ELSEVIER

Theoretical Computer Science 223 (1999) 1–72

Theoretical Computer Science

www.elsevier.com/locate/tcs

Fundamental Study

New methods for 3-SAT decision and worst-case analysis [1]

O. Kullmann [*]

Johann Wolfgang Goethe-Universität, Fachbereich Mathematik, D-60054 Frankfurt, Germany

Received July 1995; revised January 1998
Communicated by M. Nivat

Abstract

We prove the worst-case upper bound $1.5045..^n$ for the time complexity of 3-SAT decision, where n is the number of variables in the input formula, introducing new methods for the analysis as well as new algorithmic techniques. We add new 2- and 3-clauses, called "blocked clauses", generalizing the extension rule of "Extended Resolution." Our methods for estimating the size of trees lead to a refined measure of formula complexity of 3-clause-sets and can be applied also to arbitrary trees. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: 3-SAT; Worst-case upper bounds; Analysis of algorithms; Extended resolution; Blocked clauses; Generalized Autarkness

Contents

1. Introduction 2
  1.1. Survey of the fundamental ideas so far 3
  1.2. The ideas leading to the bound $1.5044..^n$ 6
  1.3. Applications and further improvements 12
  1.4. Outline of contents 12
2. Notations 14
  2.1. The "language" 14
  2.2. The "semantics" 14
  2.3. Substituting truth values for variables in clause-sets 15
  2.4. Abbreviations 15
3. Blocked clauses 15
4. Generalizations of the Autarkness principle 17
  4.1. Basic Autarkness 17
  4.2. Br-Autarkness 18
  4.3. Generalized Autarkness 19

[*] E-mail: kullmann@mi.informatik.uni-frankfurt.de.
[1] Supported by DFG–Leibniz–Programm Schn 143/5-1.

0304-3975/99/$ - see front matter © 1999 Elsevier Science B.V. All rights reserved.
PII: S0304-3975(98)00017-6

5. Polynomial reductions 19
5.1. The elementary reductions, combined by r_0 19
5.2. “One-step look-ahead” for r_0, and the reduction operator r 21
5.3. Properties of r 21
6. The algorithm 23
6.1. The overall structure of $\mathscr{N}_3$ 23
6.2. Evaluating families of partial assignments with respect to cases of immediate decision or autarkness 25
6.3. Case A: Multiple occurrences of a variable in the 2-clauses 26
6.4. Case B: Variable-disjoint 2-clauses, and a not-blocked 2-clause exists 28
6.5. Case C: The remaining case 29
7. A worst case upper bound 30
8. Estimating tree sizes 31
8.1. Some notations and a basic estimation lemma 31
8.2. Distance functions and our main estimation lemma 32
8.3. Using the τ-function for selecting branchings 34
9. The distance function 36
9.1. The hitherto existing distance functions 36
9.2. The generalized “approximation” of the number of 2-clauses 37
9.3. Using “budgets” for new 2-clauses 38
9.4. The definition of $d^3 = \Delta n - \alpha \cdot \zeta$ 39
9.5. A general upper bound for the numbers of eliminated clauses 40
10. Bounding the sum of d^3 43
11. The two worst cases and the special choices for the parameters 47
11.1. The worst case for “AN” 50
11.2. The worst case for “BN” 51
11.3. The basis of the bound on the number of leaves of the computation tree, depending on the approximation parameter k and the number of new clauses in case “BN” 51
12. Local balancings of distances 52
13. The estimation of $\boldsymbol{\tau_{\max}(d^3, \mathscr{T}_{\mathscr{N}_3}(F_0))}$ 54
13.1. The case “D” of degeneration, and case “C” 55
13.2. The case “AA” 55
13.3. Sufficient conditions for a surplus of at least α 58
13.4. The case “AN” 59
13.5. The case “B” 61
13.6. The final step 65
14. Final remarks 66
14.1. Blocked clauses and Extended Resolution 66
14.2. Comments to our method for estimating the size of trees 68
14.3. General complexity of SAT-decision 68
14.4. Leaving 3-$\mathscr{CLS}$ to obtain faster 3-SAT-decision when the variables occur on the average less than 5.9 times 70
Acknowledgements 71
References 71

1. Introduction

In this paper we study the exponential part of time complexity for 3-SAT decision and prove the worst-case upper bound $1.5044..^n$ for n the number of variables in the input formula, using new algorithmic methods as well as new methods for the analysis. These methods also deepen the already existing approaches in a systematic manner.

The following results for 3-SAT decision (abstracting from a polynomial factor in the length of the input) are known:

1. The history of 3-SAT bounds begins with $1.839..^n$ [24], or, by looking slightly closer at the formula, $1.769..^n$ [3].
2. The first important step was made by Monien and Speckenmeyer [24], yielding the bound $1.618..^n$.[2] The idea is to enforce the existence of a 2-clause, yielding a better output for 1-clause elimination.
3. In [29] this bound was improved to $1.578..^n$ with the help of a refined algorithm and a refined measure of formula complexity for 3-clause-sets (formulas in 3-CNF):

   For the previous bound the mere existence of a 2-clause had been sufficient. Compared to that, [29] starts the quantitative registration of 2-clauses (building up on an idea of B. Monien).
4. While the measure of formula complexity from [29] (invoking a certain "approximation" of the number of 2-clauses) is further developed in this article, another step in our direction is [33], reaching $1.5702..^n$. "Blocked clauses" are used in special cases, eliminating certain branches from the search tree, and also, unlike [29], by some global argumentation a (small) number of new 2-clauses can be taken into account for sure (using also some sort of "budget" for new clauses).
5. The bound $1.5044..^n$ was presented by the author in [16].
6. In the extended abstract [30] the bound $1.4962..^n$ is claimed. In [20] the author of the present article sketched how this bound can be reached by a refinement of the algorithm of this article.

For a general survey on worst-case upper bounds for SAT decision, considering also other classes and other measures of formula complexity, see Section 14.3.1 (and [22]).

1.1. Survey of the fundamental ideas so far

The basic structure of the algorithms used for upper bounds for 3-SAT decision, called "DPLL-algorithms"[3] due to [4, 5], is just "reducing and testing (or branching)": On input F_0, one uses (certain) polynomial reductions $F_0 \rightarrow r(F_0)$ where $r(F_0) =: F$ is satisfiability equivalent to F_0. If this does not decide whether F_0 is satisfiable, one divides via

$$\boxed{\begin{tikzcd}[column sep=small] & F \arrow[dl, dash] \arrow[dr, dash] & \\ r(\varphi_1 * F) & \cdots & r(\varphi_p * F) \end{tikzcd}}$$

$$F \in \mathcal{SAT} \Leftrightarrow \exists\, i \in \{1, \ldots, p\} : \varphi_i * F \in \mathcal{SAT}$$

φ_i partial assignments, $\varphi_i * F$ the result of substituting truth-values via φ_i in F.

[2] The report [24] is published in [25], where the authors additionally introduced the notion of "Autarkness" for the concept behind this improvement. Independently the same bound was also discovered in Luckhardt [23] using a slightly more elementary algorithm and a more compact analysis. The latter seems to us of importance for further progress in this field.

[3] Or "DPLL-like algorithms".

The progress made in the course of development from [24] to our results lies, very roughly speaking, in extending the realm of reductions r, in selecting and constructing better branchings $(\varphi_1, \ldots, \varphi_p)$, and, going hand in hand with these improvements, in discovering new methods for estimating the size of the backtracking tree built up by the above branchings.[4]

For the analysis of complexity the measurement of formula complexity seems to us to be most important, because there is an immediate feedback for the algorithm, which is a "greedy" one and therefore ought to know in which direction the reduction of formula complexity is big.

The basic ideas for the previous bounds (and algorithms) on 3-SAT-decision are as follows. (We assume some basic knowledge about SAT decision here. In Section 2 the reader will find the definitions of the fundamental notions used in this paper.)

1.1.1. The bounds $1.839..^n$ *and* $1.769..^n$

A (partial) assignment φ fulfilling $\varphi(a) = \varphi(b) = \varphi(c) = 0$ for a 3-clause $\{a,b,c\} \in F$ cannot satisfy F, i.e., $\varphi(F) = 0$ holds. In other words: After setting a and b to 0, the value of c is forced to be 1 (by "Unit-clause-elimination").

This trivial observation reduces the search space of assignments for a 3-clause-set F from 2^n to $1.839..^n$. The basis of this bound is given by $1.839.. = \tau(3,2,1)$, where $\tau(3,2,1)$ (see Definition 8.3) is the "cost" of the branching

$$(\langle a \to 0, b \to 0, c \to 1 \rangle,\ \langle a \to 0, b \to 1 \rangle,\ \langle a \to 1 \rangle)$$

with respect to the loss of variables.[5] This "branching tuple" $(3,2,1)$ can be seen as built up from the trivial branching $(1,1)$ and the improved branching $\mathbf{(2,1)}$: $(3,2,1) = (1+\mathbf{2}, 1+\mathbf{1}, 1)$, where the tuple $(1,1)$ comes from testing the literal a and, within branch $\langle a \to 0 \rangle$, the tuple $(2,1)$ comes from testing the (new) 2-clause $\{b,c\}$.

By noticing that also in the other branch $\langle a \to 1 \rangle$ there arises a new 2-clause (for literal a not pure in F), we can (slightly) improve this bound to $\tau(1+2, 1+1, 1+2, 1+1)^n = 1.769..^n$.

1.1.2. The bound $1.618..^n$

A better reduction of the search space is obtained if there is a 2-clause $\{a,b\}$ in F, because then already $\varphi(a) = \varphi(b) = 0 \Rightarrow \varphi(F) = 0$ holds. When the existence of a $\leqslant 2$-clause is always guaranteed, we get the bound $1.618..^n$ $(1.618.. = \tau(2,1))$. For that purpose the following elementary fact is used (a form of the "Autarkness Principle" from [25]):

$$\mathbf{(A_0)}\!:\quad (\varphi * F)^{[0,2]} \neq \emptyset \quad \text{or} \quad \varphi * F \overset{\text{sat}}{\equiv} F$$

[4] Containing only reduced formulas and thus abstracting from the polynomial reduction process.

[5] For $t = (t_1, \ldots, t_n)$ with $t_i > 0$ we define $\tau(t)$ as the solution of $\sum_{i=1}^{n} \tau(t)^{-t_i} = 1,\ \tau(t) > 0$. The τ-function is our basic tool for estimating the size of the computation tree. It enables to valuate a *whole* branching and seems to be also the right tool for that purpose in heuristic algorithms (instead of the use of products and sums like in [10] or [13]). See [19].

for a partial assignment φ and a 3-clause-set F, where $F^{[0,2]}$ is the set of clauses $C \in F$ of length at most 2, and $\overset{\text{sat}}{\equiv}$ is the satisfiability equivalence:

After application of a partial assignment to a 3-clause-set F either there is a clause of length at most 2, or the result is equivalent to F with respect to satisfiability. Thus in the "exceptional case" $(\varphi * F)^{[0,2]} = \emptyset$ we just reduce F to $\varphi * F$ (without branching).

The proof of (A_0) consists in the simple observation that $(\varphi * F)^{[0,2]} = \emptyset \Rightarrow \varphi * F \subseteq F \Rightarrow \varphi * F \overset{\text{sat}}{\equiv} F$ holds (see Section 4). Both algorithms in [23, 25] use this argument, but in a slightly different form: [23] uses (A_0) where F is the original input formula F_0 and φ the assignment corresponding to the *whole path* from the root F_0 of the test-tree to the current test-formula $\varphi * F_0$. [25] stated the stronger version (A_1) of A_0:

$$\textbf{(A}_1\textbf{):}\quad (\varphi * F)^{[0,2]} \backslash F \neq \emptyset \quad \text{or } \varphi * F \overset{\text{sat}}{\equiv} F$$

and applied it to the current test-formula F and one of its *direct successors* $\varphi * F$.

Note that $(\varphi * F)^{[0,2]} \backslash F = (\varphi * F) \backslash F$ holds, since applying a partial assignment to a 3-clause-set does not create new 3-clauses. We use the notation

$$N(\varphi, F) := (\varphi * F) \backslash F$$

for the set of new clauses.

1.1.3. The bound $1.578..^n$

While the *new* $\leqslant 2$-clause guaranteed (in the "normal case") by (A_1) $(N(\varphi, F) \neq \emptyset)$ is not necessary for the previous bound, [29] exploited this effect for his improved bound by including a certain "approximation" of the number of $\leqslant 2$-clauses in his complexity analysis. From our point of view the basis $1.57817.. = \tau_{1,2,1}$ (see Definition 11.2) of the bound in [29] results from balancing the costs of the following two worst cases of his algorithm:

1. In the first "worst case" all 2-clauses have *disjoint variables* and for a specially chosen 2-clause $\{a, b\} \in F$ one splits via $\langle a \to 0, b \to 1\rangle$, $\langle a \to 1\rangle$. (A_1) gives at least one new clause in every branch, and additionally [29] managed to guarantee for the branch $\langle a \to 0, b \to 1\rangle$ one further new 2-clause. This yields the "cost"

   $$\tau(\overbrace{2}^{*_1} + (\overbrace{2}^{*_2} - \overbrace{1}^{*_3}) \cdot \alpha, \overbrace{1}^{*_1} + (\overbrace{1}^{*_2} - \overbrace{1}^{*_3}) \cdot \alpha) = \tau(2 + \alpha, 1)$$

   for this branching, where α is the "weight of 2-clauses."

   The values under $*_1$ stand for the *loss of variables*, under $*_2$ for the number of *new* 2-*clauses* and under $*_3$ for the number of *eliminated* 2-*clauses* in the respective branches.
2. In the second "worst case" some variable *occurs twice* in the 2-clauses of F: $\{a, b\}$, $\{a, c\} \in F$, which is also the maximal number of occurrences of a variable in the 2-clauses of F. For the branching $\langle a \to 0, b \to 1, c \to 1\rangle$, $\langle a \to 1\rangle$ we have:

(a) In branch $\langle a \to 0, b \to 1, c \to 1\rangle$ at most four 2-clauses are eliminated, and because [29] counts only variable-disjoint 2-clauses (an important point which will be discussed later), only two of them are counted. [6]

(b) And in branch $\langle a \to 1\rangle$ two 2-clauses are eliminated, from which one is counted.

