction we know, however, delivers formulas of tower-like size (2 to the 2 to the $2 \ldots k$ times).

Algorithms, finally: Whenever the easily checkable conditions formulated above are satisfied, then the algorithmic problem of deciding satisfiability becomes trivial. However, whenever the Local Lemma is invoked, it is by no means obvious how to actually *construct* a satisfying assignment. This tantalising fact was resolved only recently via a randomised local repair algorithm as indicated above. We will present and analyse this method in the next section.

We return to deciding satisfiability. For k a positive integer, let us define $f(k)$ as the largest integer, so that every k-CNF formula with no variable of degree exceeding $f(k)$ is satisfiable; we know that $f(k) = \Theta(2^k/k)$. Clearly, satisfiability of k-CNF formulas with maximum variable degree at most $f(k)$ is trivially decidable in polynomial time. We might hope that slight violation of the bound may still allow for an efficient decision procedure. However, one can show that, provided $k \geq 3$, even for k-CNF formulas with max-degree at most $f(k) + 1$ the satisfiability problem becomes already NP-complete. This *sudden*

[1] The term "linear" is borrowed from hypergraph theory, where this must have been inspired by the behaviour of lines.

jump behaviour in complexity at $f(k)$ can be shown, although $f(k)$ is not known for k exceeding 4 (it is not even known whether the function f is computable). A similar immediate transition from trivial to NP-complete can be observed for the related problem for the conflict-neighbourhood size.

The remainder of this paper will treat the topic outlined above in more detail, mostly with proofs. We will also supply references and more of the historical background of the developments to today's state of knowledge.

Notation. We will assume (and have assumed) some familiarity with basic notions for boolean formulas in propositional logic and in discrete mathematics. Still, for the remaining more technical treatment, we want to clarify some notation and terminology. We like to regard clauses as sets of literals, formulas as sets of clauses. Let us actually go through a succinct recapitulation of our set-up: Given a set V of boolean variables, we set $\overline{V} := \{\overline{x} \mid x \in V\}$ and call the elements of $V \cup \overline{V}$ *literals over* V with V the *positive literals* and $\overline{V}$ the *negative literals*. A *clause* C *over* V is a set of literals over V with no pair x and $\overline{x}$ appearing simultaneously. A *CNF formula* F *over* V is a set of clauses; if all clauses in F have the same cardinality k, we call F a *k-CNF formula*. Although we regard formulas and clauses as sets, we sometimes return to the logic notation, writing $F \wedge C$ (instead of $F \cup \{C\}$) or even $F \wedge \neg C$ or similar.

An *assignment* α over variable set V is a mapping $\alpha : V \to \{0, 1\}$ that extends to $\overline{V}$ via $\alpha(\overline{x}) := 1 - \alpha(x)$ for $x \in V$ (1 for "true," 0 for "false"). α *satisfies* a clause if at least one of its literals evaluates to 1 under α. And α *satisfies* a CNF formula if it satisfies all of its clauses. A CNF formula is *satisfiable* if a satisfying assignment exists.

We denote the set of variables that occur in a clause C by $\mathrm{vbl}(C)$; for a CNF formula F, $\mathrm{vbl}(F) := \bigcup_{C \in F} \mathrm{vbl}(C)$. For a clause $C = \{u_1, u_2, \ldots, u_k\}$, we write $\overline{C} := \{\overline{u_1}, \overline{u_2}, \ldots, \overline{u_k}\}$ (note $\overline{C} \neq \neg C$, unless $k = 1$). The *neighbourhood of a clause* C in a CNF formula F is defined by $\Gamma(C) = \Gamma_F(C) := \{D \in F \mid \mathrm{vbl}(D) \cap \mathrm{vbl}(C) \neq \emptyset\}$. Analogously, the *conflict-neighbourhood of* C is $\Gamma'(C) = \Gamma'_F(C) := \{D \in F \mid C \cap \overline{D} \neq \emptyset\}$. The *degree of a variable* x *in a CNF formula* F is set to $\deg(x) = \deg_F(x) := |\{C \in F \mid x \in \mathrm{vbl}(C)\}|$.

We have already encountered $f(k)$, which we defined to be the largest integer d such that every k-CNF formula with maximum variable degree at most d is satisfiable. Similarly, let $l(k)$ be the largest integer d such that every k-CNF formula F for which $|\Gamma_F(C)| \leq d$, for all $C \in F$, is satisfiable. Let $lc(k)$ be defined analogously, but with $|\Gamma'_F(C)| \leq d$, for all $C \in F$, instead.

A *hypergraph* H is a pair (V, E) with V a finite set and $E \subseteq 2^V$; it is *k-uniform* if $|e| = k$ for all $e \in E$. H is called 2-*colourable* (or has *property B*) if there is a colouring of the vertices in V by red and blue so that no hyperedge in E is monochromatic. Extremal problems for 2-colourable hypergraphs have been considered since Erdős' papers [3, 4] in 1963. They relate to satisfiability of CNF formulas in that $H = (V, E)$ is 2-colourable iff the CNF formula $E \cup \{\overline{e} \mid e \in E\}$, with V now considered as set of boolean variables, is satisfiable. And, therefore, they will make their appearance during this presentation.

2 Local Lemma in Terms of SAT – Proof and Algorithm

Theorem 1. *Let $k \in \mathbb{N}$ and let F be a k-CNF formula. If $|\Gamma(C)| \leq 2^k/e - 1$ for all $C \in F$, then F is satisfiable.*

The statement was first formulated in the famous paper [2] by Erdős and Lovász, in terms of its application to the hypergraph 2-colouring problem. Its wide applicability to combinatorial questions soon became apparent. Nowadays it is usually formulated in general probabilistic terms in the following fashion.

Theorem 2 (Lovász Local Lemma, symmetric form)
Let $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ be any collection of events in a probability space, each one having probability at most p and such that each event is mutually independent of all but at most d of the other events. If $ep(d+1) \leq 1$, then with positive probability, none of the events in $\mathcal{A}$ occur.

The SAT formulation, Theorem 1, follows as an immediate corollary. Considering the random experiment of sampling truth assignments to the CNF formula F at random and defining A_i to be the event that clause number i becomes violated, each event has probability 2^{-k} and the desired bound follows. This way, it is a natural extension of the simple probabilistic argument bounding from below the total number of clauses in an unsatisfiable formula.

Theorem 1 is asymptotically tight. This is most simple to see as the CNF formula consisting of all 2^k clauses of size k over k variables is clearly unsatisfiable and has neighbourhoods of size $2^k - 1$ at each clause. In Section 3, we indicate how an unsatisfiable k-CNF formula having neighbourhoods of size 2^{k-1} each can be constructed, tightening even further the constant gap between the known lower and upper bounds. Note that in the general probability space setting as in Theorem 2 the constant e is known to be tight [5].

In the sequel, we give two proofs for Theorem 1. The first "existential" proof (from [2]) is beautifully short and astounding, but suffers from the mentioned shortcoming that it is non-constructive and so does not reveal how a satisfying assignment should be efficiently found. Whether this is in any way possible used to be a long-standing open problem until in 1991, Beck achieved a breakthrough by proving in [6] that a polynomial-time algorithm exists which finds a satisfying assignment to every k-CNF formula in which each clause has a neighbourhood of at most $2^{k/48}$ other clauses. His approach was deterministic and used the non-constructive version of the Local Lemma as a key ingredient, basically proving that even after truncating clauses to a 48th of their size (a step used to simplify the formula and make it fall apart into small components), a solution remains guaranteed that can then be looked for by exhaustive enumeration. Alon simplified Beck's algorithm and analysis by introducing randomness and presented an algorithm that works up to neighbourhoods of $2^{k/8}$ in size [7]. Czumaj and Scheideler later demonstrated that a variant of the method can be made to work for the non-uniform case where clause sizes vary [8]. In 2008, Srinivasan improved the bound of what was polynomial-time feasible to essentially $2^{k/4}$ by a more accurate analysis [9]. Later that year, Moser published a polynomial-time

algorithm that can cope with neighbourhood sizes up to $\mathcal{O}(2^{k/2})$ [10], and somewhat later an improved variant that allows for 2^{k-5} neighbours [11], which is asymptotically optimal with a constant gap.

The second proof we present here, finally, is a fully constructive version published by Moser and Tardos [12] which does not suffer from any gap to the existential version anymore. While it is general enough so as to apply to many applications of the Local Lemma, we will formulate it in terms of satisfiability here. The proof formalises the intuitive idea mentioned in the introduction: that the simplest possible method starting at a random point and then applying some corrections, thereby moving around violated clauses in the formula, always converges to a solution.

2.1 First Proof of Local Lemma – Existence

Let F be our k-CNF formula and let us require that each clause has a neighbourhood of at most $d := 2^k/e - 1$ other clauses. Suppose we select an assignment α of truth values to the variables uniformly at random. What is the probability that α satisfies F? If we can prove that probability to be positive, then F has to be satisfiable. Let us try to do so.

Let $F' \subset F$ be any subformula that arises from F by removing at least one clause. Let $C \in F \backslash F'$ be one of the clauses removed. α has a certain probability $Pr(F')$ of satisfying F'. We are interested to compute the drop in probability if we add back C as an additional constraint and we claim that this drop be bounded by a factor of $(1 - e2^{-k})$, that is $Pr(F' \wedge C) \geq (1 - e2^{-k})Pr(F')$ (or, equivalently, $Pr(F' \wedge \neg C) \leq e2^{-k}Pr(F')$). No matter what the factor exactly is, as long as it is positive, this readily gives what we have claimed, since the empty formula is satisfied with probability 1 and then successively adding back all of F's clauses diminishes that probability by a positive factor each step, leaving a positive probability in the very end.

So let us prove the auxiliary claim. We proceed inductively. Suppose the auxiliary claim has been proved for all subformulas F' up to a given size and now we would like to establish it for larger subformulas. First of all, there is a trivial special case. If the constraint C that we join back to F' is independent, that is, does not have any variables in common with F', then the events that F' or C are satisfied, respectively, are independent from one another and the probability decreases by a factor of exactly $(1 - 2^{-k})$. We have to understand now why lowering that factor to $(1 - e2^{-k})$ is sufficient to account for the (restricted) amount of possible dependencies that we might encounter. So, given the more problematic case that C shares some variables with F', let us get rid of those dependencies by removing, additionally, all clauses from F' that neighbour C; let $F'' := F' \backslash \Gamma(C)$. Now F'' and C are independent. Clearly, in this case

$$Pr(F'' \wedge \neg C) = 2^{-k}Pr(F'').$$

Note that we have removed from F' at most d clauses (due to the global hypothesis). By induction, we can add back all of these clauses one by one to F'' to get F' and thereby obtain

$$Pr(F') \geq (1 - e2^{-k})^d Pr(F'') \geq e^{-1} Pr(F'').$$

On the other hand, since every assignment satisfying F' satisfies F'', we have

$$Pr(F' \wedge \neg C) \leq Pr(F'' \wedge \neg C) = 2^{-k} Pr(F'').$$

The two results yield $Pr(F' \wedge \neg C)/Pr(F') \leq 2^{-k}/e^{-1}$, as claimed. □

2.2 Second Proof of Local Lemma – Algorithm

Recall our intuitive understanding of the problem setting. If we start with a randomly chosen assignment, then a 2^k-th of the clauses are, on average, violated. Now suppose that we continue in the most naive fashion: repeatedly select any of the violated clauses and just select new uniformly random values for each of the variables occurring in that clause until a satisfying assignment is reached. Such a strategy of successive local corrections might fail if correcting a violated clause causes lots of new clauses to be violated. But since the influence of a clause correction is restricted to the neighbourhood of that clause, then if such neighbourhoods are always sufficiently small, the strategy sounds intuitively promising. We will demonstrate that under the hypothesis of the Local Lemma, it converges to a satisfying assignment in an expected polynomial number of steps. The existence of such an assignment then follows with the correctness of the procedure.

Let us execute the algorithm and observe what it does, recording a *log* of what corrections are being applied, that is a mapping $L : \mathbb{N}_0 \to F$ with the meaning that in step t, the algorithm selects clause $L(t)$ for correction. We hope for the algorithm to terminate quickly, in particular after a finite number of steps, but in order to be rigorous, we have, for the moment, to allow for an infinite log and then prove that we will not ever encounter one. Moreover, let $N : F \to \mathbb{N}_0 \cup \{\infty\}$ be random variables that count the number of times a given clause occurs in the log, that is for $C \in F$, we define $N(C) := |\{t \in \mathbb{N}_0 | L(t) = C\}|$. Again, we a-priori have to allow for such a counter to take infinity as a value, but we will show that it never does. In fact, what we prove now is that for each clause $C \in F$, the expected value $E[N(C)]$ is upper bounded by a constant. Note that this implies everything we claim: Since in the expected case, each clause is corrected at most a constant number of times, the total number of clauses corrected is, in the expected case, bounded by $\mathcal{O}(|F|)$. So not only does the algorithm always terminate after a finite number of steps (implying the existence of a solution), it even returns after a polynomial number of operations.

Bounding the expected value $E[N(C)]$ is strikingly simple once we introduce a concept that goes back to Beck and Alon [6, 7]. The concept we are talking about is the one of witness trees. A *witness tree* is an unordered, rooted tree T along with a labelling $\sigma : V(T) \to F$ of its vertices $V(T)$ by clauses from F. Given a specific run of the algorithm and thus a log L, a witness tree can serve as a justification for the necessity of any of the executed correction steps. What do we mean? Let t be any time index such that $L(t)$ is defined. Now let us build

a witness tree in the following sense. Start with a root vertex r and label that vertex $\sigma(r) := L(t)$, that is by the clause corrected in step t. Now traverse the log backwards and for each time step $s = t-1, t-2, ..., 0$, check if the clause $L(s)$ has any variables that it shares with any of the labels in the tree built so far. If $L(s)$ is independent from all clauses currently serving as labels, discard it. If there *are* nodes in the tree that have variables in common with $L(s)$, then select any deepest of those nodes and create a new child node of it, labelling that new child $L(s)$. Once arriving at $s = 0$ we have built a witness tree $T(t)$ that justifies correction step t. In the following sense.

If we look at a witness tree $T(t)$, thereby forgetting everything else we have seen while the algorithm was running, we can reconstruct a significant portion of the execution history. Traversing the tree $T(t)$ in a bottom-up and level-by-level fashion (as in a reverse breadth-first-search that starts at the root), we obtain a sequence of clauses that is essentially a subsegment of the execution log. Each node we encounter during such a traversal represents some correction step in L with the label of the node being the clause corrected in that step. And the way we defined the witness $T(t)$ immediately assures us of two things: firstly, the ordering in which the corrections have taken place is similar to the ordering in which we traverse the nodes. It isn't identical, but it preserves what we will be interested in: Whenever two nodes v_1 and v_2 are labelled with clauses that depend on each other, i.e. that have common variables, then v_1 occurs before v_2 in the traversal if and only if v_1 represents a correction step occurring before v_2. Secondly, when we traverse some node v representing correction step t, then all correction steps $t' < t$ that relate to step t in the sense that $L(t)$ and $L(t')$ share common variables do occur in the tree and have therefore been traversed before.

What these two properties imply is the following: If we traverse our tree in the described way and we count the number of times some variable x has occurred so far in labelling clauses, then that number corresponds to the number of times x has been reassigned new values before the corresponding correction step. So if we have seen variable x already 10 times before we traverse a node v labelled $\sigma(v) = C$, then this means that at the time the correction v represents took place, x had its 10th new random value and was then assigned its 11th one. This in turn means that we can reconstruct, by just looking at the tree, all the 10 values x had been assigned before. This is because node v represents a time step where clause C was selected for correction, that is a time step when C was violated and thus the 10th value of x has to have been the one that dissatisfies the corresponding literal we find in C. The same holds for all other variables in the clause and for all other nodes we traverse.

Now suppose you are given a fixed witness tree T. What is the probability that exactly this tree can occur as witness for some correction step? As we have seen before, if we traverse T bottom-up we can reconstruct for each node the values the k variables in the corresponding clause were assigned before the correction step represented, that is we can reconstruct k of the random bits the algorithm has used. If the tree has n vertices, we can reconstruct nk bits in total, just looking at the tree. Since those bits are uniformly random, the probability that

all of them sample such that T can be constructed (we will say that T is *valid* if they evaluate as needed) is exactly 2^{-nk}. On the other hand, let us count how many witness trees there exist in total. Let us fix some clause $C \in F$ and some number n and let us count the number of witness trees of order n which have C as the label of their root vertex. What restricts that number is the way in which witness trees were defined, which requires that if u is a child node of v, then the label $\sigma(u)$ must be a neighbour of the clause $\sigma(v)$. This allows us to embed each witness tree rooted at label C into an infinite tree that just enumerates neighbouring nodes: Consider an infinite tree with its root labelled C and such that each node v labelled $\sigma(v)$ has $|\Gamma(\sigma(v))|$ children labelled $\Gamma(\sigma(v))$. Such a tree is at most d-ary and each witness tree is clearly a subtree of it. A two-line counting exercise shows that an infinite rooted ($\leq d$)-ary tree has at most $(ed)^n$ subtrees of size n. Therefore there are at most $(ed)^n$ witness trees of order n that have C as their root label. Since each of them may occur with a probability of at most 2^{-nk}, the expected number of witness trees of size n that can occur is bounded by $(ed2^{-k})^n$. Plugging in d and summing over all possible sizes $n \geq 1$, this becomes a geometric series that converges to a constant. Hence, there is at most a constant expected number of valid witness trees rooted at C.