Thus for the second worst case we get the cost

$$\tau(3-2\alpha,\ 1-\alpha).$$

The optimal value $\alpha_{1,2,1}$ for α is the value where $\tau(2+\alpha,1)=\tau(3-2\alpha,1-\alpha)$ holds true. Numerical calculations yield $\alpha_{1,2,1}=0.200780..$ and the basis

$$\tau_{1,2,1}=\tau(2+\alpha_{1,2,1},1)=\tau(3-2\alpha_{1,2,1},1-\alpha_{1,2,1})=1.57817..$$

of the bound.

1.2. The ideas leading to the bound $1.5044..^n$

1.2.1. The role of 2-clauses for 3-SAT-decision

The analysis of SAT-algorithms has to gauge (at each step) the reduction in formula complexity achieved by the algorithm.

For the bound $1.618..^n$ only the number n of variables of $F \in 3\text{-}\mathcal{CLS}$ is used for measuring the complexity of F (thus measuring the reduction in formula complexity by Δn, the loss of variables). Now I cannot figure out any way of improving $\Delta n = (2,1)$, achieved by the branching

$$(\langle l \to 0, x \to 1\rangle,\ \langle l \to 1\rangle) \quad \text{for } \{l,x\} \in F,$$

in the case that all 2-clauses are *variable-disjoint* (considering the worst case).

Thus, in order to improve that bound (and the algorithm(!)), *another aspect of structural simplification* than the loss of variables must additionally be taken into account.

Let us consider in the variable-disjoint case the effect of the test-assignments $\varphi_0 = \langle l \to 0, x \to 1\rangle$ and $\varphi_1 = \langle l \to 1\rangle$ on the 3-clauses of F.

If φ_i does not create a new $\leqslant 2$-clause from some 3-clause (by assigning truth value 0 to one or two literals of some 3-clause and not affecting the other literals in that clause), i.e., the set

$$N(\varphi_i, F) = (\varphi_i * F) \setminus F$$

of new clauses is empty, then by (A_1) (see Section 1.1.2) $\varphi_i * F$ is satisfiability equivalent to F. So let us assume that at least *one new* $\leqslant 2$*-clause* has been created by φ_i, i.e., $N(\varphi_i, F) \neq \emptyset$.

[6] [29] treats the case, that in the first "worst case" the new 2-clauses are not variable-disjoint to the existing ones or to each other, by an exhaustive case distinction and including the immediately following branching into the calculation: This incisive complication is eliminated in our algorithm due to of our improved analysis (see Section 1.2.4 in the next). In the second "worst case" new 2-clauses are not counted.

If the empty clause is created ($\bot \in N(\varphi_i, F)$) then this branch is completed. If a new 1-clause arises ($N(\varphi_i, F)^{[1]} \neq \emptyset$) then we are also lucky since by 1-clause-elimination one further variable can be eliminated. So let us assume that only new 2-clauses have been created by φ_i, i.e., $N(\varphi_i, F) = N(\varphi_i, F)^{[2]}$.

Now, since the only existing 2-clause affected by φ_i is $\{l, x\}$ (the 2-clauses are variable-disjoint!), the total number of 2-clauses in F has not been decreased by applying φ_i to F: One 2-clause has been eliminated, and $|N(\varphi_i, F)| \geqslant 1$ many 2-clauses have been created.

If in at least one branch φ_0 or φ_1 also an *additional new* 2-*clause* can be guaranteed, that is $|N(\varphi_i, F)| \geqslant 2$, then in fact in that branch the *number of* 2-*clauses increases*!

This increase could be exploited directly by considering the following two cases (for simplicity we assume that in *both* branches two new 2-clauses are created):

1. The set of 2-clauses stays variable-disjoint: This case occurs maximally $n/2$-times on any path in the backtracking tree (or computation tree as we say), and thus the computation tree has been cut down.[7]
2. Otherwise there exists a variable which occur at least twice in the 2-clauses of F, and testing such a variable yields (due to 1-clause-elimination) a bigger reduction with respect to n than $\Delta n = (2, 1)$.[8]

However, this approach complicates the algorithm and ignores "distant effects." (The reader should note that the case-distinction in only "locally complete," i.e., for one single node, while in general both cases may occur together in the computation tree, and thus prevent the simple bounds.)

We choose another more general approach, combining both sights (the elimination of variables vs. the increase in the number of (variable-disjoint) 2-clauses) by some sort of linear combination.

1.2.2. The basic idea of refined measurement for 3-clause-sets, incorporating some "compatible" amount of 2-clauses into the measure

As starting point we refine the measure n (number of variables) for the complexity of F to a difference $\boldsymbol{m} = \boldsymbol{n} - \boldsymbol{z}$ where z reflects in a suitable way the number of $\leqslant 2$-clauses of F. To determine this "suitable way" is a non-trivial task and is treated in this work the first time systematically, motivated by an analogous quantity within the proofs from [29].

Some remarks:

– z has to be subtracted from n since an *increase* in the number of 2-clauses should *decrease* the formula complexity.
– In our framework it is preferable to incorporate into the measure z only the number of 2-clauses instead of $\leqslant 2$-clauses, since the empty clause $\bot$ simply aborts the current path in the computation tree and 1-clauses belong to the difference in the number of variables (1-clause-elimination is applied always automatically).

[7] If this case would be guaranteed for the whole tree, one gets (at least) the bound $\sqrt{2}^n$.

[8] And if this case would generally hold, the bound $\tau(3, 1)^n = 1.465..^n$ would be established.

– z has to be an "approximation" of the number of 2-clauses since the number of 2-clauses itself can be quadratically in n, but a measure (of formula complexity) should take only non-negative values.

The "progress" made in the test-tree from F to one direct successor F', which, taken together, determines our bound for the size of the test-tree (with the help of the τ-function), is now no longer the simple difference $\Delta n = n(F) - n(F')$, but

$$\Delta m = \Delta n - \Delta z = (n(F) - n(F')) - (z(F) - z(F')).$$

The reduction of F is the bigger the greater Δm is, i.e., the greater $\Delta n = n(F) - n(F')$ and the less $\Delta z = z(F) - z(F')$ (negative if possible) are.

In the whole we try to improve the bound $1.618..^n$ by using negative Δz in the case of "small" Δn (mainly the case of variable disjoint 2-clauses), corresponding to an increase in the number of 2-clauses. In the case of "big" Δn, the above combined difference Δm (in contrast) is decreased by a positive Δz (more 2-clauses vanish than arise), and thus the whole method can be seen as a *balancing of the measurements of the different cases*.

The following trivial equation is basic for our considerations:

$$\Delta z = \text{number of vanishing “}z\text{-clauses”} - \text{number of new “}z\text{-clauses”}.$$

Thus we have to *maximize* the number of new "z-clauses" and to *minimize* (control) the number of vanishing "z-clauses," where by a "z-clause" we mean a 2-clause which is counted by the approximation z.[9]

1.2.3. New 2-clauses

To obtain a maximal number of *new* "z-clauses" we use the following two new methods:

1. We introduce the notion of "*Blocked Clauses*," a generalization of the addition of new clauses by the extension rule of "Extended Resolution." "Blocked Clauses" are special (fast decidable) cases of "redundant clauses," i.e., clauses which can be satisfiability equivalently added to or eliminated from a given clause set. For testing a clause-set F with variable disjoint 2-clauses we use blocked clauses to obtain new 2-clauses in the following ways:
   (a) We choose the branching variable from a *not-blocked* 2-*clause*, because this has a greater impact on the clause-set (i.e., yields more new 2-clauses).
   (b) Blocked 2-clauses are "virtually" eliminated in the "*br-Autarkness*"-*case* of our algorithm, a generalization of Autarkness, in order to establish the existence of the not-blocked 2-clause in (a).
   (c) Such *blocked* 3-*clauses* (without new variables) are *added* to F in some cases, which become 2-clauses after branching.

[9] Although this is only an "illusion," since the 2-clauses counted by z are not materially fixed, this picture is very helpful.

(d) In order to improve the efficiency of (c), in F all blocked 3-clauses are *eliminated* (before).

(e) All possible *blocked* 2-*clauses* (without new variables) have been *added* to F (before), since they reveal better branching possibilities.

A predecessor of the notion of "Blocked Clauses" is Purdom's "Complement Search" in [28]: In a more general context conditions for eliminating certain branches from the test tree are given, which correspond (for CNF) to the effect of addition of blocked clauses and subsequent 1-clause-eliminations (compare (c) and (e)).

2. If a partial assignment φ creates only "few" new 2-clauses, i.e., $N(\varphi, F)$ is "small," then with the concept of "*Generalized Autarkness*" we construct from φ a new branching, which, in case the old envisaged one (to which φ belonged to) was not already "good",[10] is better than this old one. (Ordinary) "Autarkness" is just the special case where zero new 2-clauses are created.

1.2.4. How new 2-clauses are taken into account

The problem that a new 2-clause in general is not a new "*z-clause*" (because z is only an approximation of the total number of 2-clauses), and *we do not know which clauses* are taken into account and which are not, is overcome by a general method, tracing the new 2-clauses over the whole computation tree.

The "distance function" d^3. We introduce the *distance function* $d^3(F, F')$ (replacing $\Delta n - \Delta z$[11]), depending on a "level of approximation" k. Let $\rho(F)$ denote the maximal number of occurrences of a variable in the 2-clauses of F. We distinguish two main cases:

1. If in F the number of occurrences of variables in the 2-clauses is *bounded by* k (i.e., $\rho(F) \leqslant k$), then d^3 allows to *account new* 2-*clauses up to a fixed amount*.
2. Otherwise if $\rho(F) > k$, then *only vanishing* 2-*clauses are counted*. (Motivated by the fact, that in this case "*many*" *variables vanish* because of 1-clause-eliminations: At least $\rho(F) + 2$ together in both branches by testing a literal which realizes $\rho(F)$.)

The number of vanishing 2-clauses which are counted by d^3 is (in any case) *bounded by a function of* Δn *and* k, since d^3 considers only (maximal) sub-clause-sets $P \subseteq F^{[2]}$ of the set of 2-clauses of F with $\rho(P) \leqslant k$[12] (plus a "reserve" for new 2-clauses).

The main idea for the realization of d^3*: Consider the whole subsequent computation.* Consider a clause-set F with $\rho(F) \leqslant k$ and one of its immediate successors F' in the computation tree.

If also $\rho(F') \leqslant k$ holds, then all new 2-clauses (arising at F' and new relative to F) are counted for this branch, since for $\rho \leqslant k$ the above P contains all 2-clauses.

Otherwise consider the first successor F^* of F' in the computation tree with $\rho(F^*) \leqslant k$. Because we did not count new 2-clauses at the intermediate points

[10] W.r.t. $\tau(\Delta n)$, where Δn stands for the whole tuple of differences, each single difference corresponding to one branch of the branching we consider at this point.

[11] $d^3(F, F')$ depends on F and F' in a more general way than being a difference of measure values for clause-sets.

[12] This generalizes the consideration of variable disjoint 2-clauses in [29] (there $k = 1$).

between F and F^*, the number of new 2-clauses could not accumulate and at last at F^* the number of new 2-clauses at F must have been taken into account! [13]

The framework for the realization of d^3. The framework for this more global attempt, compared to what can be achieved by using the ordinary methods of recursion equations, is given by the "τ-lemma" which enables us to estimate the number of leaves in a tree with the help of an *arbitrary distance function*, i.e., a labeling of the edges of the tree by positive real numbers (using necessarily the *original* tree, not the one belonging to the (local) recursion equations).

The calculation of the upper bound on the number of leaves consists in the calculation of the maximal τ-value over all (inner) nodes of the tree, constituting the basis of the (exponential) bound, and in calculating the maximal sum of the distances over all paths of the tree, constituting the exponent of the bound.

Every tree has (up to a positive factor) exactly one distance function such that the bound is exact. This optimal distance function is characterized by the condition that all τ-values over inner nodes of the tree are equal, motivating our general strategy of balancing of different cases. [14]

Within this framework, the above idea, that new 2-clauses must *eventually* have been taken into account, could already be involved in the definition of d^3, and the main problem left is to bound the maximal sum of d^3-values over all paths in the computation tree by a (reasonable) function of n.

1.2.5. Controlling the number of vanishing 2-clauses

Back to our problem of minimizing Δz (from Section 1.2.2), we have the following principles to control the number of *vanishing* 2-clauses:

1. The "basic test" consists of the two branches $[\langle l \to b\rangle]_F$ for $b \in \{0,1\}$ for some literal l, where $[\ \ ]_F$ means the addition of all possible 1-clause-eliminations. If in one branch b "many" 2-*clauses* are eliminated, then in at least one of the two branches $b, \overline{b}$ also "many" *variables* must have been eliminated, since if in branch b only "few" variables are eliminated, then one of the eliminated variables of branch b must occur "often" in the 2-clauses of F, and now, according to the special choice of l as a literal such that the sum of occurrences of l and $\overline{l}$ in the 2-clauses of F is *maximal*, [15] l or $\overline{l}$ itself occurs "often" in the 2-clauses of F with the consequence that in the other branch $\overline{b}$ "many" variables must disappear. The case of a symmetrical distribution of the number of disappearing variables on both branches is even more favorable (owing to a fundamental property of the τ-function).

[13] If these new 2-clauses in fact vanished at an intermediate point, we distinguish three cases: If they vanished inside the respective P they already have been taken into account, but if they vanish outside of the respective P then either they are replaced by new 2-clauses or their loss must be compensated with the help of a bigger Δn (for this case we need the boundedness of the number of new 2-clauses which are taken into account).

[14] Also it is necessary that all sums of distances along a path from the root to a leaf are the same. But this condition is, in our context, more or less "automatically" fulfilled.

[15] I.e., ρ is attained for l.

2. According to 1. the number of vanishing 2-clauses is connected to the number of eliminated variables. However, the bound is quadratically,[16] and furthermore only holds for the whole branching, but not for one single branch. To control a single branch and to optimize the estimation we use the approximation mentioned above (choosing at the end k and the weight α optimal).