What does this mean? Clause C occurs $N(C)$ times $t_1, t_2, \ldots, t_{N(C)}$ in the execution log. For each of those times we can ask for a witness tree $T(t_1)$, $T(t_2), \ldots,$ $T(t_{N(C)})$ to justify that correction step. All of those trees have to be valid, and they are distinct since $T(t_{i+1})$ needs to have basically the same vertices as $T(t_i)$ (though maybe arranged differently) and at least one more (to represent step t_{i+1}). So $N(C)$ is at most as large as the number of valid witness trees rooted at C. Since the latter number is bounded by a constant in expectation, the former is so, too. And this concludes the argument. □

2.3 A Stronger Variant – Conflicts

There is a slightly stronger version of the Lovász Local Lemma, referred to as the lopsided Local Lemma ([13, 1, 14]), which does not only distinguish between dependent and independent events but also discriminates between positive and negative correlations. In terms of satisfiability, this means that the bound on the maximum neighbourhood size is replaced by a bound on conflict neighbourhoods.

Theorem 3. *Let $k \in \mathbb{N}$ and let F be a k-CNF formula. If $|\Gamma'(C)| \leq 2^k/e - 1$ for all $C \in F$, then F is satisfiable.*

Both the purely existential and the constructive proof we detailed above can be adapted so as to demonstrate this statement. For the latter, the same algorithm will work and for the analysis it suffices to observe that witness trees built by attaching only lopsided neighbours during backward traversal of the log equally allow to reconstruct k bits of the randomness used per vertex, irrespective of the fact that a smaller amount of information might be encoded by the tree.

The lopsided Local Lemma has successfully been used to establish better bounds for the number of dependencies or the number of occurrences per variable

that we can allow, still being guaranteed that all formulas within that class are satisfiable. Berman, Karpinski and Scott, e.g., have demonstrated in [15], using the lopsided Local Lemma, that every 6-, 7-, 8- or 9-CNF formula in which every variable occurs at most 7, 13, 23 or 41 times, respectively, is satisfiable. Their argument can be used to obtain bounds for other values of k as well and it can be made constructive by the methods we presented in the previous section. Let us now have a closer look at bounded occurrence instances of satisfiability.

3 Bounded Variable Degree

Let us call a k-CNF formula in which no variable occurs in more than d clauses a (k, d)-*CNF formula*. Recall $f(k)$ which we can now equivalently define as the unique integer so that all $(k, f(k))$-CNF formulas are satisfiable and an unsatisfiable $(k, f(k) + 1)$-CNF formula exists. For $k \geq 1$, a $(k, 0)$-CNF formula has to be empty and thus satisfiable. The CNF formula of all 2^k k-clauses over a given set of k variables constitutes an unsatisfiable $(k, 2^k)$-CNF formula, so $f(k)$ exists with $0 \leq f(k) \leq 2^k$.

The first to consider $f(k)$ was Tovey [16] in 1984 (with Christos Papadimitriou raising the question). He showed $f(k) \geq k$ (by an argument based on Hall's Theorem which we will provide later in Lemma 3); he suspected his bound to be weak and actually conjectured that all $(k, 2^{k-1} - 1)$-CNF formulas are satisfiable [16, Conjecture 2.5].

A clearer picture of $f(k)$ has evolved since then. For, if every variable occurs at most d times in a k-CNF formula, no clause can collect more than $k(d - 1)$ neighbours. Thus, with the Local Lemma (as in Theorem 1), the inequality $k(d - 1) \leq 2^k/e - 1$ implies that every (k, d)-CNF formula is satisfiable. This connection and the implied bound of $f(k) \geq \lfloor 2^k/(ek) \rfloor$ was first established by Kratochvíl, Savický, and Tuza [17] – still the best lower bound known for k large.

They supplied also an upper bound of $2^{k-1} - 2^{k-4} - 1$. Significant progress on the upper end was made when Savický and Sgall [18] showed $f(k) = O(k^{-0.26}2^k)$. This was further improved to $f(k) = O((2^k \log k)/k)$ by Hoory and Szeider [19], now only a log-factor shy of the lower bound. Recently, the gap has been closed up to a constant factor by Gebauer [20] and $f(k) = \Theta(2^k/k)$ is settled. (A brief discussion of the situation for small k is postponed to the end of this section.)

Theorem 4. *For k a large enough integer,*

$$\left\lfloor \frac{2^k}{ek} \right\rfloor \leq f(k) < \frac{2^{k+1}}{k}\,.$$

If k is a sufficiently large power of 2 we have $f(k) < 2^k/k$.

SAT connects to many (sometimes seemingly unrelated) problems. The proof of the upper bound in Theorem 4 is another example for this fact: The actual construction was originally developed for refuting a conjecture of Beck on Combinatorial games [21]. In such a game Maker and Breaker take turns in choosing

vertices from a given hypergraph. Maker wants to completely occupy a hyperedge and Breaker tries to prevent this. The problem is to find the minimum $d = d(k)$ such that there is a k-uniform hypergraph of maximum vertex degree d where Maker has a winning strategy.

One possible strategy Maker can use is to partition all but at most one of the vertices into pairs and whenever Breaker claims one vertex of a pair, Maker takes the other one. If Maker uses such a *pairing strategy*, this game on hypergraphs is in some sense equivalent to unsatisfiability. Indeed, given a hypergraph H together with a pairing P we can interpret this as a CNF formula F where the hyperedges of H are understood as clauses and the two vertices of a pair of P are considered as complementary literals. It is easily seen that Maker wins the game on H using the pairing strategy according to P if and only if F is unsatisfiable.

If there is a k-uniform hypergraph of maximum vertex degree d with a winning pairing strategy for Maker, then there is an unsatisfiable $(k, 2d)$-CNF formula.

This clearly shows the relation between the two problems. For the proof a third player enters the picture: binary trees. In this presentation we will proceed directly from binary trees to CNF formulas.

3.1 Trees with All Leaves Deep, But Few Leaves Close Below Any Node

We consider binary trees where every node has either two or no children. In such a binary tree we say that a leaf v is *ℓ-close* to a node w if w is an ancestor of v, at distance at most ℓ from v. For k and d positive integers, we call a binary tree T a *(k, d)-tree* if (i) every leaf has depth[2] at least $k - 1$ and (ii) for every node u of T there are at most d leaves $(k-1)$-close to u; (from (i) it follows that every leaf is $(k-1)$-close to exactly k nodes). A moment of reflection shows that every binary tree with all leaves at depth at least $k - 1$ is a $(k, 2^{k-1})$-tree. The following lemma motivates a search for (k, d)-trees with d smaller than 2^{k-1}.

Lemma 1. *Let T be a (k, d)-tree, k and d positive integers. Then there is a k-CNF formula $F = F(T)$ with the following properties.*

(a) *For m, the number of leaves of T, we have $|F| = 2m$.* (b) *Every* literal *occurs in at most d clauses of F.* (c) *If* $\mathrm{vbl}(C) \cap \mathrm{vbl}(D) \neq \emptyset$ *for two distinct clauses C and D in F, then these clauses are conflict-neighbours with a* unique *variable that appears in C and D with opposite signs.* (d) *F is unsatisfiable.* (e) *If, for every node u in T, there is at least one leaf that is $(k-1)$-close to u, then $|F| = |\mathrm{vbl}(F)| + 1$ and F is minimal unsatisfiable.*

(1) *F is an unsatisfiable $(k, 2d)$-CNF formula; thus, $f(k) \leq 2d - 1$.*

(2) *F is an unsatisfiable k-CNF formula with $|\Gamma(C)| \leq kd$ for all clauses $C \in F$; hence, $l(k) \leq kd - 1$.*

[2] The root has depth 0, its children have depth 1, ...

Proof. We move to a binary tree $\widehat{T}$ by attaching the roots of two copies of T as the two children of a new root r. This yields a (k, d)-tree (actually, with all nodes of depth at least k). Obviously, it has $2m$ leaves and, as it goes with binary trees, $4m - 1$ nodes altogether. Two nodes are called *siblings*, if they share the same parent. Leaving aside the root, the remaining $4m - 2$ nodes of $\widehat{T}$ can be partitioned into $2m - 1$ sibling pairs.

For V some set of $2m - 1$ boolean variables, we label the nodes of $\widehat{T}$ other than the root by literals in $V \cup \overline{V}$ so that every literal appears exactly once and siblings get complementary literals. With every leaf v we associate a clause C_v by walking along a path of length $k - 1$ from v towards the root and collecting all labels encountered on this path (i.e. the labels of all nodes to which v is $(k - 1)$-close). The set of clauses C_v, over all leaves v of $\widehat{T}$, constitutes F.

The fact that every leaf of $\widehat{T}$ has depth at least k guarantees that paths of length $k - 1$ starting at leaves will never reach the root. So there are indeed always k literals to collect and F is a k-CNF formula. Also $|F| = 2m$ is obvious (therefore (a)). The defining property of (k, d)-trees guarantees that no label is collected more than d times (hence (b)). F is unsatisfiable (as claimed in (d)), for if an assignment α over V is given, it defines a path from the root to a leaf, say v, by always proceeding to the unique child whose label is mapped to 0 by α; C_v's fate of being violated by α is determined.

Let us settle the claimed neighbour property in (c). Suppose that $\mathrm{vbl}(C_u) \cap \mathrm{vbl}(C_v) \neq \emptyset$ for leaves u and v, $u \neq v$. If $x \in C_u$ and $\overline{x} \in C_v$, then the parent w of the siblings labelled by x and $\overline{x}$, respectively, is the lowest common ancestor of u and v (i.e. the node of maximum depth that appears on both paths from u and v, respectively, to the root); therefore x is unique. For the existence of a complementary pair in C_u and C_v, consider the lowest common ancestor w of u and v; w cannot be a leaf since $u \neq v$. The literals occurring in the two subtrees rooted at the children of w do not share any common variable other than the complementary pair placed at the children of w. Hence, the literals associated with these two children must appear in the clauses, one literal in C_u the other complementary literal in C_v, since otherwise their variable sets are disjoint. Assertion (c) is shown.

Next we prove (e). With the assumption given, all $2(2m - 1)$ literals appear in some clause, so $|\mathrm{vbl}(F)| + 1 = 2m = |F|$. It remains to show satisfiability of $F_v := F \setminus \{C_v\}$ for all leaves v. For v a leaf of $\widehat{T}$, consider the following procedure, in the course of which, besides defining an assignment, we make nodes responsible for clauses in F_v.

First, set the literals of all nodes on the path from v to root r (excluding r) to 0 and initialise S as the set of all siblings of these nodes. Now, while S is nonempty, (i) remove some $u \in S$ from S, (ii) set its literal to 1, (iii) choose a leaf v_u (maybe u itself) that is $(k - 1)$-close to u, (iv) set all literals of nodes on the path from v_u to u (excluding u) to 0, (v) add all siblings of these nodes to S, and, finally, (vi) make u responsible for C_{v_u}.

In this proceeding we see an invariant maintained: The subtrees rooted at the nodes in S are disjoint and they comprise exactly the nodes with their literals

undefined. Thus the value of a literal is set at most once and, actually, at least once, since the node of minimal depth with its literal undefined would have to be in S (and therefore the procedure couldn't possibly have stopped already). Now it is easily seen that this procedure gives a valuation of the literals that is indeed an assignment of V. Whenever a node u is responsible for a clause, then this clause is satisfied due to u's literal. Also, a clause cannot have two nodes responsible for it, while every node with its literal set to 1 is responsible for some clause. It becomes obvious that the responsibility map is a bijection between the $2m-1$ nodes with their literal set to 1 and the $2m-1$ clauses in F_v. Hence, F_v is satisfied. And so are we.

Implication (1) follows from (b) and (d). (2) follows from (b-d): In particular, if we define $\mathrm{occ}(u) := |\{C \in F \mid u \in C\}$, then (c) allows us to write $|\Gamma(C)|$ as $\sum_{u \in C} \mathrm{occ}(\overline{u})$ – hence, at most kd – for every clause $C \in F$. □

In a similar fashion, one can show that a (k,d)-tree yields a k-uniform hypergraph of maximum vertex degree d where Maker has a winning pairing strategy. We are left with the task of constructing (k,d)-trees with sufficiently small d.

Lemma 2. (i) *A $(k, \lfloor 2^k/k \rfloor)$-tree exists for every sufficiently large k.* (ii) *If k is a sufficiently large power of 2 then a $(k, 2^{k-1}/k)$-tree exists.*

Lemmas 1(1) and 2 imply the upper bounds in Theorem 4. Also Lemmas 1(2) and 2(ii) give us an upper bound of $l(k) < 2^{k-1}$ for large enough powers of 2. So let us now summarise also our findings for $l(k)$ and $lc(k)$.

Theorem 5. *We have*

$$\lfloor 2^k/e \rfloor - 1 \leq lc(k) \leq l(k) < 2^{k-1}$$

where the upper bound holds for k any sufficiently large power of 2 but can be replaced by $2^k - 1$ for all positive integers k.

Moreover, the relation to $l(k)$ tells us that we will not be able to find (k,d)-trees with $d < (l(k)+1)/k$ and, therefore, – along the lower bounds of $l(k)$ in Section 2 – the range $d < 2^k/(ek) - 1$ is inaccessible.

The proof of Lemma 2 is tedious and too long for this presentation. However, a weaker statement, still settling the asymptotics of $f(k)$, can be shown with less effort.

Proof of existence of $(k, 2^{k+1}/k)$-trees for k a power of 2. We have $2^{k+1}/k \geq 2^{k-1}$ for $k \leq 4$, so we can assume that $k \geq 8$ in our proof. Let T' be a full binary tree of height $k-1$ (i.e. a binary tree where all leaves have the same depth $k-1$). We subdivide its leaves into intervals of length $k/2$. For $\{v_0, \ldots, v_{k/2-1}\}$ such an interval, we attach a full binary subtree of height i to v_i. Let T denote the resulting tree and let the *leaf-range* $r(v)$ of a node v denote the number of leaves $(k-1)$-close to v.

It suffices to show that $r(v) \leq \frac{2^{k+1}}{k}$ *for all nodes of T*. We apply induction on the depth i of v. For $i = 0$ the claim holds. Indeed, note that out of each of the

$\frac{2^{k-1}}{k/2}$ intervals, exactly one vertex (v_0, respectively) is $(k-1)$-close to the root and, hence, the leaf-range of the root is $\frac{1}{2}\frac{2^{k+1}}{k}$. Now suppose that v has depth $i \in \{1, \ldots, k/2-1\}$. Note that the set of descendants of v at depth $k-1$ can be subdivided into $\frac{2^{k-1-i}}{k/2}$ intervals, i.e. at least one interval for the values of k we consider. Let v' denote the parent of v. By construction the number of leaf descendants which have distance at most $k-2$ from v equals $r(v')/2$. Moreover, every interval $\{v_0, \ldots, v_{k/2-1}\}$ gives rise to 2^i leaves on level $k-1+i$, implying that the number of leaf descendants of v which have distance exactly $k-1$ from v equals $\frac{2^{k-1-i}}{k/2} \cdot 2^i = \frac{1}{2}\frac{2^{k+1}}{k}$. So altogether $r(v) \leq \frac{r(v')}{2} + \frac{1}{2}\frac{2^{k+1}}{k} \leq \frac{2^{k+1}}{k}$. It remains to consider the case where v has depth at least $k/2$. By construction no leaf of T has depth larger than $k/2+k-2$, implying that the leaf-range of v is at most the leaf-range of its parent. □

3.2 Small Values

Although the lower bounds on $f(k)$, $l(k)$ and $lc(k)$ we can derive via the Local Lemma grow exponentially, they are weak for small values of k. Here another classic from combinatorics enters the picture: Hall's marriage theorem.

Lemma 3. (1) $f(k) \geq k$ *for* $k \geq 1$ [16] *and* (2) $l(k) \geq lc(k) \geq k$ *for* $k \geq 2$.

Proof. (1) For $k \geq 1$, let F be a k-CNF formula over a variable set V with no variable occurring in more than k clauses of F. Consider the incidence graph between clauses and variables, i.e. the bipartite graph with vertex set $F \cup V$, where $\{C, x\}$ is an edge iff $x \in \mathrm{vbl}(C)$. In this graph, clause-vertices have degree exactly k and by assumption variable-vertices have degree at most k. Therefore, Hall's condition for a matching covering all clause-vertices holds. An assignment is now defined by letting every variable x that is matched to a clause C map to the value so that it satisfies C. The matching property prevents conflicts in doing so. No matter how we complete the assignment for unmatched variables, it will satisfy all clauses.

(2) Let $k \geq 2$. We will actually prove a bound of $lc(k) \geq \lceil (f(k)+1)/2 \rceil + k - 2$; with the bound on $f(k)$ from (1) this yields a lower bound of $\lceil (3k-3)/2 \rceil \geq k$.

So we have to show that every unsatisfiable k-CNF formula F contains a clause C with $|\Gamma'_F(C)| \geq \lceil (f(k)+1)/2 \rceil + k - 1$. First we pass to a minimal unsatisfiable k-CNF formula $G \subseteq F$. G has a variable x with $\deg_G(x) \geq f(k)+1$; w.l.o.g. we assume that literal $\overline{x}$ occurs at least $\lceil (f(k)+1)/2 \rceil$ times. Now choose a clause $C \in G$ with literal x. $\Gamma'_G(C)$ contains all clauses with $\overline{x}$. In addition, according to Lemma 4, for all variables $z \in C \setminus \{x\}$ there has to be a clause $D_z \in G$ with the property that z is the unique variable that appears in C and D_z with opposite signs. It follows that $|\Gamma'_G(C)| \geq \lceil (f(k)+1)/2 \rceil + k - 1$. This concludes the argument, since $\Gamma'_F(C) \supseteq \Gamma'_G(C)$. □

In fact, $f(k) = k$ is known for $k \leq 4$ ([16] for 3 and [22] for 4). The best known bounds for $k = 5$ are $5 \leq f(5) \leq 7$, [23]. $k = 6$ is the first value for which the bound in Lemma 3(1) is known not to be tight, [15]: $7 \leq f(6) \leq 11$. See [23] for

a further discussion of $f(k)$ for small k with currently best bounds known (and also Section 3.3 below). The bound for $lc(k)$ suffices our needs in the proof of Theorem 9 below, but perhaps even the lower bound of 3 for $lc(3)$ is not tight; we have not followed up this matter further.