1.2.6. Bringing together the different ideas

In the algorithm as well as in the analysis we have to combine all the above ideas in an optimal way ("optimal" according to our knowledge of the shape of the "general clause-set"). This combination, achieved with the help of d^3, has to balance the mentioned cases:

For variable disjoint 2-clauses small Δn, but an increase in the number of 2-clauses, and in the case of a multiple occurrence of a variable in the 2-clauses "big" Δn, but (in general) a *decreasing* number of 2-clauses.

This job seems not to be straightforward, and some effort is necessary to obtain global tools (like d^3) for the proof of the upper bound which work in *any case*.

Last, but not least, is should be mentioned that our decision algorithm also uses certain polynomial reductions which help us to get rid off some special ill-conditioned cases (establishing some "normal form").

1.2.7. Where the number 1.5044.. *comes from*

Analogously to our analysis of the bound $1.578..^n$ from [29], we give the two worst cases of our algorithm and show how the bound is computed:

1. Again, in the first "worst case" all 2-clauses have disjoint variables and for a specially chosen 2-clause $\{l,x\} \in F$ one splits via $\langle l \to 0, x \to 1\rangle$, $\langle l \to 1\rangle$. Our algorithm now achieves *three new* 2-*clauses* (in the "normal case") for both branches and thus we obtain the "cost"

$$\tau(\overbrace{2}^{*_1} + (\overbrace{3}^{*_2} - \overbrace{1}^{*_3}) \cdot \alpha, \overbrace{1}^{*_1} + (\overbrace{3}^{*_2} - \overbrace{1}^{*_3}) \cdot \alpha) = \tau(2+2\alpha,\, 1+2\alpha)$$

for this branching, where $*_1$, $*_2$ and $*_3$ are as before.

2. Due to our (optimal) choice $k=2$ for the parameter k, in the second "worst case" a literal l occurs three $(=k+1)$ times in the 2-clauses of $F: \{l,x_1\}$, $\{l,x_2\}$, $\{l,x_3\} \in F$, which is also the maximal number of occurrences of a variable in the 2-clauses of $F(\rho(F)=3=k+1)$. For the branching

$$(\langle l \to 0, x_1 \to 1, x_2 \to 1, x_3 \to 1\rangle, \langle l \to 1\rangle)$$

[16] We are only able to handle linear bounds, because we need a constant weight for the (approximation of the) number of 2-clauses in the combined measure.

we have

(a) In branch $\langle l \to 0, x_1 \to 1, x_2 \to 1, x_3 \to 1\rangle$ at most nine $(=(k+1)^2)$ 2-clauses are eliminated, and since we consider only the elimination of 2-clauses from a (maximal) set P of 2-clauses of F with $\rho(P) \leqslant k = 2$, only six $(=(k+1)\cdot k)$ of them are counted;

(b) In branch $\langle l \to 1\rangle$ three 2-clauses are eliminated, from which $2(=k)$ are counted.

Thus for the second worst case we get the cost

$$\tau(4-6\alpha,\ 1-2\alpha).$$

The optimal value $\alpha_{2,3,3}$ for α is the value where $\tau(2+2\alpha, 1+2\alpha) = \tau(4-6\alpha, 1-2\alpha)$ holds true. We obtain $\alpha_{2,3,3} = 0.12393..$ and the basis

$$\tau_{2,3,3} = \tau(2+2\alpha_{1,2,1}, 1+2\alpha_{1,2,1}) = \tau(4-6\alpha_{1,2,1}, 1-2\alpha_{1,2,1}) = 1.50443..$$

of the bound.

1.3. Applications and further improvements

The SAT algorithm presented in this paper has not yet been implemented. However, it seems to be more likely that the complete algorithm is of more theoretical interest, while practical applications may result from heuristic versions of the involved ideas (see [19]).

That transformation of general formulas into 3-CNF indeed can be of practical importance has been demonstrated by the patent [31] and its commercializing. [17]

For a survey on SAT algorithms and applications see [11].

I decided not to include the (relatively small) improvements of the bounds indicated by [30], since the effort seems to be disproportionate to me. See [20], where it is shown how to refine the algorithm of this paper to obtain the bound claimed in [30].

I believe that further progress, below the bound 1.49^n, is only possible when finding a general structure in these refinements.

1.4. Outline of contents

After introducing the basic notations in Section 2 we present the concept of "Blocked Clauses" in Section 3 and two generalizations of the "Autarkness Principle" in Section 4.

The polynomial reductions used in our 3-SAT algorithm $\mathcal{N}_3$ are the subject of Section 5, while $\mathcal{N}_3$ is presented in Section 6.

The analysis of $\mathcal{N}_3$ is the subject of Sections 7–13. The basis of our analysis is the "τ-Lemma" 8.2 in Section 8, enabling us to calculate a bound on the number of leaves for any tree T, given a "distance function" d, an arbitrary labeling of the edges of T by positive real numbers.

[17] In essence the patent is claimed for the (standard) method of transforming arbitrary propositional formulas into 3-CNF, already mentioned in [32]!

The choice of the distance function for the "computation tree" $\mathscr{T}_{\mathscr{N}_3}(F_0)$ is the subject of Section 9. The first attempt is to use $d^0 = \Delta n$, the loss of variables (used in [23, 25]). [29] used (implicitly) the improved distance function $d^1 = \Delta m_1 = \Delta n - \alpha \cdot \Delta z_1$ where z_1 is the maximal number of variable-disjoint 2-clauses in the clause-set and $\alpha \in \mathbb{R}_+$ is a parameter to be chosen optimally.

In Section 9.2 we generalize z_1 to z_k for $k \in \mathbb{N}_0 \cup \{+\infty\}$, where $z_k(F)$ is the maximal size of a set of 2-clauses of F such that every variable occurs at most k times:

$$z_k(F) := \max\{\, |P| : P \subseteq F^{[2]} \wedge \rho(P) \leqslant k \,\},$$

where $\rho(P) = \max_{v \in \operatorname{Var}(P)} |\{C \in P : v \in C \vee \overline{v} \in C\}|$ is the maximal number of occurrences of a variable in the 2-clauses of F.

Although $d^2 = \Delta n - \alpha \cdot \Delta z_k$ is already an improvement over $d^1 = \Delta n - \alpha \cdot \Delta z_1$, since we can optimize k,[18] it is not the end of the story, because z_k counts only *certain* 2-clauses and thus new 2-clauses may *not* increase $\Delta n - \alpha \cdot \Delta z_k$.

A solution for this problem is given in Subsection 9.3, where we introduce the distance function d^3 for $\mathscr{T}_{\mathscr{N}_3}(F_0)$. By using "budgets," d^3 admits to credit an *increase* in the number of 2-clauses for sure, but only to a certain amount, allowing on the other side to restrict the negative effect of a *decreasing* number of 2-clauses.

Of central importance for the application of d^3, in order to obtain a bound in the input parameter $n(F_0)$, is that the maximal sum of d^3-values over all paths in $\mathscr{T}_{\mathscr{N}_3}(F_0)$ is reasonably connected to $n(F_0)$, which is treated in Section 10.

The special (optimal) values for the parameters of d^3 (the size of the "budgets," the "level of approximation" for the 2-clauses, and their "weight") are determined in Section 11. Eventually the final estimations of the τ-values with respect to the distance function d are done in Sections 12 and 13, containing most of the combinatorial properties of $\mathscr{N}_3$.

In the "Final Remarks" we discuss the following topics:
- the connection of the addition of blocked clauses with "Extended Resolution";
- some further results on our (general) method for estimating the size of trees;
- the general (time) complexity of SAT-decision (presenting the other known bound for SAT decision, and recognizing the complexity of "(3, 2)-SSS" (a generalization of 3-coloring, see [1, 2]) as a 3-SAT complexity with respect to a special measurement of formula complexity (namely the number of 3-clauses));
- and at last, we combine the 3-SAT-decision algorithm $\mathscr{N}_3$ with the SAT-decision algorithm from [22] realizing the upper bound $2^{1/10 \cdot \ell}$ for the number ℓ of literal occurrences, and improve the bound 1.5045^n for all 3-clause-sets F for which the average number ℓ / n of occurrences of literals is less than 5.9.

[18] In [23, 25] $k = 0$ was optimal, in [29] $k = 1$ and here $k = 2$.

2. Notations

2.1. The "language"

Let $\mathcal{VA}$ be a non-empty set (of "*variables*") and $\mathcal{LIT} := \mathcal{VA} \uplus \{\overline{v} : v \in \mathcal{VA}\}$ the set of *literals*. Let $\overline{\boldsymbol{l}}$ denote the *complement* and $\mathbf{Var}(\boldsymbol{l}) \in \mathcal{VA}$ the *underlying variable* of $l \in \mathcal{LIT}$. For $L \subseteq \mathcal{LIT}$ we use $\overline{\boldsymbol{L}} := \{\overline{l} : l \in L\}$ and $\mathbf{Var}(\boldsymbol{L}) := \{\mathrm{Var}(l) : l \in L\}$.

A *clause* is a finite and complement-free set of literals, the set of all clauses is denoted by $\mathcal{CL} := \{C \subseteq \mathcal{LIT} : C \text{ finite} \wedge C \cap \overline{C} = \emptyset\}$.

$\mathcal{CLS} := \{F \subseteq \mathcal{CL} : F \text{ finite}\}$ is the set of all *clause-sets*. We use $\mathbf{Lit}(\boldsymbol{F}) := \bigcup_{C \in F} C$ and $\mathbf{Var}(\boldsymbol{F}) := \bigcup_{C \in F} \mathrm{Var}(C)$ for $F \in \mathcal{CLS}$.

A special clause is the *empty clause* $\bot$ ($\bot := \emptyset \in \mathcal{CL}$) and a special clause-set is the *empty clause-set* $\top$ ($\top := \emptyset \in \mathcal{CLS}$).

A $\boldsymbol{p}$**-clause** C is a clause of length $p \in \mathbb{N}_0$: $|C| = p$, while a $\leqslant\boldsymbol{p}$**-clause** is a clause of length at most p.

For $F \in \mathcal{CLS}$, $i, j \in \mathbb{Z}$ let $\boldsymbol{F}^{[i,j]} := \{C \in F : i \leqslant |C| \leqslant j\}$ denote the sub-clause-set of F of all clauses with length between i and j; $\boldsymbol{F}^{[i]} := F^{[i,i]}$. For $p \in \mathbb{Z}$ let $\boldsymbol{p}\text{-}\mathcal{CLS} := \{F \in \mathcal{CLS} : F = F^{[0,p]}\}$ denote the set of all *p-clause-sets*.

For $l \in \mathcal{LIT}$ and $F \in \mathcal{CLS}$ we define $\#_{\boldsymbol{l}}(\boldsymbol{F})$ as the number of occurrences of l in $F : \#_l(\boldsymbol{F}) := |\{C \in F : l \in C\}|$, and $\#^i_{\boldsymbol{l}}(\boldsymbol{F}) := \#_l(F^{[i]})$ as the number of occurrences of l in the i-clauses of F, while $\boldsymbol{\ell}(\boldsymbol{F}) := \sum_{l \in \mathcal{LIT}} \#_l(F) = \sum_{l \in \mathcal{LIT}} \sum_{i=1}^{\infty} \#^i_l(F)$ is the number of literal occurrences in F at all.

And by $\boldsymbol{n}(\boldsymbol{F}) := |\mathrm{Var}(F)|$ we denote the number of variables in F.

2.2. The "semantics"

A *partial assignment* is a mapping $\varphi : L \to \{0, 1\}$ ($L = \mathrm{dom}(\varphi)$) with $L \subseteq \mathcal{LIT}$ and $\overline{L} = L$ (L is closed under complement) such that φ preserves complements:

$$\forall l \in L : \varphi(\overline{l}) = \overline{\varphi(l)} \quad (\overline{0} = 1, \overline{1} = 0).$$

The set of all partial assignments is $\mathcal{PASS}$.

We write "$\varphi(l) = \ldots$" iff $l \in \mathrm{dom}(\varphi)$. For $\boldsymbol{\varphi} \in \mathcal{PASS}$ we define: $\mathbf{Var}(\boldsymbol{\varphi}) := \mathrm{Var}(\mathrm{dom}(\varphi))$, and $\boldsymbol{n}(\boldsymbol{\varphi}) := |\mathrm{Var}(\varphi)|$.

We extend $\boldsymbol{\varphi}$ to clauses and clause-sets in the natural way ($\boldsymbol{\varphi} \in \mathcal{PASS}$, $C \in \mathcal{CL}$, $F \in \mathcal{CLS}$):

$$\boldsymbol{\varphi}(\boldsymbol{C}) := \begin{cases} 0 & \text{if } \forall l \in C : \varphi(l) = 0, \\ 1 & \text{if } \exists l \in C : \varphi(l) = 1, \\ \text{undefined} & \text{else;} \end{cases}$$

$$\boldsymbol{\varphi}(\boldsymbol{F}) := \begin{cases} 0 & \text{if } \exists C \in F : \varphi(C) = 0, \\ 1 & \text{if } \forall C \in F : \varphi(C) = 1, \\ \text{undefined} & \text{else;} \end{cases}$$

$\mathcal{SAT} := \{F \in \mathcal{CLS} : \exists \varphi \in \mathcal{PASS} : \varphi(F) = 1\}$ (trivially $\top \in \mathcal{SAT}$, $\{\bot\} \in \mathcal{CLS} \setminus \mathcal{SAT}$).

The equivalence relation corresponding to the partition $\{\mathcal{SAT}, \mathcal{CLS} \setminus \mathcal{SAT}\}$ is the *satisfiability equivalence*: $F \overset{\text{sat}}{\equiv} F'$ iff either F and F' are both satisfiable or both unsatisfiable.

For the purpose of defining special partial assignments (as sets of ordered pairs) we use for literals $l_1, \ldots, l_s \in \mathcal{LIT}$ and truth values $\varepsilon_1, \ldots, \varepsilon_s \in \{0,1\}$:

$$\langle \boldsymbol{l_1 \to \varepsilon_1, \ldots, l_s \to \varepsilon_s} \rangle := \{(l_i, \varepsilon_i), (\overline{l_i}, \overline{\varepsilon_i}) : 1 \leqslant i \leqslant s\}.$$

2.3. Substituting truth values for variables in clause-sets

We substitute truth values for variables in clauses $C \in \mathcal{CL}$ and clause-sets $F \in \mathcal{CLS}$ via partial assignments $\boldsymbol{\varphi} \in \mathcal{PASS}$:

$$\boldsymbol{\varphi * C} := C \setminus \varphi^{-1}(\{0\}) \quad \text{for } \varphi(C) \neq 1;$$

$$\boldsymbol{\varphi * F} := \{\varphi * C : C \in F \text{ and } \varphi(C) \neq 1\}.$$

$\varphi * F$ emerges from F by eliminating all clauses which become true via φ and eliminating all remaining literals which become false via φ.