We conclude with the missing lemma employed in the proof of Lemma 3(2) (in the spirit of the properties of so-called blocked clauses introduced by Kullmann [24, Definition 3.1]).

Lemma 4. *Let F be a minimal unsatisfiable CNF formula. Consider x and $C \in F$ with $x \in \mathrm{vbl}(C)$. Then there is a clause D with the property that x is the* unique *variable that appears in C and D with opposite signs.*

Proof. Since F is minimal, $F \setminus \{C\}$ has a satisfying assignment α. By assumption of unsatisfiability of F, α cannot satisfy C. Now switch the value of x in the assignment, thereby satisfying C and thus violating some other clause $D \in F$. Now, as easily seen, D serves the purpose. □

3.3 A Special Class of Unsatisfiable CNF Formulas – MU(1)

MU(1) is defined as the set of all minimal unsatisfiable CNF formulas F with $|F| = |\mathrm{vbl}(F)| + 1$. This definition may appear somewhat arbitrary, so why care? First, when searching for extremal unsatisfiable CNF formulas with the properties we are interested in, we can confine ourselves to minimal unsatisfiable CNF formulas. Second, as observed by A. Tarsi (cf. [25]), $|F| - |\mathrm{vbl}(F)|$, called *deficiency*, is positive for every minimal unsatisfiable CNF formula F; hence, CNF formulas in MU(1) are minimal w.r.t. to deficiency. Formulas in MU(1) can be recognised efficiently [26, 27, 28] and they have been shown to be universal in the sense that every unsatisfiable CNF formula F has a $G \in \mathrm{MU}(1)$ with a homomorphism from G to F [29]. We do not define homomorphisms for CNF formulas here. They preserve unsatisfiability but alter other properties, e.g. due to "clause collapses" they can decrease neighbourhood sizes and variable degrees. Still, many extremal unsatisfiable CNF formulas can be drawn from MU(1).

With this in mind and, returning to our problem, given that we do not even know whether $f(k)$ is computable, Hoory and Szeider [23] headed for the more modest goal of investigating $f_1(k)$, the largest integer such that every $(k, f_1(k))$-CNF formula in MU(1) is satisfiable. They show that f_1 is computable, in fact, reasonably efficiently. With $f(k) \leq f_1(k)$, this allows them to derive the best known upper bounds for $f(k)$ for small k: $f(5) \leq 7, f(6) \leq 11, f(7) \leq 17, f(8) \leq 29, f(9) \leq 51$.

While the previous bound of $f(k) = O((2^k \log k)/k)$ in [19] was not established through formulas in MU(1), the constructions in [20] reside in MU(1); indeed, Lemma 1(e) provides this fact, since it is not too hard to see that if a (k, d)-tree exists, then we can modify it so that the assumption of Lemma 1(e) holds. Therefore, we can conclude that $f(k)$ and $f_1(k)$ are within a constant factor of each other. Whether $f(k) = f_1(k)$ for all k is an interesting open question.

4 Linear Formulas

We call a CNF formula F *linear* if $|\mathrm{vbl}(C) \cap \mathrm{vbl}(D)| \leq 1$ for any two distinct clauses C and D in F. For example, the formula $\{\{x, y\}, \{\bar{y}, z\}, \{\bar{x}, \bar{z}\}\}$ is linear, whereas the formula $\{\{x, y, z\}, \{\bar{x}, y, u\}\}$ is not. This class of formulas has been introduced in [30]. It is a natural analogue of the notion of *linear hypergraphs*: A hypergraph $H = (V, E)$ is linear if $|e \cap f| \leq 1$ for any two distinct edges $e, f \in E$. In this section we investigate the following questions: Are there unsatisfiable linear k-CNF formulas, for each k? If yes, how large do they have to be? There is the analogous question asking for k-uniform linear hypergraphs that are not 2-colourable (see last paragraph of introduction). Existence of those has been shown by Abbott [31], and Erdős and Lovász [2] give lower bounds on their size (in terms of number of vertices and hyperedges). Bounds for chromatic numbers exceeding 2 can be found in [32]. Given a k-uniform non-2-colourable hypergraph H with m hyperedges, we immediately obtain an unsatisfiable k-CNF formula $F(H)$ with $2m$ clauses (as described at the end of the introduction). However, for $k \geq 2$, even if H is linear, $F(H)$ is certainly not. Therefore, it is not clear (but true, as we will see) that bounds on the size of unsatisfiable linear k-CNF formulas are similar to those of non-2-colourable linear k-uniform hypergraphs.

Let $f_{\mathrm{lin}}(k)$ be the largest integer so that every linear $(k, f_{\mathrm{lin}}(k))$-CNF formula is satisfiable. Note $f_{\mathrm{lin}}(k) \geq f(k) \geq \lfloor 2^k/(ek) \rfloor$.

Theorem 6 ([33]). *Any unsatisfiable linear k-CNF formula has at least*

$$\tfrac{1}{k}\left(1 + f_{\mathrm{lin}}(k-1)\right)^2 > \tfrac{4^k}{4e^2k^3}$$

clauses. There exists an unsatisfiable linear k-CNF formula with at most $8k^3 4^k$ clauses.

Remark. $\frac{1}{k}\left(1 + f_{\mathrm{lin}}(k-1)\right)^2 \leq 8k^3 4^k$ follows; thus $f_{\mathrm{lin}}(k-1) \in \mathcal{O}(k^2\, 2^k)$.

Proof. The proof of the lower bound is similar to the one for the size of non-2-colourable linear k-uniform hypergraphs in [2]. The following lemma is instrumental.

Lemma 5. *Let F be a linear k-CNF formula. If there are at most $f_{\mathrm{lin}}(k-1)$ variables of degree exceeding $f_{\mathrm{lin}}(k-1)$, then F is satisfiable.*

First we show how the lemma implies the lower bound. Let X be the set of variables x with $\deg_F(x) > f_{\mathrm{lin}}(k-1)$. If F is unsatisfiable, then by the lemma, $|X| > f_{\mathrm{lin}}(k-1)$. Therefore, the lower bound follows from

$$|F| = \tfrac{1}{k}\textstyle\sum_{x \in \mathrm{vbl}(F)} \deg_F(x) \geq \tfrac{1}{k}(1 + f_{\mathrm{lin}}(k-1))|X| \geq \tfrac{1}{k}\left(1 + f_{\mathrm{lin}}(k-1)\right)^2$$

Proof (of the lemma). For a literal u, let $\deg_F(u)$ denote the degree of the variable underlying u in F. First we construct a linear $(k-1)$-CNF formula F' as follows: For every clause $C \in F$, let u_C be a literal of C that maximises $\deg_F(u_C)$ and write $C' := C \setminus \{u_C\}$; F' is obtained as $F' := \{C' \,|\, C \in F\}$. We

claim that $\deg_{F'}(x) \leq f_{\text{lin}}(k-1)$ for all variables x; thus, F' is satisfiable and therefore F is satisfiable.

Consider a variable x. Clearly, $\deg_{F'}(x) \leq \deg_F(x)$ and so if $\deg_F(x) \leq f_{\text{lin}}(k-1)$, we are done. Otherwise, let $C'_1, \ldots, C'_t$, $t = \deg_{F'}(x)$, be the clauses in F' containing x or $\bar{x}$. There are clauses $C_1, \ldots, C_t$ in F such that $C'_i = C_i \setminus \{u_{C_i}\}$, $1 \leq i \leq t$. By choice of u_{C_i}, $\deg_F(u_{C_i}) \geq \deg_F(x) > f_{\text{lin}}(k-1)$. Since F is linear, the u_{C_i}'s have to be distinct, thus by assumption, $t \leq f_{\text{lin}}(k-1)$. □

We now prove the upper bound. Take a linear k-uniform hypergraph $H = (V, E)$ with n vertices and m edges, to be determined later. By interpreting the vertices of H as variables and the hyperedges as clauses, this is a (satisfiable) linear k-CNF formula. We now replace each literal in each clause by its complement with probability $\frac{1}{2}$, independently in each clause. Let F denote the resulting (random) formula. Any fixed assignment α has a $1 - 2^{-k}$ chance of satisfying a given clause of F, and thus

$$\Pr[[]\alpha \text{ satisfies } F] = (1 - 2^{-k})^m < e^{-m2^{-k}} .$$

There are 2^n distinct assignments, hence by the union bound

$$\Pr[[]\text{some } \alpha \text{ satisfies } F] < 2^n e^{-m2^{-k}} = e^{\ln(2)n - m2^{-k}} .$$

If $m/n \geq \ln(2)2^k$, the above expression is at most 1, and hence with positive probability, no assignment satisfies F. In other words, at least one F obtained in the above fashion is not satisfiable.