The basic properties are: $\varphi * F \in \mathcal{CLS}$, $\varphi(F) = 1 \Leftrightarrow \varphi * F = \top$ and $\varphi(F) = 0 \Leftrightarrow \bot \in \varphi * F$.

For the purpose of SAT-decision we have for a literal $l \in \mathcal{LIT}$:

$$F \in \mathcal{SAT} \Leftrightarrow (\langle l \to 0 \rangle * F \in \mathcal{SAT} \text{ or } \langle l \to 1 \rangle * F \in \mathcal{SAT}).$$

The use of the notation "$\varphi * F$" reflects that here *two* objects are involved: the clause-set F and the partial assignment φ, indicating that certain calculation are done with φ itself. The sign "$*$" is chosen analogously to the notation of scalar multiplication in Linear Algebra (the semi-group $\mathcal{PASS}$ acts on the set $\mathcal{CLS}$ as a semi-module).

2.4. Abbreviations

We use "*w.r.t.*" for "with respect to," "*s.t.*" for "such that" and "*w.l.o.g.*" for "without loss of generality."

3. Blocked clauses

A new concept is the concept of a **"Blocked Clause"** w.r.t. a given clause-set.[19] A blocked clause is a special (fastly recognizable) case of a "redundant clause", that means a clause which can be sat-equivalently eliminated from or adjoined to the given clause-set.

The effects of blocked clauses are

1. Eliminating blocked clauses (generalization of the elimination of pure literals) causes that the remaining clauses are "stronger linked to each other."

[19] A predecessor is [28], where the concept has been formulated in terms of eliminating branches in DPLL-algorithms under certain circumstances.

2. Addition of blocked clauses (without new variables) eliminates some branches from the search-tree (or computation tree), and may also increase our measure for the number of 2-clauses.

Since addition of blocked clauses generalizes the Extension Rule of [32], it is also helpful in analyzing Extended Resolution (see [9, 18, 21]). In Section 14.1 we discuss this relationship a bit closer.

Definition 3.1. A clause $C \in \mathscr{CL}$ is called *blocked for* $l \in \mathscr{LIT}$ *w.r.t.* $F \in \mathscr{CLS}$ iff $l \in C$ holds and for all $C' \in F$ we have $C \cap \overline{C'} \neq \{l\}$.

A clause C is called *blocked w.r.t.* F iff there is a literal l such that C is blocked for l w.r.t. F.

C is blocked for $l \in C$ w.r.t. F iff every clause of F either does not contain $\overline{l}$ or additionally contains another complementary literal (i.e., its resolvent with C is tautological).

C is *not* blocked w.r.t. F iff for all $l \in C$ there is $C' \in F$ with $C \cap \overline{C'} = \{l\}$.

Lemma 3.1 (Blocking-Lemma). *For $F \in \mathscr{CLS}$ and $C \in \mathscr{CL}$ blocked w.r.t. F we have*: $F \cup \{C\} \overset{\text{sat}}{\equiv} F \overset{\text{sat}}{\equiv} F \setminus \{C\}$.

Proof. It is enough to show $F \overset{\text{sat}}{\equiv} F \setminus \{C\}$ for $C \in F$. The direction $F \in \mathscr{SAT} \Rightarrow F \setminus \{C\} \in \mathscr{SAT}$ is obvious.

Consider $\varphi \in \mathscr{PASS}$ with $\varphi(F \setminus \{C\}) = 1$. W.l.o.g.: $\mathrm{Var}(\varphi) = \mathrm{Var}(F \cup \{C\})$. If $\varphi(C) = 1$ then immediately also $\varphi(F) = 1$. Otherwise, let C be blocked for l w.r.t. F. We define φ' by flipping the value of l:

$$\varphi' := (\varphi \setminus \langle l \to 0 \rangle) \cup \langle l \to 1 \rangle.$$

Now $\varphi'(F) = 1$ holds because on the one hand we have $\varphi'(l) = 1 \Rightarrow \varphi'(C) = 1$, and on the other hand $\varphi'(C') = 1$ holds for $C' \in F \setminus \{C\}$ due to:
- if $\overline{l} \notin C'$, then $\varphi'(C') = 1$ (because of the definition of φ' and $\varphi(C') = 1$);
- if $\overline{l} \in C'$ then there is another $a \in C \setminus \{l\}$ with $\overline{a} \in C'$ (because of the blocking condition) and by $\varphi'(a) = \varphi(a) = 0$ we have $\varphi'(C') = 1$ as well. □

We conclude this section by introducing special notions for blocked clauses without new variables:

Definition 3.2. For $F \in \mathscr{CLS}$ and $l \in \mathscr{LIT}$:

$$\boldsymbol{B(F)} := \{C \in \mathscr{CL} : \mathrm{Var}(C) \subseteq \mathrm{Var}(F) \wedge C \text{ blocked w.r.t. } F\}$$

$$\boldsymbol{B_l(F)} := \{C \in B(F) : C \text{ blocked for } l \text{ w.r.t. } F\}.$$

By iterating Lemma 3.1 we easily obtain

Lemma 3.2. *For $F \in \mathscr{CLS}$ and $l \in \mathscr{LIT}$ we have: $F \overset{\text{sat}}{\equiv} F \cup B_l(F)$.*

4. Generalizations of the Autarkness principle

4.1. Basic Autarkness

The key observation (with trivial proof) leading to the first non-trivial bound "$1.618..^n$" for 3-SAT-decision in [25] (also [24]), called "Autarkness Principle," is the (easy) statement that when applying a partial assignment to a 3-clause-set F either a *new* $\leqslant 2$-clause is created, or the resulting clause-set is sat-equivalent to F (see Lemma 4.1).

Independently, this upper bound has been obtained also in [23] using a weaker form of Autarkness: In the process of testing there is either a $\leqslant 2$-*clause at all* in the current 3-clause-set F (resulting from the original input F_{input} by a series of applications of partial assignments), or F is sat-equivalent to F_{input}.[20]

The *new* $\leqslant 2$-clause guaranteed by the Autarkness Principle in fact is firstly employed in [29] for the improved upper bound "$1.578..^{n(F_0)}$." We will strengthen the creation of at least one new 2-clause both qualitatively (more from the new 2-clause is known) and quantitatively (more than one new 2-clause is created). We start with a reformulation of the Autarkness Principle from [25].

Definition 4.1. A partial assignment $\varphi \in \mathscr{PASS}$ is called *autark* for a clause-set $F \in \mathscr{CLS}$ iff $\varphi * F \subseteq F$ holds (i.e., φ makes all clauses come true which are affected by it[21]).

Lemma 4.1 ("Autarkness–Lemma", cf. [25]). *Consider a partial assignment $\varphi \in \mathscr{PASS}$ and a clause-set $F \in \mathscr{CLS}$.*

1. *Autark assignments can be applied sat-equivalently*:

$$\varphi \text{ autark for } F \Rightarrow F \overset{\text{sat}}{\equiv} \varphi * F.$$

2. *A partial assignment is autark for a p-clause-set iff it does not create a new clause* (*which must be of length at most $p-1$*):

$$F \in p\text{-}\mathscr{CLS} \Rightarrow (\varphi \text{ autark } \Leftrightarrow (\varphi * F)^{[0,p-1]} \setminus F = \emptyset).$$

The lemma is used as follows for SAT decision:

In each projected branching $(\varphi_0, \ldots, \varphi_m)$ for a clause-set F[22] we search whether there exists φ_{i_0} which is autark for F – in this case we do not have to branch but can immediately reduce F to $\varphi_{i_0} * F$.

Otherwise we know that in every branch there must be a new clause (which in case of $F \in p\text{-}\mathscr{CLS}$ must be of length $\leqslant p-1$).[23]

[20] See the discussion of differences between [23] and [25] in [22].

[21] More precisely: $\forall C \in F : \varphi(C) \neq 1 \Rightarrow \varphi * C \in F$.

[22] I.e., F is divided into the subproblems $\varphi_0 * F, \ldots, \varphi_m * F$.

[23] For 3-SAT decision in this way the bound $1.618..^n = \tau(2,1)^n$ (see Definition 8.3) is established, improving the trivial bound $1.839..^n = \tau(3,2,1)^n$ for 3-SAT-decision (which is still the best known for *counting* satisfying assignments).

We generalize Lemma 4.1 in two directions:

– We want to guarantee not only the existence of an *arbitrary* (new) $\leqslant 2$-clause but of a *not blocked* (new) $\leqslant 2$-clause, and this not only in $\varphi * F$ but in the *reduced* $r(\varphi * F)$ (see Section 4.2).
– We want to have an (effective) way to react not only in the case there are *zero* new $\leqslant 2$-clauses but for an *arbitrary* number of new $\leqslant 2$-clauses (see Section 4.3).[24]

Lemma 4.2. *The basic observations for* "*Autarkness*" (*with trivial proofs*)
1. *For $F, F' \in \mathscr{CLS}$ with $F' \subseteq F$ the implication $F \in \mathscr{SAT} \Rightarrow F' \in \mathscr{SAT}$ holds.*
2. *For $F \in \mathscr{CLS}$ and $\varphi \in \mathscr{PASS}$ the implication $\varphi * F \in \mathscr{SAT} \Rightarrow F \in \mathscr{SAT}$ holds.*

4.2. Br-Autarkness

For a convenient handling of the different forms of autarkness the following notions for new clauses are useful.

Definition 4.2. For $r: \mathscr{CLS} \to \mathscr{CLS}$, $\varphi \in \mathscr{PASS}$ and $F \in \mathscr{CLS}$ we denote by $\boldsymbol{N_r(\varphi, F)}$ the set of new clauses created by applying first φ and then r, and by $\boldsymbol{N'_r(\varphi, F)}$ those new clauses which are not blocked:

$$\boldsymbol{N_r(\varphi, F)} := r(\varphi * F) \setminus F,$$

$$\boldsymbol{N'_r}(\varphi, \boldsymbol{F}) := N_r(\varphi, F) \setminus B(r(\varphi * F)).$$

In case of omitted r or φ we use the "neutral elements" instead, that is $\mathrm{id}_{\mathscr{CLS}}$ for r and $\emptyset$ (the empty assignment) for φ.

Since the following easy observations are often used, we explicitly state them:

Lemma 4.3. *Consider $\varphi \in \mathscr{PASS}$ and $F \in \mathscr{CLS}$.*
1. *φ autark for $F \Leftrightarrow N(\varphi, F) = \emptyset$.*
2. *In case of $F \in p\text{-}\mathscr{CLS}$ we have for any r fulfilling $\forall G \in p\text{-}\mathscr{CLS}: N_r(G) \in (p-1)\text{-}\mathscr{CLS}$*

$$N_r(\varphi, F) = (r(\varphi * F))^{[0, p-1]} \setminus F.$$

Lemma 4.4 (Br-Autarkness). *For $r: \mathscr{CLS} \to \mathscr{CLS}$ fulfilling $\forall F \in \mathscr{CLS}: r(F) \overset{\text{sat}}{\equiv} F$, and for $\varphi \in \mathscr{PASS}$, $F \in \mathscr{CLS}$ we have*

$$N'_r(\varphi, F) = \emptyset \;\Rightarrow\; \varphi * F \overset{\text{sat}}{\equiv} F \;(\overset{\text{sat}}{\equiv} r(\varphi * F)).$$

(*If after application of a partial assignment and reductions all new clauses are blocked, then the reduced clause-set is sat-equivalent to the original one.*)

Proof. By Lemmas 4.2 and 3.1. □

[24] Naturally, we then have to branch and the branching is the worse the bigger the number of new 2-clauses is.

4.3. Generalized Autarkness

Consider for $F \in \mathcal{CLS}$ and $\varphi \in \mathcal{PASS}$ the set $N(\varphi, F)$ of new clauses. The basic observation is just that for such $\psi \in \mathcal{PASS}$ for which all new clauses vanish, i.e., $\psi(N(\varphi, F)) = 1$ holds, φ becomes autark for $\psi * F$!

Thus we can build a new "Autarkness-branching" by testing all variables in $N(\varphi, F)$ via a complete branching $(\psi_1, \ldots, \psi_p)$ and adjoining φ to those ψ_i with $\psi_i(N(\varphi, F)) = 1$. Basic Autarkness is the special case with $N(\varphi, F) = \emptyset$, where the "Autarkness-branching" degenerates to a single branch: $p = 1$, $\psi_1 = \emptyset$.

Lemma 4.5. *For $F \in \mathcal{CLS}$ and compatible $\varphi, \psi \in \mathcal{PASS}$*[25] *the following holds*:
1. $N(\varphi, \psi * F) \subseteq \psi * N(\varphi, F)$.
2. $\psi(N(\varphi, F)) = 1 \Rightarrow \varphi$ *is autark for* $\psi * F$.

Proof. Part 1 is an easy exercise (note that $\varphi * (\psi * F) = (\phi \cup \psi) * F = \psi * (\varphi * F)$), and part 2 follows from part 1. $\square$

For later use we state explicitly in the next lemma the conclusion from part 2 and Lemma 4.1.

Lemma 4.6. *Assume a clause-set $F \in \mathcal{CLS}$ and partial assignments $\varphi, \psi \in \mathcal{PASS}$ are given fulfilling* $\mathrm{Var}(\psi) \subseteq \mathrm{Var}(N)$ *and* $\psi(N(\varphi, F)) = 1$. *Then the assignment ψ is extended sat-equivalently by φ: $\psi * F \overset{\text{sat}}{\equiv} (\psi \cup \varphi) * F$.* $\square$

5. Polynomial reductions

In this section we introduce the polynomial reductions used in our 3-SAT-algorithm $\mathcal{N}_3$.