We construct a linear k-uniform hypergraph with few hyperedges, but with a large hyperedge-vertex ratio. Let $q \in \{k, \ldots, 2k\}$ be a prime power. Choose $d \in \mathbb{N}$ such that $q^2 \ln(2)2^k \leq q^d < q^3 \ln(2)2^k$ and set $n := q^d$. Consider the d-dimensional vector space $\mathbb{F}_q^d$ over the field $\mathbb{F}_q$. It has n elements, called *points*. In a vector space, there is a line through any pair of points, and a line has q elements. Hence there are exactly $\binom{n}{2}/\binom{q}{2} \geq \frac{n^2}{q^2}$ lines in $\mathbb{F}_q^d$. By choice of d, we have $n \ln(2)2^k \leq \frac{n^2}{q^2}$, hence we can choose $m := n \ln(2)2^k$ distinct lines in $\mathbb{F}_q^d$. From each such line arbitrarily select k points and form a hyperedge. Let E be the set of all m hyperedges formed this way. (The reader may easily check that two distinct lines cannot yield the same hyperedge.) Thus, $H = (\mathbb{F}_q^d, E)$ is a k-uniform hypergraph. It is a linear hypergraph, since any pair of distinct lines intersect in at most one point. By construction, $\frac{m}{n} = \ln(2)2^k$, and $m = n \ln(2)2^k \leq q^3 \ln(2)^2 4^k \leq ln(2)^2 8k^3 4^k$, which proves the upper bound. □

This proof is simpler than the probabilistic construction of a non-2-colourable k-uniform linear hypergraph in [32]. This has a good reason: In our case, we inject randomness by choosing the signs of literals, whereas in the hypergraph case, there are no signs, and randomness comes only in the form of selecting hyperedges from some large set. This cannot be done independently for every hyperedge, since one has to guarantee linearity.

A question that might have formed in the reader's mind is what happens when one relaxes linearity to require that two clauses share at most one *literal*,

i.e. $|C \cap D| \leq 1$, as opposed to the stricter requirement that $|\mathrm{vbl}(C) \cap \mathrm{vbl}(D)| \leq 1$. The answer is that little changes. We can prove almost the same lower bound as in Theorem 6, sacrificing only a constant factor. However, if we require that any two distinct clauses C, D have at most one conflict, i.e. $|C \cap \overline{D}| \leq 1$, things change dramatically, see Section 4.2 below.

What happens if we require that $|\mathrm{vbl}(C) \cap \mathrm{vbl}(D)| \leq \ell$, for some $\ell \geq 2$? Here our bounds also change significantly, and the exponential function in upper and lower bound will no longer be 4^k, but $2^{\frac{\ell+1}{\ell}k}$. If ℓ is a constant (i.e. does not grow with k), upper and lower bounds are still only a polynomial factor apart. The proof goes along similar lines as for Theorem 6 and the details can be found in [33]. The bounds comply nicely with those in [32] for 2-colourability of hypergraphs in which two hyperedges can intersect in at most ℓ vertices.

4.1 Why Are Small Explicit Constructions So Hard to Come Up with?

It is often surprisingly easy to obtain rather tight bounds through a probabilistic construction – and frustratingly difficult to come up with an *explicit* one. In the case of linear formulas, explicit constructions have been given in [34, 35]. However, the size of the constructed formulas is terrifying: For an unsatisfiable linear k-CNF formula, it takes $2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}$ clauses, where the size of the tower is k. Actually, we can provide some evidence for why small explicit constructions may be difficult. Loosely speaking, there are three ways to come up with unsatisfiable CNF formula: First, one can build formulas where unsatisfiability follows immediately from construction, by *local* considerations. This is the case for the "complete formula" mentioned in Section 2, and also for the formula constructed in the proof of Theorem 4. Second, unsatisfiability can follow from a probabilistic or counting argument, as in the proof of Theorem 6. Third, unsatisfiability can follow from some more *global* combinatorial principle (e.g. the pigeon hole principle or the fact that in a graph, the number of vertices having odd degree must be even). This is typically the case for formulas with provably large resolution complexity (see Ben-Sasson and Wigderson [36] for several beautiful proofs on resolution complexity). Normally, formulas obtained the first way have small resolution complexity, even short *treelike* resolution proofs. Therefore, it seems unlikely to obtain unsatisfiable linear k-CNF formulas of reasonable size going the first way, for one can prove the following:

Theorem 7 ([33]). *Any resolution tree of any unsatisfiable linear k-CNF formula has at least $2^{2^{(k-1)/2}-1}$ nodes.*

Let us recall the class MU(1) from Section 3.3. The extremal examples for most parameters considered so far are in MU(1): for f, l and lc via the tree derived formulas from Section 3; also an unsatisfiable k-CNF formula with 2^k clauses can be found in MU(1) (see argument for Proposition 1 below). For linear CNF formulas, the explicit constructions in [34, 35] can be adapted to result in formulas

in MU(1), but they exhibit the tremendous size as mentioned. This is inherently so for linear CNF formulas in MU(1).

Theorem 8 ([33]). *For every $\epsilon > 0$ there exists some $c \in \mathbf{N}$ such that every linear k-CNF formula in* MU(1) *has at least $a^{a^{\cdot^{\cdot^{\cdot^{a}}}}}$ clauses, where $a = 2 - \epsilon$ and the size of the tower is $k - c$.*

4.2 1-Conflicts

Clauses C and D are called 1-*conflict neighbours*, if exactly one variable occurs in C and D with opposite signs,[3] i.e. $|C \cap \overline{D}| = 1$. We know already from the introduction (see also Lemma 4) that 1-conflicts have to occur in unsatisfiable CNF formulas (unless there is an empty clause); hence, they are crucial and deserve special attention. In the presence of linear formulas, it seems natural to consider CNF formulas where all conflicts are restricted to 1-conflicts, so let us call a CNF formula *conflict-linear* if each pair of clauses either has no conflict or has a 1-conflict. Here the typical questions we ask can be easily resolved.

Proposition 1. *For every nonnegative integer k, there is an unsatisfiable conflict-linear k-CNF formula with 2^k clauses.*

This is tight, since *all* k-CNF formulas with less than 2^k clauses are satisfiable.

Proof. Follow the construction of an unsatisfiable k-CNF formula $F(T)$ as in Lemma 1 starting from a full binary tree T of height $k-1$ (with 2^{k-1} leaves). This readily delivers a CNF formula as required. □

We define $lc_1(k)$ as the largest integer so that all k-CNF formulas with all 1-conflict neighbourhoods of size at most $lc_1(k)$ are satisfiable. Note that if we take all 2^k k-clauses over a given set of k variables, then in the resulting unsatisfiable k-CNF formula all 1-conflict neighbourhoods have size k. Therefore, $lc_1(k) \leq k - 1$. This bound is already tight: Given an unsatisfiable k-CNF formula F, consider a minimal unsatisfiable subset G of F. By Lemma 4, for all clauses in G the number of 1-conflict neighbours in G has to be at least k and 1-conflict neighbourhoods in F can only be larger. We have not only determined $lc_1(k) \geq k-1$ in this way; in fact, we have shown that every unsatisfiable k-CNF formula has at least 2^k clauses with 1-conflict neighbourhoods of size at least k.

Proposition 2. $lc_1(k) = k - 1$.

5 A Sudden Jump in Complexity

Satisfiability of $(k, f(k))$-CNF formulas is trivially decidable in polynomial time. If the degree bound is relaxed, we agree that some inspection of instances is required, but we would hope that the problem does not immediately develop the

[3] Equivalently, C and D are 1-conflict neighbours iff their resolvent exists.

full computational complexity of SAT. Tovey [16], however, proved that for 3-CNF formulas with maximum variable degree $f(3)+1 = 4$ satisfiability is already NP-complete. Kratochvíl, Savický, and Tuza [17] generalised this sudden jump behaviour to general k: For every fixed $k \geq 3$, satisfiability of $(k, f(k)+1)$-CNF formulas is NP-complete. It may be somewhat intriguing that one can prove such a result, given that we do not even know the values of $f(k)$ for $k \geq 5$; but we will see. Berman, Karpinski, and Scott [15] obtained similar results, showing that for $(k, f(k)+1)$-CNF formulas it is even hard to approximate the maximum number of clauses that can be simultaneously satisfied.

We will also approach the related problems for the size of neighbourhoods and conflict-neigbourhoods. While we can show that the latter performs a similar sudden jump, we have to leave a slack for the neighbourhood bound.