As usual a *reduction* is a subset of the satisfiability equivalence $\overset{\text{sat}}{\equiv}$ (called "correctness" of the reduction). All reductions we use can be seen as special combinations of the following building blocks:
- elimination of *subsumed clauses*;
- addition/elimination of *resolvents*;
- addition/elimination of *blocked clauses*;
- application of *br-autarkness* for special partial assignments.

We combine our (nine) reductions by the "reduction operator" $\boldsymbol{r} : 3\text{-}\mathcal{CLS} \to 3\text{-}\mathcal{CLS}$ (see Definition 5.3), whose relevant properties are stated in Lemma 5.1.

5.1. The elementary reductions, combined by r_0

We start with the (eight) elementary reductions for $F \in 3\text{-}\mathcal{CLS}$:

[25] That is: $\varphi|(\mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi)) = \psi|(\mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi))$.

1. $F \to \langle l \to 1\rangle * F$ (*Elimination of a variable*)
   for $l \in \mathrm{Lit}(F)$ and one of
   (a) "1-clause-elimination": $\{l\} \in F^{[1]}$;
   (b) "resolution with subsequent one-clause-elimination": $\{l,x\}, \{l,\overline{x}\} \in F^{[2]}$;
   (c) "pure literals": $\overline{l} \notin \mathrm{Lit}(F)$.
2. $F \to (F \backslash \{C\}) \cup \{C \backslash \{l\}\}$ (*Elimination of a literal occurrence*)
   for $l \in C \in F^{[3]}$ and $\exists\, C' \in F\ [\,\overline{l} \in C' \wedge C' \backslash \{\overline{l}\} \subseteq C \backslash \{l\}\,]$
   ("Resolution with subsequent subsumption of one parent clause").
3. $F \to F \cup \{\{a,b\}\}$ (*Addition of a clause*)
   for $\{a,b\} \in B(F)^{[2]} \backslash F$ ("Addition of blocked 2-clauses").
4. $F \to F \backslash \{C\}$ (*Elimination of a clause*)
   for $C \in F^{[3]}$ and one of
   (a) "Subsumption": $\exists\, C' \in F\ [\, C' \subset C\,]$;
   (b) "Resolution with subsequent subsumption of one not-parent clause":
   $\exists\, C', C'' \in F \backslash \{C\}, l \in C'\ [\, C' \cap \overline{C''} = \{l\} \wedge (C' \backslash \{l\}) \cup (C'' \backslash \{\overline{l}\}) \subseteq C\,]$;
   (c) "Blocked clauses": $C \in B(F)$.

Correctness of Reductions 1(a), 1(c) and 4(a) is obvious, for Reductions 3 and 4(c) use Lemma 3.1, and correctness of Reductions 1(b), 2 and 4(b) follows from the correctness of Resolution, that means that $\{C, C'\} \models (C \backslash \{l\}) \cup (C' \backslash \{\overline{l}\})$ holds for clauses $C, C' \in \mathscr{CL}$ with $C \cap \overline{C'} = \{l\}$ (i.e., C and C' have (exactly) one complementary literal l in common).

Note that only by reductions of group 1 a 2-clause can vanish (and these reductions eliminate at least one variable), and none of the reductions from 1 – 4 create a new 3-clause or a new variable.

Definition 5.1. Let $r_0 : 3\text{-}\mathscr{CLS} \to 3\text{-}\mathscr{CLS}$ be polynomially computable s.t. $r_0(F)$ emerges from F by successive applications of the eight reductions above (chosen accordingly to their above ordering) and none of these reductions is applicable to $r_0(F)$.

Obviously there are such r_0's (Reduction 3 does not cause problems since for n variables there are at most $4\binom{n}{2}$ 2-clauses at all).

An useful notation for partial assignments $\varphi \in \mathscr{PASS}$ and $F \in \mathscr{CLS}$ is the "*1-clause-closure*" $[\boldsymbol{\varphi}]_{\boldsymbol{F}}$ of φ w.r.t. F, computed (polynomially) by successively extending φ by possible 1-clause-eliminations:

Definition 5.2. For $F \in \mathscr{CLS}$ and $\varphi \in \mathscr{PASS}$ we define $[\varphi]_{\boldsymbol{F}}$ as the result of the following procedure:

```
$[\varphi]_F := \varphi$;
WHILE ($\bot \notin [\varphi]_F * F \ \wedge\ ([\varphi]_F * F)^{[1]} \neq \emptyset$) DO
  choose $\{l\} \in ([\varphi]_F * F)^{[1]}$;
  $[\varphi]_F := [\varphi]_F \cup \langle l \to 1\rangle$
END WHILE.
```

In case of $\bot \notin [\varphi]_F * F$ the assignment $[\varphi]_F$ is well-defined and does not depend on the order of choices in Definition. 5.2. We have the following properties:

$$[\varphi]_F \in \mathscr{PASS}; \quad \varphi \subseteq [\varphi]_F; \quad [\varphi]_F * F \overset{\text{sat}}{\equiv} \varphi * F; \quad ([\varphi]_F * F)^{[1]} = \emptyset \quad \text{or } \bot \in [\varphi]_F * F.$$

5.2. "One-step look-ahead" for r_0, and the reduction operator r

The last reduction is as follows:

5. $F \to r_0(\langle l \to 1\rangle * (F \cup E))$

for a literal $l \in \mathrm{Lit}(F)$ and a clause-set E with either $E = \emptyset$ or $E = B_{\bar{l}}^{[3]}(F)$ fulfilling

$N'_{r_0}(\langle l \to 1\rangle, F \cup E) = \emptyset.$

By Lemma 3.2 we have $F \overset{\text{sat}}{\equiv} F \cup E$, and thus correctness of Reduction 5 follows by Lemma 4.4 about br-autarkness.

Definition 5.3. Let $r : 3\text{-}\mathscr{CLS} \to 3\text{-}\mathscr{CLS}$ be polynomially computable s.t. $r(F)$ emerges from F by successive applications of r_0 and Reduction 5 (r_0 with priority), and none of the nine reductions from 1 – 5 is applicable to $r(F)$.

The set of reduced 3-clause-sets we denote by

$\mathbf{3}\text{-}\mathscr{CLS}_{\mathbf{r}} := r(3\text{-}\mathscr{CLS}) \setminus \{\{\bot\}\}.$

(It is convenient for further use to exclude the empty clause here.)

What is the purpose of reduction 5?. In our proof of the upper bound $1.5044..^n$ for $\mathscr{N}_3$ we have to consider the following situation:

Suppose that in $\mathscr{N}_3$ for input $F_0 \in 3\text{-}\mathscr{CLS}_r$, for any partial assignment $\boldsymbol{\varphi}$ part of a branching envisaged by $\mathscr{N}_3$, and any modified version $F \overset{\text{sat}}{\equiv} F_0$ of F_0 used in $\mathscr{N}_3$, the case $N'_r(\varphi, F) = \emptyset$ occurs.

By br-Autarkness now F_0 can be reduced to $r(\varphi * F)$.

However, in $r(\varphi * F)$ possibly there is now no not-blocked 2-clause at all, which "normally" would be used for branching at $r(\varphi * F)$, and so we demand $n(r(\varphi * F)) \leqslant n(F) - 2$ as compensation.

This loss of variables yet is established by Reduction 5, since Reduction 5 covers all cases where only one variable is eliminated (see part 5 of Lemma 5.1). The extension E in Reduction 5 thereby captures the (relevant) modification of F_0 to F.

5.3. Properties of r

What we need for r is collected in the following lemma.

Lemma 5.1. *For $F \in 3\text{-}\mathscr{CLS}$ the following holds*:

1. $r(F) \overset{\text{sat}}{\equiv} F$, $r(r(F)) = r(F)$, *$r(F)$ is polynomially computable in* $\ell(F) = \sum_{C \in F} |C|$.
2. *$r(F)$ is obtained from F by a series of the following operations*:
   – *application of a partial assignment*;

– *deletion of a literal-occurrence in a* 3*-clause*;
– *addition of* 2*-clause* (*without new variables*);
– *elimination of a* 3*-clause.*

3. *Basic relations between* $r(F)$ *and* F:
   (a) $r(F) \in 3\text{-}\mathscr{CLS}$;
   (b) $\mathrm{Var}(r(F)) \subseteq \mathrm{Var}(F)$ (*no new variables are introduced*);
   (c) $(r(F))^{[3]} \subseteq F^{[3]}$ (*no new* 3*-clauses are created*);
   (d) $F^{[0,2]} \not\subseteq (r(F))^{[0,2]} \Rightarrow \mathrm{Var}(r(F)) \subset \mathrm{Var}(F)$ (*elimination of a* $\leqslant 2$*-clause is always accompanied by elimination of a variable*);
   (e) $\bot \in F \Rightarrow r(F) = \{\bot\}$.
4. *For* $F \in 3\text{-}\mathscr{CLS}_r$ *we have*:
   (a) $F \neq \top \Rightarrow n(F) \geqslant 3$ ($F \neq \top$ *contains at least three variables*);
   (b) $F^{[0,1]} = \emptyset$ (F *consists only of* 2*- and* 3*-clauses*);
   (c) $(\{l,a\}, \{l,b\} \in F^{[2]},\ \ \{l,a\} \neq \{l,b\}) \ \Rightarrow\ \mathrm{Var}(a) \neq \mathrm{Var}(b)$ (*if two* 2*-clauses have a common literal, then the other literals in these clauses have distinct variables*);
   (d) $\overline{\mathrm{Lit}(F)} = \mathrm{Lit}(F)$ (*if a literal is in* F, *then also its complement*);
   (e) $\{l,a,b\} \in F^{[3]} \Rightarrow \{\bar{l}\}, \{\bar{l},a\}, \{\bar{l},b\}, \{\bar{l},a,b\} \notin F$ (*after elimination of complementary literal-pairs there are no subsumed clauses*);
   (f) (i) $\forall\, a,b \in \mathrm{Lit}(F),\ a \neq b : \forall\, C \in F\ [\, a \in C \Rightarrow b \in C\,] \Rightarrow \{\bar{a},\bar{b}\} \in F^{[2]}$ (*if a literal* a *is always accompanied by the literal* b, *then the* 2*-clause* $\{\bar{a},\bar{b}\}$ *is in* F);
   (ii) $\#_l(F) = 1 \wedge l \in C \in F \Rightarrow \#^2_{\bar{l}}(F) \geqslant |C| - 1$ (*if a literal occurs only once* (*in a* 2*-/*3*-clause*), *then its complement occurs in some* 2*-clauses* (*at least once/twice*));
   (g) $\{x,y,z\} \in F^{[3]} \Rightarrow \{x,y\}, \{x,z\}, \{y,z\} \notin F$ (*there are no subsumed clauses*);
   (h) $(l \in C_1 \in F,\ \bar{l} \in C_2 \in F) \ \Rightarrow\ (C_1 \backslash \{l\}) \cup (C_2 \backslash \{\bar{l}\}) \notin F^{[3]}$ (*no* 3*-clause is a resolvent of clauses in* F);
   (i) $\forall\, C \in F^{[3]}\, \forall\, l \in C\, \exists\, C' \in F : C \cap \overline{C'} = \{l\}$ (*for every* 3*-clause and every literal* l *in it there is a clause in* F *containing* $\bar{l}$, *but no other complementary literals*).
5. *For a* (*reduced*) 3*-clause-set* $F \in 3\text{-}\mathscr{CLS}_r$, *a literal* l, *a partial assignment* $\varphi \in \mathscr{PAS}$ *with* $\emptyset \neq \mathrm{Var}(\varphi) \subseteq \mathrm{Var}(F)$, *and for* E *with either* $E = \emptyset$ *or* $E = B^{[3]}_{\bar{l}}(F)$, *where* $\varphi(l) = 1$ *is assumed, we have*
   $N'_r(\varphi, F \cup E) = \emptyset \Rightarrow n(r(\varphi * (F \cup E))) \leqslant n(F) - 2$ (*if in* $r(\varphi * (F \cup E))$ *all new clauses are blocked, then* $r(\varphi * (F \cup E))$ *must have at least two variables less than* F).
6. *If there are clauses* $\{\bar{x},b\} \in F^{[2]}$ *and* $\{\bar{x},\bar{b},\bar{c}\} \in F^{[3]}$, *then* $\{\bar{x},\bar{c}\} \in r(F)$ *or* $\{\bar{b},\bar{c}\} \in r(F)$ *or* $\mathrm{Var}(r(F)) \subset \mathrm{Var}(F)$ *must hold.*
7. *For* $\varphi \in \mathscr{PAS}$ *we have*: $\bot \in r(\varphi * F) \cap r([\varphi]_F * F)$ *or* $r(\varphi * F) = r([\varphi]_F * F)$.

Proof. Part 1 we have already discussed. Part 2 follows immediately from the definition of r. Part 3 follows from part 2. For 4(a) note that F with $0 < n(F) \leqslant 2$ could be reduced using the reductions of group 1 and 5. Properties 4(b) – 4(i) follow from the definitions of Reductions 1–4 in the same order ("$\bot \notin F$" in part 4(b) follows by

Subsumption and the definition of -3-$\mathscr{CLS}_r$, and part 4(f)ii is an immediate consequence of 4(f)i).

Part 5: Assume $n(r(\varphi * (F \cup E))) = n(F) - 1$. Thus $n(\varphi) = 1$ and Reduction 5 had not been applied: $r(\varphi * (F \cup E)) = r_0(\varphi * (F \cup E))$ – but now in case $N'_r(\varphi, F \cup E) = \emptyset$ we could apply Reduction 5 to F contradicting $F \in 3\text{-}\mathscr{CLS}_r$.

Part 6: Assume $\operatorname{Var}(r(F)) = \operatorname{Var}(F)$. Thus no reduction of group 1 has been applied to F and hence, according to the ordering of reductions, Reduction 2 must have been applied to $\{\overline{x}, \overline{b}, \overline{c}\}$ (because of $\{\overline{x}, b\} \in F$). If the literal $\overline{c}$ of $\{\overline{x}, \overline{b}, \overline{c}\}$ would have been eliminated then thereafter Reduction 1(b) could have been applied yielding a contradiction. So one of the literals $\overline{b}, \overline{c}$ must have been eliminated from $\{\overline{x}, \overline{b}, \overline{c}\}$.

Part 7: 1-clause-elimination is the first reduction in the ordering and is confluent except of the different ways to produce $\bot$. □

6. The algorithm

In this section we present the 3-SAT decision algorithm $\mathscr{N}_3$, explain its main features, and prove its correctness.

6.1. The overall structure of $\mathscr{N}_3$

The recursive procedure

$$\mathscr{N}_3 : 3\text{-}\mathscr{CLS}_r \to \{0, 1\}$$

is specified by $\mathscr{N}_3(F_0) = 1 \Leftrightarrow F_0 \in \mathscr{SAT}$ for inputs $F_0 \in 3\text{-}\mathscr{CLS}_r$.

We use as data structure finite families $\hat{\varphi}$ of partial assignments with integers as indices:

Definition 6.1.

$$\boldsymbol{\mathscr{FPASS}} := \bigcup_{\substack{I \subseteq \mathbb{N}_0 \\ I \text{finite}}} \mathscr{PASS}^I$$

For $\hat{\varphi} \in \mathscr{FPASS}$ we denote by $\boldsymbol{I(\hat{\varphi})} := \operatorname{dom}(\hat{\varphi})$ the index set of $\hat{\varphi}$, and for $i \in I(\hat{\varphi})$ we use $\hat{\varphi}_i := \hat{\varphi}(i)$.

The very topmost perspective on $\mathscr{N}_3$ is that it consists of three (structured) instructions:

1. If $F = \top$, then output "satisfiable".
2. Otherwise compute a "branching" $(\hat{\varphi}, F)$ for input F_0, where F is a variant of F_0, and $\hat{\varphi}$ the list of "test assignments".
3. Then branching by $\hat{\varphi}$ (on F) is performed.

PROCEDURE $\mathscr{N}_3(F_0 \in 3\text{-}\mathscr{CLS}_r)$: $\{0, 1\}$;

```
  VARIABLES
    $F \in 3\text{-}\mathscr{CLS}$;
    $\hat{\varphi} \in \mathscr{FPASS}$;
  BEGIN
 (1) IF $F = \top$ THEN
       RETURN $\mathscr{N}_3(F_0) := 1$;
 (2) $(\hat{\varphi}, F) :=$ branching$(F_0)$;
 (3) IF $I(\hat{\varphi}) = \emptyset$ THEN
(3a)   RETURN $\mathscr{N}_3(F_0) := 0$
     ELSE
(3b)   RETURN $\mathscr{N}_3(F_0) := \max_{i \in I(\hat{\varphi})} \mathscr{N}_3(r(\hat{\varphi}_i * F))$
     END IF
  END $\mathscr{N}_3$;
```

$\mathscr{N}_3$ uses the function

$$\textbf{branching}: 3\text{-}\mathscr{CLS}_r \setminus \{\top\} \to \mathscr{FPASS} \times 3\text{-}\mathscr{CLS}$$

which must fulfill the following correctness conditions (C1)–(C3) for any $F_0 \in 3\text{-}\mathscr{CLS}_r$ (using branching$(F_0) =: (\hat{\varphi}, F)$):
(C1) $\forall\, i \in I(\hat{\varphi}) : \emptyset \neq \operatorname{Var}(\hat{\varphi}_i) \subseteq \operatorname{Var}(F) = \operatorname{Var}(F_0)$
(C2) $F_0 \in \mathscr{SAT} \Leftrightarrow \exists\, i \in I(\hat{\varphi}) : \hat{\varphi}_i * F \in \mathscr{SAT}$
(C3) $\forall i \in I(\hat{\varphi}) : \bot \notin r(\hat{\varphi}_i * F)$.
(C1) ensures the termination of $\mathscr{N}_3$, (C2) the correctness of the result, and (C3) the input condition "$F_0 \in 3\text{-}\mathscr{CLS}_r$" for 3(b).

The function "branching" considers three cases for F_0, depending on $\boldsymbol{\rho(F_0)}$, the maximal number of occurrences of a variable in the 2-clauses of F_0:

Definition 6.2. For $F \in 3\text{-}\mathscr{CLS}$: $\boldsymbol{\rho(F)} := \max_{v \in \mathscr{VA}} (\#^2_v(F) + \#^2_{\overline{v}}(F))$.

```
PROCEDURE branching($F_0 \in 3\text{-}\mathscr{CLS}_r \setminus \{\top\}$) : $\mathscr{FPASS} \times 3\text{-}\mathscr{CLS}$;
BEGIN
  IF $\rho(F_0) \geqslant 2$ THEN
    RETURN branching$(F_0) :=$ (branching$A(F_0)$, $F_0$)
  ELSE IF $F_0^{[2]} \setminus B(F_0) \neq \emptyset$ THEN
    RETURN branching$(F_0) :=$ branching$B(F_0)$
  ELSE
    RETURN branching$(F_0) :=$ (branching$C(F_0)$, $F_0$)
  END IF
END branching;
```

The three cases (A)–(C) in words:
(A): Some variable occurs at least twice in the 2-clauses of F_0.
(B): The 2-clauses of F_0 are variable-disjoint, and at least one of them is not blocked.

(C): The 2-clauses of F_0 are variable-disjoint, and every 2-clause is blocked (including the case that there is no 2-clause at all).

Before presenting in Sections 6.3–6.5 the functions

$$\textbf{branching}A, \textbf{branching}C : 3\text{-}\mathscr{CLS}_r \setminus \{\top\} \to \mathscr{FPASS}$$

$$\textbf{branching}B : 3\text{-}\mathscr{CLS}_r \setminus \{\top\} \to \mathscr{FPASS} \times 3\text{-}\mathscr{CLS},$$

we give in the subsequent subsection a frequently used auxiliary procedure for evaluating families of partial assignments.

6.2. *Evaluating families of partial assignments with respect to cases of immediate decision or autarkness*

The function

$$\textbf{eval} : \mathscr{FPASS} \times 3\text{-}\mathscr{CLS} \to \mathscr{FPASS}$$

performs the following task on input $(\hat{\varphi}, F)$:

> Each $\hat{\varphi}_i$ is replaced by $[\hat{\varphi}_i]_F$. Then branches creating $\bot$ are eliminated, and if there is a branch where basic autarkness or br-autarkness occurs, only such a branch is left.

Definition 6.3. For $\hat{\varphi} \in \mathscr{FPASS}$ and $F \in 3\text{-}\mathscr{CLS}$ we define:

$$\boldsymbol{I'(\hat{\varphi}, F)} := \{i \in I(\hat{\varphi}) \colon \bot \notin r([\hat{\varphi}_i]_F * F)\}$$

$$\boldsymbol{A(\hat{\varphi}, F)} := \{i \in I'(\hat{\varphi}, F) \colon N([\hat{\varphi}_i]_F, F) = \emptyset \vee N'_r([\hat{\varphi}_i]_F, F) = \emptyset\}$$

$$\textbf{eval}(\boldsymbol{\hat{\varphi}, F}) := \begin{cases} ([\hat{\varphi}_i]_F)_{i \in I'(\hat{\varphi}, F)} & \text{if } A(\hat{\varphi}, F) = \emptyset \\ ([\hat{\varphi}_a]_F)_{i \in \{a\}} & \text{for some } a \in A(\hat{\varphi}, F) \text{ else.} \end{cases}$$

The properties of "eval" we need in the sequel are listed in the following lemma.

Lemma 6.1. *For $F \in \mathscr{CLS}$ and $\hat{\varphi} \in \mathscr{FPASS}$ we have* (*using* $\hat{\varphi}' := \text{eval}(\hat{\varphi}, F)$):

1. "*$\hat{\varphi}'$ is as good as $\hat{\varphi}$ with respect to satisfiability*":

$$\exists i \in I(\hat{\varphi})[\hat{\varphi}_i * F \in \mathscr{SAT}] \Leftrightarrow \exists i \in I(\hat{\varphi}')[\hat{\varphi}'_i * F \in \mathscr{SAT}].$$

2. *$I(\hat{\varphi}') \subseteq I(\hat{\varphi})$, and for each $i \in I(\hat{\varphi}')$ we have $\hat{\varphi}'_i = [\hat{\varphi}_i]_F$ and $r(\hat{\varphi}'_i * F) \in 3\text{-}\mathscr{CLS}_r$.*
3. *In case of $|I(\hat{\varphi}')| \geqslant 2$ neither basic autarkness nor br-autarkness can be applied to any branch*:

$$\forall i \in I(\hat{\varphi}') \colon N(\hat{\varphi}'_i, F) \neq \emptyset \wedge N'_r(\hat{\varphi}'_i, F) \neq \emptyset.$$

4. *If there is $i \in I(\hat{\varphi}')$ with $N'_r(\hat{\varphi}', F) = \emptyset$, then we have $I(\hat{\varphi}') = \{i\}$. Assume that additionally the following assumption is fulfilled*:

There is $F_0 \in 3\text{-}\mathscr{CLS}_r$, a literal $l \in \mathrm{Lit}(F_0)$ with $\hat{\varphi}'_i(l) = 1$,

and E with either $E = \emptyset$ or $E = B_{\bar{l}}(F_0)$, such that

$$F = F_0 \cup E.$$

Then we know $n(r(\hat{\varphi}'_i * F)) \leqslant n(F) - 2$.

Proof. Parts 2 and 3 follow by Definition 6.3. Part 1 is an immediate consequence of Lemma 4.4, and for part 4 use Lemma 5.1, part 5. □

6.3. Case A: Multiple occurrences of a variable in the 2-clauses

The main features of the handling of case (A) are:
- Branching on a variable occurring maximally often (and in the most symmetrical way) in the 2-clauses of F_0.
- Applying Generalized Autarkness to branches creating only one new clauses (but only in cases where we are sure to obtain a "good" branching).

```
PROCEDURE branching$A$($F_0 \in 3\text{-}\mathscr{CLS}_r \setminus \{\top\}$) : $\mathscr{FPASS}$;
VARIABLES
    $l \in \mathscr{LIT}$;
    $\hat{\varphi} \in \mathscr{FPASS}$;
    $\varphi \in \mathscr{PASS}$;
BEGIN
 (1)  $l :=$ branchLitA($F_0$);
 (2)  $I(\hat{\varphi}) := \{0,1\}$; $\hat{\varphi}_0 := \langle l \to 0 \rangle$; $\hat{\varphi}_1 := \langle l \to 1 \rangle$;
      $\hat{\varphi} :=$ eval($\hat{\varphi}, F_0$);
      IF $|I(\hat{\varphi})| \leqslant 1$ THEN
         RETURN branching$A$($F_0$) := $\hat{\varphi}$;
 (3)  IF $\exists i \in \{0,1\}$: $|N(\hat{\varphi}_i, F_0)| = 1 \wedge n(\hat{\varphi}_i) \geqslant 2 \wedge (n(\hat{\varphi}_0) - 1) \cdot (n(\hat{\varphi}_1) - 1) \leqslant 1$ THEN
(3a)     choose such $i$;
         RETURN branching$A$($F_0$) := GenAut($\hat{\varphi}_i, F_0$)
      ELSE
(3b)     RETURN branching$A$($F_0$) := $\hat{\varphi}$
  END branching$A$;
```

The algorithm in words: In (1) the branching literal l is selected, using

$$\text{branchLitA} : 3\text{-}\mathscr{CLS} \setminus \{\top\} \to \mathscr{LIT},$$

and the corresponding binary branching $\hat{\varphi}$ is evaluated in (2).

Then in case there is $\hat{\varphi}_i$ creating only one new clause, with $n(\hat{\varphi}_i)$ "big enough" and the whole branching $\hat{\varphi}$ not already "a very good one", in (3a) the "Generalized Autarkness Branching", based on $\hat{\varphi}_i$ and given by

$$\textbf{GenAut} : \mathscr{PASS} \times 3\text{-}\mathscr{CLS} \to \mathscr{FPASS},$$

is computed and used as output of "branching$A$". (See Section 8.3 for a motivation of the condition in (3).)

Otherwise $\hat{\varphi}$ itself is the result in (3b).

Lemma 6.2. *Suppose that the function* "GenAut" *terminates, and its result fulfills correctness conditions* (C1)–(C3) *from Section* 6.1 *with* $F = F_0$ *and* $\hat{\varphi} = \mathrm{GenAut}(\varphi, F_0)$ (*as it will be shown in Lemma* 6.3). *Then* (C1)–(C3) *are valid also for the output of* "branching$A$" (*with* $F = F_0$).

6.3.1. The branching rule

In case (A) we choose as branching variable a variable occurring maximally often in the 2-clauses of F_0, and in case there is more than one of this kind we balance the numbers $\#_v^2(F_0)$ and $\#_{\bar{v}}^2(F_0)$ of occurrences in both signs.