Theorem 9. *Let $k \geq 3$. Then,*

(1) *deciding satisfiability of k-CNF formulas with variable degrees at most $f(k)+1$ is NP-complete (cf.* [17]*),*
(2) *deciding satisfiability of k-CNF formulas with clause neighbourhoods of size at most*[4] $\max\{k+3, l(k)+2\}$ *is NP-complete, and*
(3) *deciding satisfiability of k-CNF formulas with clause conflict-neighbourhoods of size at most $lc(k)+1$ is NP-complete.*

Before engaging in the proof, we describe a general construction that takes a k-CNF formula F and produces a CNF formula $\widehat{F}$ which is satisfiable iff F is satisfiable, so that $\widehat{F}$ is very sparsely interleaved – at the expense of the appearance of 2-clauses. We will later expand these 2-clauses to k-clauses in a fashion tailored to which of the three claims we wish to prove.

We first introduce a useful gadget. Given a set of $j \geq 2$ variables $U = \{x_0, x_1, \ldots, x_{j-1}\}$, the 2-CNF formula

$$\{\{x_0, \overline{x_1}\}, \{x_1, \overline{x_2}\}, \ldots, \{x_{j-2}, \overline{x_{j-1}}\}, \{x_{j-1}, \overline{x_0}\}\}$$

is called an *equaliser of U*; the equaliser of a singleton set U is the empty formula. As it is easily seen, such an equaliser is satisfied by an assignment to U iff all variables in U are mapped to the same value.

Now let F be a k-CNF formula, $k \geq 3$. For each variable $x \in \mathrm{vbl}(F)$, we replace every occurrence (as x or $\overline{x}$) by a new variable inheriting the sign of x in this occurrence. This yields a k-CNF formula F' with $|F|$ clauses over a set of $k|F|$ variables. Moreover, for each variable $x \in \mathrm{vbl}(F)$, we add an equaliser for the set of variables that have replaced occurrences of x. This gives an extra set F'' of at most[5] $k|F|$ 2-clauses. By the property of equalisers, $\widehat{F} := F' \cup F''$ is satisfiable iff F is satisfiable; $\widehat{F}$ can be readily obtained from F in polynomial time. Interleaving is sparse in that

[4] Note that $2^k/e - 1 \geq k+1$ for $k \geq 5$. Therefore, $\max\{k+3, l(k)+2\} = l(k)+2$ in that range and we actually suspect that this holds for all $k \geq 3$.

[5] F'' contains exactly $k|F|$ clauses unless there are variables in F occurring only once.

- every variable of $\mathrm{vbl}(\widehat{F})$ occurs at most 3 times in $\widehat{F}$,
- each k-clause in F' does not share variables with any other clause in F' and the number of its neighbouring 2-clauses in F'' is at most $2k$ – however, at most k of the 2-clauses are in the conflict-neighbourhood –, and
- each 2-clause in F'' neighbours two k-clauses in F' and at most two 2-clauses in F'' (all four clauses may be in the conflict-neigbourhood).

Proof of (1) (variable degrees). Let $k \geq 3$ and fix some minimal (w.r.t. set inclusion) unsatisfiable $(k, f(k)+1)$-CNF formula G. Choose some clause C in G and replace one of its literals by $\overline{x}$ for a new variable x. This new formula, which we denote by $G(x)$, has the property that (i) it is satisfiable (otherwise G would not be minimal), (ii) every satisfying assignment has to set x to 0 (since otherwise G would be satisfiable), (iii) all variables have degree at most $f(k)+1$, and (iv) the newly introduced variable x has degree 1 in $G(x)$.

A reduction from satisfiability of general k-CNF formulas follows. Given such a k-CNF formula F we first generate $\widehat{F}$ as described above. Then we augment each 2-clause in $\widehat{F}$ by $k-2$ positive literals of new variables so that it becomes a k-clause. For each of the new variables x we add a copy of $G(x)$ to our formula; by renaming variables in G these copies are chosen so that their variable sets are pairwise disjoint. The new formula is k-CNF, it is satisfiable iff $\widehat{F}$ is satisfiable. Moreover, the maximum variable degree is $\max\{3, f(k)+1\}$ which is $f(k)+1$, since we assumed $k \geq 3$. This constitutes a polynomial reduction of satisfiability of general k-CNF formulas to satisfiability of k-CNF formulas with maximum variable degree $f(k)+1$. Assertion (1) in Theorem 9 is established. □

Proof of (2) (neighbourhoods). Again, let $k \geq 3$. Fix some minimal unsatisfiable k-CNF formula G where all neighbourhoods have size at most $l(k)+1$. We choose some clause C and replace one of its literals by $\overline{x}$ for a new variable x, resulting in a k-CNF formula $G(x)$ that forces x to 0 in every satisfying assignment.

Starting from a 3-CNF formula F (yes, we mean 3-CNF, not k-CNF) we proceed as before, first producing $\widehat{F}$ consisting of 3- and 2-clauses. Then we augment all clauses in $\widehat{F}$ to k-clauses along with disjoint copies of $G(x)$ for each new variable x. What happened to the neighbourhood sizes? A 3-clause in F' had 6 neighbours in $\widehat{F}$ and gained $k-3$ new neighbours, so there are at most $k+3$. A 2-clause, now extended to a k-clause, had 4 neighbours to begin with and gets an extra neighbour for each of the $k-2$ new literals – which makes $k+2$ neighbours. In a copy $G(x)$ all clauses stay with a neighbourhood of size at most $l(k)+1$ except for the special clause C where we have planted the new literal $\overline{x}$. This clause may now have $l(k)+2$ neighbours. Altogether a bound of $\max\{k+3, l(k)+2\}$ for neighbourhoods holds and the polynomial reduction from satisfiability of general 3-CNF formulas is completed. □

Why did we miss a reduction to k-CNF formulas with neighbourhoods of size at most $l(k)+1$? We could have succeeded, if we had a minimal unsatisfiable k-CNF formula G with neighbourhoods of size at most $l(k)+1$ at our disposal, where at least one clause has a neighbourhood of size at most $l(k)$. Even if all

neighbourhoods had size $l(k)+1$, we would have been happy with one clause C in G which links to some other clause D via a single variable (we could then replace the literal of this variable in C, thereby leaving D as neighbour behind). Fortunately, this idea actually helps when we deal with conflict-neighbourhoods in the next proof.

We will also have to employ equalisers more carefully (wastefully, one might say). Given a variable set $U = \{x_0, x_1, \ldots, x_{j-1}\}$, $j \geq 2$, of concern, let $W = \{z_0, z_1, \ldots, z_{j-1}\}$ be a set of variables disjoint from U. The $(U \cup W)$-equaliser

$$\{\{x_0, \overline{z_0}\}, \{z_0, \overline{x_1}\}, \{x_1, \overline{z_1}\}, \{z_1, \overline{x_2}\}, \ldots, \{z_{j-2}, \overline{x_{j-1}}\}, \{x_{j-1}, \overline{z_{j-1}}\}, \{z_{j-1}, \overline{x_0}\}\}$$

is called a *stretched equaliser of* U – still serving the purpose of forcing all variables in U to the same value. If we use such stretched equalisers in the otherwise identical construction of $\widehat{F}$, we benefit in that

- the 2-clauses in stretched equalisers have a conflict with two other 2-clauses but to at most one of the k-clauses in F'.

Proof of (3) (conflict-neighbourhoods). For $k \geq 3$, fix some minimal unsatisfiable k-CNF formula G where conflict-neighbourhoods have size at most $lc(k)+1$. As before, we want to replace a literal in a clause by a new literal, but now we want to be more careful about where we want to do this. Recall from Lemma 4 that G must have a pair of clauses, C and D, say, which share a unique variable y in a conflicting manner, i.e. $y \in C$ and $\overline{y} \in D$ (there may be other variables in $\mathrm{vbl}(C) \cap \mathrm{vbl}(D)$, but they have to appear with the same sign on either side). So here we do our little surgery: Choose a new variable x and replace y in C by $\overline{x}$. This gives the building block $G(x)$ forcing x to be 0. Note that the clause C' containing $\overline{x}$ (obtained from modifying C) has a conflict-neigbourhood of size at most $lc(k)$ since it lost the conflict-neighbour D.

A reduction from satisfiability of k-CNF formulas now follows in the manner customary. Given F, a k-CNF formula, we move on to $\widehat{F}$ – now with stretched equalisers – and then expand 2-clauses with the help of new variables that are forced to 0 by disjoint copies of $G(x)$. In the final product of this proceeding k-clauses in F' have at most k conflict-neighbours, k-clauses obtained from augmenting 2-clauses have at most $3+(k-2) = k+1$ conflict-neighbours, and, finally, clauses in copies of $G(x)$ do not have conflict-neighbourhoods of size exceeding $lc(k)+1$ due to our careful construction of $G(x)$. That is, the maximum size of a conflict neighbourhood is $\max\{k+1, lc(k)+1\}$ which equals $lc(k)+1$ due to the lower bound of $lc(k) \geq k$ in Lemma 3. □

6 Open Problems

We know $f(k)$, $l(k)$, and $lc(k)$ up to a constant, so one might hope to eventually determine them exactly. (The upper bounds in Theorems 4 and 5 can be improved by a factor of $\frac{63}{64}$ [20]). Progress on the lower bounds we would find very interesting.