For standardization we actually use a branching *literal* requiring additionally that l (itself) occurs at least once in the 2-clauses of F_0.

```
PROCEDURE branchLitA($F_0 \in 3\text{-}\mathscr{CLS}_r \setminus \{\top\}$) : $\mathscr{LIT}$;
BEGIN
  $L := \{l \in \mathrm{Lit}(F) : (\#_l^2 + \#_{\bar{l}}^2)(F_0) = \rho(F_0) \wedge \#_l^2(F_0) \geqslant 1\}$;
  choose $l \in L$ with maximal $\min(\#_l^2(F_0), \#_{\bar{l}}^2(F_0))$;
  RETURN branchLitA($F_0$) := $l$
END branchLitA;
```

6.3.2. The "Generalized Autarkness branching"

The function "GenAut" is defined on inputs $(\varphi, F_0) \in \mathscr{PASS} \times 3\text{-}\mathscr{CLS}$ fulfilling

$$(\mathrm{GA})_1: \quad \mathrm{N}(\varphi, \mathrm{F}_0) = \mathrm{N}(\varphi, \mathrm{F}_0)^{[2]} \wedge |\mathrm{N}(\varphi, \mathrm{F}_0)| = 1.$$

According to the idea of "Generalized Autarkness"

> "test the variables in $N(\varphi, F_0)$ and extend those branches by φ which satisfy $N(\varphi, F_0)$"

the "Generalized Autarkness branching" is computed. The construction is iterated if in this way $n(\varphi)$ increases.

```
PROCEDURE GenAut($\varphi \in \mathscr{PASS}, F_0 \in 3\text{-}\mathscr{CLS}$) : $\mathscr{FPASS}$;
VARIABLES
    $\hat{\varphi} \in \mathscr{FPASS}$;
    $a, b \in \mathscr{LIT}$;
BEGIN
 (1) LET $N(\varphi, F_0) = \{\{a, b\}\}$;
 (2) $I(\hat{\varphi}) := \{0, 1, 2\}$;
    $\hat{\varphi}_0 := \langle a \to 0, b \to 0 \rangle$; $\hat{\varphi}_1 := \langle a \to 0, b \to 1 \rangle \cup \varphi$; $\hat{\varphi}_2 := \langle a \to 1 \rangle \cup \varphi$;
    $\hat{\varphi} := \mathrm{eval}(\hat{\varphi}, F_0)$;
```

```
         IF $|I(\hat{\varphi})| \leqslant 2$ THEN
            RETURN GenAut($\varphi, F_0$) := $\hat{\varphi}$;
   (3)   IF $\forall\, i \in I(\hat{\varphi}) : |N(\hat{\varphi}_i, F_0)| \geqslant 2 \vee n(\hat{\varphi}_i) \leqslant n(\varphi)$ THEN
  (3a)      RETURN GenAut($\varphi, F_0$) := $\hat{\varphi}$
         ELSE
  (3b)      choose $i \in I(\hat{\varphi})$ with $|N(\hat{\varphi}_i, F_0)| = 1$ and $n(\hat{\varphi}_i) > n(\varphi)$;
            RETURN GenAut($\varphi, F_0$) := GenAut($\hat{\varphi}_i, F_0$)
     END GenAut;
```

In (1) the new clause is named "$\{a, b\}$", Then in (2) the "Generalized Autarkness branching" is generated and evaluated.

If all $\hat{\varphi}_i$ now either create at least two new clauses or do not eliminate more variables than φ, then the result of "GenAut" is $\hat{\varphi}$ (in (3a)).

Otherwise in (3b) the same construction is repeated with $\hat{\varphi}_i$ instead of φ.

Lemma 6.3. *The procedure "GenAut" terminates for inputs $\varphi \in \mathscr{PASS}$ and $F_0 \in$ 3-$\mathscr{CLS}$ fulfilling $(GA)_1$, and the correctness conditions* (C1)–(C3) *are fulfilled for $\hat{\varphi} = \mathrm{GenAut}(\varphi, F_0)$ and $F = F_0$.*

Proof. The termination is guaranteed because of the increasing $n(\varphi)$-values. For the correctness conditions use Lemmas 6.1 and 4.6 with $\psi = \langle a \to 0, b \to 1\rangle$ resp. $\psi = \langle a \to 1\rangle$. □

6.4. Case B: Variable-disjoint 2-clauses, and a not-blocked 2-clause exists

In case (B) the following ideas are used:
1. The branching literal is chosen from a 2-clause which is *not blocked.*
2. In case not "enough" new clauses are created, *blocked clauses* are added for compensating the missing new clauses.

```
PROCEDURE branching$B$($F_0 \in$ 3-$\mathscr{CLS}_r \setminus \{\top\}$) : $\mathscr{FPASS} \times$ 3-$\mathscr{CLS}$;
VARIABLES
    $l \in \mathscr{LIT}$;
    $\hat{\varphi} \in \mathscr{FPASS}$;
    $F \in$ 3-$\mathscr{CLS}$;
BEGIN
(1) $l$ := branchLitB($F_0$);
(2) $I(\hat{\varphi}) := \{0, 1\}$; $\hat{\varphi}_0 := \langle l \to 0\rangle$; $\hat{\varphi}_1 := \langle l \to 1\rangle$;
    $\hat{\varphi}$ := eval($\hat{\varphi}, F_0$);
    IF $|I(\hat{\varphi})| \leqslant 1$ THEN
       RETURN branching$B$($F_0$) := ($\hat{\varphi}, F_0$);
(3) IF $|N(\hat{\varphi}_0, F_0)| \geqslant 3 \wedge |N(\hat{\varphi}_1, F_0)| \geqslant 3$ THEN
       RETURN branching$B$($F_0$) := ($\hat{\varphi}, F_0$);
(4) IF $|N(\hat{\varphi}_0, F_0)| = 2$ THEN
```

```
(4a)    $F := F_0 \cup B_{\overline{l}}^{[3]}(F_0)$
        ELSE
(4b)    $F := F_0 \cup B_l^{[3]}(F_0)$;
 (5) RETURN branching$B(F_0) := (\mathrm{eval}(\hat{\varphi}, F), F)$
 END branching$B$;
```

The algorithm in words. In (1) the branching literal is selected, and the corresponding binary branching $\hat{\varphi}$ is evaluated in (2).

If both branches create at least three new clauses (before applying r) then in this "normal case" the result is $\hat{\varphi}$ (in (3)), without changing F_0.

Otherwise blocked clauses are added in (4a) resp. (4b):

In (4a) we choose "blocked for $\overline{l}$" since there are only few l-occurrences, while in (4b) there are only few $\overline{l}$-occurrences in F_0.

Lemma 6.4. *The result of "*branching$B$*" fulfills correctness conditions* (C1)–(C3).

Proof. Use Lemmas 6.1 and 3.2. □

6.4.1. The branching rule

```
PROCEDURE branchLitB($F_0 \in 3\text{-}\mathscr{CLS}_r \setminus \{\top\}$) : $\mathscr{LIT}$;
BEGIN
    choose $C \in F_0^{[2]} \setminus B(F_0)$;
    choose $l \in C$;
    RETURN branchLitB($F_0$) := $l$
END branchLitB;
```

6.5. Case C: The remaining case

In this last case, which can only occur at the root of the test tree generated by $\mathscr{N}_3$ or after application of (br)-autarkness (compare Lemma 6.1, part 4), we can simply choose any branching literal:

```
PROCEDURE branching$C$($F_0 \in 3\text{-}\mathscr{CLS}_r \setminus \{\top\}$) : $\mathscr{FPASS}$;
BEGIN
  $l :=$ branchLitC($F_0$);
  $I(\hat{\varphi}) := \{0, 1\}$; $\hat{\varphi}_0 := \langle l \to 0 \rangle$; $\hat{\varphi}_1 := \langle l \to 1 \rangle$;
  RETURN eval($\hat{\varphi}, F_0$)
END branching$C$;
PROCEDURE branchLitC($F_0 \in 3\text{-}\mathscr{CLS}_r \setminus \{\top\}$) : $\mathscr{LIT}$;
BEGIN
    choose $l \in \mathrm{Lit}(F)$;
    RETURN branchLitC($F_0$) := $l$
END branchLitC.
```

Theorem 1. *Algorithm $\mathcal{N}_3$ terminates on every input $F_0 \in 3\text{-}\mathcal{CLS}_r$ and gives the correct answer, i.e., $\mathcal{N}_3(F_0) = 0 \Leftrightarrow F \notin \mathcal{SAT}$ and $\mathcal{N}_3(F_0) = 1 \Leftrightarrow F \in \mathcal{SAT}$.*

Proof. Use Lemmas 6.2 and 6.4. □

7. A worst case upper bound

The basic concept for analyzing the running rime of algorithm $\mathcal{N}_3$ is that of a "computation tree":

Definition 7.1. For a (reduced) 3-clause-set $F_0 \in 3\text{-}\mathcal{CLS}_r$ the *computation tree* $\boldsymbol{\mathcal{T}_{\mathcal{N}_3}(F_0)}$ reproduces the recursive calls of the procedure $\mathcal{N}_3$ for input F_0, whereby we continue evolving the tree also when we found the formula satisfiable.

The *node-labelings* $F_0, F : \text{nds}(\mathcal{T}_{\mathcal{N}_3}(F_0)) \to 3\text{-}\mathcal{CLS}$ give the corresponding input-formula respectively its variant.

More precisely we define for $F_0 \in 3\text{-}\mathcal{CLS}_r$:

(i) If $F_0 = \top$ then $\boldsymbol{\mathcal{T}_{\mathcal{N}_3}(F_0)}$ is the trivial tree with one node w and $\boldsymbol{F_0(w)} := F_0 := \boldsymbol{F(w)}$.

(ii) Otherwise, using $(\hat{\varphi}, F) := \text{branching}(F_0)$, let $\boldsymbol{\mathcal{T}_{\mathcal{N}_3}(F_0)}$ be the tree with $|I(\hat{\varphi})|$-many subtrees $\mathcal{T}_{\mathcal{N}_3}(r(\hat{\varphi}_i * F))$ $(i \in I(\hat{\varphi}))$ at the root. Using w for the root: $\boldsymbol{F_0(w)} := F_0$ and $\boldsymbol{F(w)} := F$.

In fact the tree $\mathcal{T}_{\mathcal{N}_3}(F_0)$ is not uniquely defined but depends on the special choices performed in "branching". Of course all our results hold for any of these choices.

For every reasonable model of computation (e.g., Turing machines) and every (reasonable) realization $\mathcal{M}$ of the (abstract) algorithm $\mathcal{N}_3$ in this model, there exists a polynomial p s.t. for the number $t_{\mathcal{M}}(F_0)$ of steps performed by $\mathcal{M}$ for any input F_0 (suitably coded)

$$t_{\mathcal{M}}(F_0) \leqslant p(\ell(F_0)) \cdot \#\text{lvs}\,(\mathcal{T}_{\mathcal{N}_3}(F_0))$$

holds, where $\ell(F_0)$ is the number of literal occurrences in F_0 and $\boldsymbol{\#\text{lvs}(\mathcal{T}_{\mathcal{N}_3}(F_0))} := |\text{lvs}(\mathcal{T}_{\mathcal{N}_3}(F_0))|$ is the number of leaves in the computation tree $\mathcal{T}_{\mathcal{N}_3}(F_0)$.

We are interested only in the "mathematical part" $\#\text{lvs}\,(\mathcal{T}_{\mathcal{N}_3}(F_0))$ of the number of steps $t_{\mathcal{M}}(F_0)$, and abstract from the polynomial part $p(\ell(F_0))$ (which depends on the model of computation and on the realization).

Theorem 2. *For every (reduced) 3-clause-set $F_0 \in 3\text{-}\mathcal{CLS}_r$ the number of leaves in the computation tree $\mathcal{T}_{\mathcal{N}_3}(F_0)$ is bounded in the worst case by*

$$\#\text{lvs}\,(\mathcal{T}_{\mathcal{N}_3}(F_0)) < 0.55 \cdot 1.50444^{n(F_0)}$$

where $n(F_0)$ is the number of variables in F_0.

The proof of Theorem 2 is the subject of Sections 8 to 13.

8. Estimating tree sizes

8.1. Some notations and a basic estimation lemma

For us a *tree* T is any finite partial order $T=(P,\leqslant)$ with least element root (T) s.t. there is exactly one path from the root to any node.

We use ***nds*(*T*)** $:=P$ for the set of *nodes* of T, and for a node $w\in \mathrm{nds}(T)$ let $\mathrm{ds}_T(w)$ be the set of *direct successors* of w in T.[26]

Furthermore, let $\mathrm{lvs}(T)$ be the set of *leaves*, $\#\mathrm{lvs}(T)$ the number of leaves, and $\mathrm{nds}_k(T)$ the set of nodes with at least k direct successors.[27]

A *path* in T is a sequence $(w_0,\dots,w_n)$ $(n\geqslant 0)$ of nodes of T s.t. w_{i+1} is a direct successor of w_i.

An *edge labeling* d of T is a mapping

$$d:\{(w,w'):w\in \mathrm{nds}(T)\wedge w'\in \mathrm{ds}_T(w)\}\to \mathbb{R}.$$

For $w,w'\in \mathrm{nds}(T)$ with $w'>w$ we define

$$\boldsymbol{d_{w'}(w)}:=d(w,w''),\ \text{where } w'' \text{ is determined by } w''\in \mathrm{ds}_T(w)\wedge w''\leqslant w'$$

("the value of d at w in direction w'").

As *node labelings* we consider any mapping with domain $\mathrm{nds}(T)$.

Lemma 8.1. *Consider a tree T with a "transition probability" p, i.e., an edge labeling of T with values in the interval* $[0,1]$ *such that for every inner node the sum of probabilities of outgoing edges is* 1.[28]

For a leaf $w\in \mathrm{lvs}(T)$ let $p(w)$ be the resulting probability for reaching w.[29]

Then we can estimate the number of leaves by

$$\#\mathrm{lvs}(T)\leqslant \max_{w\in \mathrm{lvs}(T)} p(w)^{-1}.$$

Proof. By induction one easily proves

$$\sum_{w\in \mathrm{lvs}(T)} p(w)=1,\ \Rightarrow \#\mathrm{lvs}(T)\cdot \min_{w\in \mathrm{lvs}(T)} p(w)\leqslant 1. \qquad \square$$

For every tree T there is exactly one optimal transition probability p_T with respect to the estimation in Lemma 8.1 (i.e., where the inequality becomes an equality). p_T is characterized by the condition that the probabilities for all leaves are the same.

[26] i.e., $\mathrm{ds}_T(w):=\{w'\in \mathrm{nds}(T):w'>w\wedge \neg\exists_{w''\in \mathrm{nds}(T)}[w<w''<w']\}$.

[27] $\mathrm{nds}_k(T):=\{w\in \mathrm{nds}(T):|\mathrm{ds}_T(w)|\geqslant k\}$, $\mathrm{lvs}(T):=\mathrm{nds}(T)\setminus \mathrm{nds}_1(T)$, $\#\mathrm{lvs}(T):=|\mathrm{lvs}(T)|$.

[28] $\forall w\in \mathrm{nds}_1(T):\sum_{w'\in \mathrm{ds}_T(w)} p_{w'}(w)=1$

[29] $p(w):=\prod_{v\in \mathrm{nds}(T),\, v<w} p_w(v)$

8.2. Distance functions and our main estimation lemma

Our approach for estimating tree sizes is not based on the transition probabilities of Lemma 8.1 but on "*distance functions*" d (from which (implicitly) transition probabilities are derived).

Distance functions are edge labelings by *positive real numbers* which gauge the progress in reducing the complexity: The bigger $d(w,w')$ for a node w and one of its direct successors w' is, the bigger is (resp. "should be") the reduction of complexity achieved by the branch $w \to w'$.