Open Problem 1. *Is it possible to improve any of the known lower bounds on $f(k)$, $l(k)$, and $lc(k)$ by a constant factor?*

One possible approach would be to better understand how these functions depend on each other. For example, the current lower bound on $f(k)$ follows by a very simple argument from a lower bound on $l(k)$. Can this relation be improved?

Open Problem 2. *Is there a constant $c_0 > 1$ with $f(k) \geq c_0 l(k)/k$ for k large enough?*

So far, it seems hard to discriminate between $l(k)$ and $lc(k)$.

Open Problem 3. *Is there a constant $c_1 > 1$ such that $l(k) \geq c_1 lc(k)$ for k large enough?*

Note also that we have not even settled the computability of these functions.

Open Problem 4. *Are the functions $f(k)$, $l(k)$ and $lc(k)$ computable?*

While further progress as expressed in the "wish list" above is desirable, one should not forget about the broader picture. Our goal is to find interesting and at the same time simple combinatorial conditions on a CNF formula that entail satisfiability. The bounds on $l(k)$ (and $lc(k)$) express such conditions as degree bounds in the graph of dependencies (and graph of conflicts, resp.) of the clauses of a CNF formula. The consideration of linear CNF formulas imposes a restriction on the quality of dependencies – we have seen that this can make a significant difference in how many clauses are needed for an unsatisfiable k-CNF formula. A possible next step is to combine these two types of criteria.

We conclude with problems that arose in the course of the investigations. Firstly, the CNF formulas providing the best known upper bound on $f(k)$ for k large are obtained via the construction of suitable binary trees. Let $f_{\mathsf{tree}}(k)$ be the largest integer d such that no (k,d)-tree exists. A detour to k-CNF formulas and employment of the Local Lemma shows $f_{\mathsf{tree}}(k) \geq 2^k/(ek) - 2$. A more "direct approach" may allow improvement of this bound.

Open Problem 5. *Is there a constant $c_2 > 1$ such that $f_{\mathsf{tree}}(k) \geq c_2 \frac{2^k}{ek}$ for infinitely many k?*

The known proof of small unsatisfiable linear k-CNF formulas is probabilistic. Known explicit constructions give huge formulas – not an unfamiliar situation.

Open Problem 6. *Give an explicit construction of an unsatisfiable linear k-CNF formula of singly exponential size.*

Acknowledgement. We thank Andreas Razen, Patrick Traxler, and Philipp Zumstein for many helpful discussions on the topic of this survey; Andreas Razen, in particular, for carefully reading the manuscript and many suggestions.

References

1. Alon, N., Spencer, J.H.: The Probabilistic Method, 3rd edn. John Wiley & Sons Inc., Chichester (2008)
2. Erdős, P., Lovász, L.: Problems and results on 3-chromatic hypergraphs and some related questions. In: Hajnal, A., Rado, R., Sós, V.T. (eds.) Infinite and Finite Sets (to Paul Erdős on his 60th birthday), vol. II, pp. 609–627. North-Holland, Amsterdam (1975)
3. Erdős, P.: On a combinatorial problem. Nordisk Mat. Tidskr. 11, 5–10, 40 (1963)
4. Erdős, P.: On a combinatorial problem. II. Acta Math. Acad. Sci. Hungar. 15, 445–447 (1964)
5. Shearer, J.B.: On a problem of Spencer. Combinatorica 5(3), 241–245 (1985)
6. Beck, J.: An algorithmic approach to the Lovász Local Lemma. I. Random Struct. Algorithms 2(4), 343–365 (1991)
7. Alon, N.: A parallel algorithmic version of the local lemma. Random Struct. Algorithms 2(4), 367–378 (1991)
8. Czumaj, A., Scheideler, C.: A new algorithm approach to the general Lovász Local Lemma with applications to scheduling and satisfiability problems. In: Proc. 32nd Ann. ACM Symp. on Theory of Computing, pp. 38–47 (2000)
9. Srinivasan, A.: Improved algorithmic versions of the Lovász Local Lemma. In: Proc. 19th Ann. ACM-SIAM Symp. on Discrete Algorithms, pp. 611–620 (2008)
10. Moser, R.A.: Derandomizing the Lovász Local Lemma more effectively. CoRR **abs/0807.2120** (2008)
11. Moser, R.A.: A constructive proof of the Lovász Local Lemma. CoRR **abs/0810.4812** (2008); Proc. 41st Ann. ACM Symp. on Theory of Computing (to appear)
12. Moser, R.A., Tardos, G.: A constructive proof of the general Lovász Local Lemma. CoRR **abs/0903.0544** (2009)
13. Erdős, P., Spencer, J.: Lopsided Lovász Local Lemma and Latin transversals. Discrete Appl. Math. 30(2-3), 151–154 (1991)
14. Lu, L., Székely, L.: Using Lovász Local Lemma in the space of random injections. Electron. J. Combin. 14(1), 13, Research Paper 63 (2007) (electronic)
15. Berman, P., Karpinski, M., Scott, A.D.: Approximation hardness and satisfiability of bounded occurrence instances of SAT. Electronic Colloquium on Computational Complexity (ECCC) 10(022) (2003)
16. Tovey, C.A.: A simplified NP-complete satisfiability problem. Discrete Appl. Math. 8(1), 85–89 (1984)
17. Kratochvíl, J., Savický, P., Tuza, Z.: One more occurrence of variables makes satisfiability jump from trivial to NP-complete. SIAM J. Comput. 22(1), 203–210 (1993)
18. Savický, P., Sgall, J.: DNF tautologies with a limited number of occurrences of every variable. Theoret. Comput. Sci. 238(1-2), 495–498 (2000)
19. Hoory, S., Szeider, S.: A note on unsatisfiable k-CNF formulas with few occurrences per variable. SIAM J. Discrete Math. 20(2), 523–528 (2006)
20. Gebauer, H.: Disproof of the neighborhood conjecture with implications to SAT. CoRR abs/0904.2541 (2009)
21. Beck, J.: Combinatorial Games: Tic Tac Toe Theory, 1st edn. Encyclopedia of Mathematics and Its Applications, vol. 114. Cambridge University Press, Cambridge (2008)

22. Stříbrná, J.: Between Combinatorics and Formal Logic, Master's Thesis. Charles University, Prague (1994)
23. Hoory, S., Szeider, S.: Computing unsatisfiable k-SAT instances with few occurences per variable. Theoret. Comput. Sci. 337(1-3), 347–359 (2005)
24. Kullmann, O.: New methods for 3-SAT decision and worst-case analysis. Theor. Comput. Sci. 223(1-2), 1–72 (1999)
25. Aharoni, R., Linial, N.: Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. J. Comb. Theory, Ser. A 43(2), 196–204 (1986)
26. Davydov, G., Davydova, I., Kleine Büning, H.: An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF. Ann. Math. Artificial Intelligence 23(3-4), 229–245 (1998)
27. Kleine Büning, H.: An upper bound for minimal resolution refutations. In: Gottlob, G., Grandjean, E., Seyr, K. (eds.) CSL 1998. LNCS, vol. 1584, pp. 171–178. Springer, Heidelberg (1999)
28. Kleine Büning, H.: On subclasses of minimal unsatisfiable formulas. Discrete Appl. Math. 107(1-3), 83–98 (2000)
29. Szeider, S.: Homomorphisms of conjunctive normal forms. Discrete Appl. Math. 130(2), 351–365 (2003)
30. Porschen, S., Speckenmeyer, E., Randerath, B.: On linear CNF formulas. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 212–225. Springer, Heidelberg (2006)
31. Abbott, H.: An application of Ramsey's Theorem to a problem of Erdős and Hajnal. Canad. Math. Bull. 8, 515–517 (1965)
32. Kostochka, A., Mubayi, D., Rödl, V., Tetali, P.: On the chromatic number of set systems. Random Struct. Algorithms 19(2), 87–98 (2001)
33. Scheder, D.: Unsatisfiable linear CNF formulas are large, and difficult to construct explicitely. CoRR abs/0905.1587 (2009)
34. Porschen, S., Speckenmeyer, E., Zhao, X.: Linear CNF formulas and satisfiability. Discrete Appl. Math. 157(5), 1046–1068 (2009)
35. Scheder, D.: Unsatisfiable linear k-CNFs exist, for every k. CoRR abs/0708.2336 (2007)
36. Ben-Sasson, E., Wigderson, A.: Short proofs are narrow—resolution made simple. J. ACM 48(2), 149–169 (2001)