Additionally we use the notion of a *measure*[30] μ with induced distance function $\Delta\mu$. A measure is a node labeling with non-negative real numbers such that the differences $\Delta\mu$ give a distance function. μ has the meaning of an estimation of "the" (decision) complexity of the formula at the given node.

As an introductory example consider the measure $\mu = n$, the number of variables (every node is labeled with the number of variables of the formula at this node). The induced distance function $\Delta\mu$ gives the number of eliminated variables (for each branch).

Definition 8.1. A *distance function* d for T is an edge-labeling of T with values in $\mathbb{R}_+^*$, where for "single edges" (w,w')[31] also $d(w,w') = 0$ is allowed.

Definition 8.2. Given a real-valued node labeling μ for a tree T, the edge labeling $\boldsymbol{\Delta\mu}$ is defined by

$$\boldsymbol{\Delta\mu(w, w')} := \mu(w) - \mu(w')$$

for $w \in \operatorname{nds}(T)$, $w' \in \operatorname{ds}_T(w)$.

If μ has its values in fact in $\mathbb{R}_+$ and $\Delta\mu$ is a distance function, then we call μ a *measure*. We assign to every distance function d a measure $\sum d$ by defining $(\sum d)(w)$ as the maximal sum of d-values on the paths from w to any leaf:

$$\sum \boldsymbol{d(w)} := \max \left\{ \sum_{\substack{v \in \operatorname{nds}(T) \\ w \leqslant v < w'}} d_{w'}(v) : w' \in \operatorname{lvs}(T) \wedge w \leqslant w' \right\}.$$

On "standardized" measures and distance functions the operations Δ and $\sum$ are inverse to each other,[32] and thus for trees the notions of "measure" and "distance function" are (essentially) equivalent. However we consider the concept of a distance function as more basic, since given a measure μ one can easily calculate the

[30] Perhaps "potential" would be a somewhat better choice.

[31] $\operatorname{ds}_T(w) = \{w'\}$.

[32] Measures $\mu = \sum d$ induced by distance functions are 0 on all leaves, while for distance functions $d = \Delta\mu$ induced by measures the sum of d-values on all paths from the root to a leaf is the same – now for measures μ and distance functions d with these properties we have $\Delta\sum d = d$ and $\sum\Delta\mu = \mu$.

corresponding distance function $\Delta\mu$ by *local operations*, but to obtain from a given distance function d the corresponding measure $\sum d$ involves *global calculations* over the whole tree.

Furthermore the measures and distance functions used in this paper[33] do not depend on the tree, but only on the formulas labeling the nodes, and then the concept of distance functions is in fact more general than that of measures.

In order to be able to transform any distance function into a transition probability, we introduce the "τ-*function*" on "*branching tuples*" in the next definition.

Definition 8.3. A *branching tuple* t is a tuple of positive real numbers. For breadth one also 0 is allowed. The set of all branching tuples is $\mathscr{BT}$:

$$\mathscr{BT} := \bigcup_{n \geqslant 1} (\mathbb{R}_+^*)^n \cup \{(0)\}.$$

For $t = (t_1, \dots, t_m) \in \mathscr{BT} \setminus \{(0)\}$ we define $\boldsymbol{\tau(t)}$ as the unique positive solution of the equation

$$\sum_{i=1}^{m} \tau(t)^{-t_i} = 1,$$

while for (0) we define $\tau((0)) := 1$.

Given a tree T with a distance function d, we induce a mapping $d : \mathrm{nds}_1(T) \to \mathscr{BT}$ by

$$\boldsymbol{d(w)} := (d(w, w'))_{w' \in \mathrm{ds}_T(w)}.$$

(Considering $d(w)$ as a branching tuple, we use in fact the notion of "branching tuples modulo permutations of the components", which is justified by the invariance of the τ-function against permutation.)

Now for $w \in \mathrm{nds}_1(T)$ also $\boldsymbol{\tau(d(w))}$ is well defined, gauging the cost of a single branching w. We obtain a "worst-case gauging" $\boldsymbol{\tau_{\max}(d, T)}$ of whole T by

$$\boldsymbol{\tau_{\max}(d, T)} := \max(\{\tau(w)\}_{w \in \mathrm{nds}_1(T)} \cup \{1\}).$$

As basic properties of τ just note:

1. $\tau : \mathscr{BT} \to [1, +\infty[$;
2. $\tau((t_1, \dots, t_m)) = 1 \Leftrightarrow m = 1$;
3. $\tau((t_1, \dots, t_m)) = \tau((t_{\pi(1)}, \dots, t_{\pi(m)}))$ for any permutation π;
4. $\tau((t_1, t_2, \dots, t_m)) < \tau((t_1', t_2, \dots, t_m))$ for $m \geqslant 2$ and $t_1 > t_1'$.

Lemma 8.2. *Consider a tree T with a distance function d.*

From d we derive a "transition probability" p^d by defining

$$p^d(w, w') := \tau(d(w))^{-d(w, w')}$$

for $w \in \mathrm{nds}(T)$ and $w' \in \mathrm{ds}_T(w)$.

[33] Or (implicitly) elsewhere in the literature.

The estimation of Lemma 8.1 *now can be extended to*

$$\#\mathrm{lvs}(T) \leqslant \max_{w \in \mathrm{lvs}(T)} p^d(w)^{-1} \leqslant \tau_{\max}(d,T)^{\sum d(\mathrm{root}(T))}.$$

Proof.

$$\begin{aligned}\max_{w \in \mathrm{lvs}(T)} p^d(w)^{-1} &= \max_{w \in \mathrm{lvs}(T)} \prod_{v<w} p^d_w(v)^{-1} = \max_{w \in \mathrm{lvs}(T)} \prod_{v<w} \tau(d(v))^{d_w(v)}\\ &\leqslant \max_{w \in \mathrm{lvs}(T)} \prod_{v<w} \tau_{\max}(d,T)^{d_w(v)} = \max_{w \in \mathrm{lvs}(T)} \tau_{\max}(d,T)^{\sum_{v<w} d_w(v)}\\ &= \tau_{\max}(d,T)^{\sum d(\mathrm{root}(T))}. \qquad \square\end{aligned}$$

Preliminary versions of this "τ-lemma" can be found in [8, 23]. The benefits of this new lemma, compared with its predecessors, are as follows:

1. One can handle arbitrary positive *real numbers* as edge-labelings and thus optimize d.
2. More generally, for the first time we consider the *whole* d as one object, enabling us to perform *global* optimizations on d, involving the whole tree, which compares favorably to the use of recursion equations.

In all the τ-lemma for the special purpose of estimating tree sizes improves the ordinary method of recursion equations, allowing to handle globally the situation where a set of *alternative recursion* equations is given with possible interplay between the different cases. Optimizations with continuous parameters are supported, and a framework for taking global properties of the considered trees into account is established.

On the contrary by using recursion equations the original tree in effect is substituted by a new one, consisting of the branchings belonging to the (different) recursion equations, and thus, according to this loss of information, the use of an *arbitrary* distance function d in the τ-lemma makes no sense anymore since we cannot even *formulate* the notion of the maximal sum of d-values over all paths in the *original* tree.

The main point in using the τ-lemma is the right choice of the distance function d. In Section 14.2 we briefly discuss this subject from a general point of view.

For 3-SAT-algorithms the definition of an appropriate distance function d is the subject of the subsequent Section 9.

An additional handling of nodes with "bad" τ-value but with predecessors or successors with "good" τ-value is presented in Section 12.

8.3. Using the τ-function for selecting branchings

The τ-function seems to be the right tool for the purpose of selecting a branching from a set of alternatives in backtracking SAT-algorithms, provided a "good" measure of formula complexity is given, which can guide the "greedy" backtracking algorithm. (See [19].)

In our 3-SAT-algorithm $\mathscr{N}_3$ the τ-function is *not directly* used, since the calculation of the corresponding branching tuples seems to be too expensive, but with the help of certain simplifying assumptions the τ-valuations *guided the design* of the algorithm. For example now we motivate the exceptional case in which in case (A) of $\mathcal{N}_3$ the originally proposed branching is replaced by the "Generalized Autarkness-Branching" (see instruction (3) in "branchingA").

In this special case the branching (φ_0, φ_1) is replaced by the branching

$$(\langle a \to 0,\, b \to 0,\, x \to 1\rangle, \langle a \to 0,\, b \to 1\rangle \cup \varphi, \langle a \to 1\rangle \cup \varphi),$$

where φ is φ_0 or φ_1, $N(\varphi, F_0) = \{\{a,b\}\}$ and $\{a,b,x\} \in F_0$ with $\varphi(x) = 0$ (see Section 6.3.2). Let (p,q) be Δn for the original branching. W.l.o.g.: $p = n(\varphi)$. Then Δn for the new branching is (at least) $(3, p+2, p+1)$.

Lemma 8.3. *For $p, q \in \mathbb{N}$ with $p \geqslant 2$ and $(p-1)\cdot(q-1) \leqslant 1$ we have*

$$\tau(p,\, q) > \tau(3,\ p+2,\ p+2).$$

Proof. First assume $q = 1$. Then $\tau(p,q) = \tau(p,1) = \tau(p, 1+p, 1+1)$ (see Lemma 8.4) $= \tau(p, p+1, 2) > \tau(p+2, p+1, 3)$.

Otherwise we have $p = q = 2$. Now $\tau(p,q) = \tau(2,2) = \tau(2, 2+2, 2+2)$ (see Lemma 8.4) $= \tau(2,4,4) > \tau(2+1, 4-1, 4)$ (see Lemma 8.5) $= \tau(3,3,4) = \tau(3, 2+2, 2+1)$. $\square$

The first auxiliary lemma used in the proof states that the τ-value of a composed branching tuple t^*, that is, a branching tuple t' is concatenated to one branch of another branching tuple t, lies between the τ-values of the parts t, t'.

Lemma 8.4. *Consider two arbitrary branching tuples $t = (t_1, \dots, t_m)$ and $t' = (t'_1, \dots t'_n)$. Then for $t^* := (t_1 + t'_1, \dots, t_1 + t'_n,\, t_2, \dots, t_m)$ we have*:
(i) $\tau(t') = \tau(t) \Rightarrow \tau(t^*) = \tau(t)$,
(ii) $\tau(t') < \tau(t) \Rightarrow \tau(t') < \tau(t^*) < \tau(t)$,
(iii) $\tau(t') > \tau(t) \Rightarrow \tau(t') > \tau(t^*) > \tau(t)$.

Proof. Directly from the definition of the τ-function. $\square$

The second auxiliary lemma essentially states that for two binary branching tuples with the same sum of the components that tuple which is more symmetric (has a smaller distance between the two components) has the smaller τ-values.

Lemma 8.5. *Consider two branching tuples t and t' of the same breadth which coincide except of two components*:

$$t = (t_1, \dots, t_n), \qquad t' = (t'_1, \dots, t'_n), \quad n \geqslant 2, \qquad t_i = t'_i \ \text{ for } i \geqslant 3.$$

Assume $t'_1 + t'_2 \geqslant t_1 + t_2$ and $\min(t'_1, t'_2) \geqslant \min(t_1, t_2)$. Then we have $\tau(t') \leqslant \tau(t)$. And if additionally $t'_1 + t'_2 > t_1 + t_2$ or $\min(t'_1, t'_2) > \min(t_1, t_2)$ holds, then we have $\tau(t') < \tau(t)$.

Proof. For a branching tuple $t=(t_1,\dots,t_n)$ let

$$\chi(t)(x):=\sum_{i=1}^{n} x^{-t_i}.$$

Now $\chi(t)-\chi(t')=\chi((t_1,t_2))-\chi((t'_1,t'_2))$ holds, and in general we have:

$$\chi((t'_1,t'_2))\leqslant\chi((t_1,t_2))\Leftrightarrow(t'_1+t'_2\geqslant t_1+t_2 \text{ and } \min(t'_1,t'_2)\geqslant\min(t_1,t_2)). \qquad \square$$

9. The distance function

The aim of this section is to motivate, define and discuss the distance function $\boldsymbol{d^3}$ for $\mathscr{T}_{\mathscr{N}_3}(F_0)$, the main tool for the analysis of $\mathscr{N}_3$. For that purpose the following conventions are useful.

Definition 9.1. Every "formula measure" $\mu:3\text{-}\mathscr{CLS}\to\mathbb{R}$ induces a node labeling of $\mathscr{T}_{\mathscr{N}_3}(F_0)$ via $\boldsymbol{\mu(w)}:=\mu(F_0(w))$ for $w\in\mathrm{nds}(\mathscr{T}_{\mathscr{N}_3}(F_0))$, as well as any "formula distance function" $d:(3\text{-}\mathscr{CLS})^2\to\mathbb{R}$ induces an edge labeling of $\mathscr{T}_{\mathscr{N}_3}(F_0)$ via $\boldsymbol{d(w,w')}:=d(F_0(w),F_0(w'))$ for $w\in\mathrm{nds}(T)$ and $w'\in\mathrm{ds}_T(w)$. (In this context μ and d are arbitrary mappings.)

9.1. The hitherto existing distance functions

The problem of Section 9 is to find a ("good") distance function d fulfilling

$$\sum d(\mathrm{root}\,(\mathscr{T}_{\mathscr{N}_3}(F_0)))\leqslant n(F_0)$$

since we are interested in an upper bound depending on $n(F_0)$.

The first approach is to take $d^0=\Delta n$. Using d^0 [23, 25] (exploiting Lemma 4.1) obtained the bound $\tau_{\max}(d^0,\,\mathscr{T}_{\mathrm{L/MS}})\leqslant 1.6180\ldots=\tau(2,1)=\tau_{0,1,1}$ (see Definition 11.2) for their underlying computation tree $\mathscr{T}_{\mathrm{L/MS}}$.

In my opinion the improved bound obtained by [29] is based essentially on the (implicit) use of the refined distance function $d^1=\Delta m_1$ where

$$m_1(F):=n(F)-\alpha\cdot z_1(F),$$

$$z_1(F):=\max\{|P|:\ P\subseteq F^{[2]}\wedge\forall\, C,C'\in P\ [\,C\neq C'\Rightarrow\mathrm{Var}(C)\cap\mathrm{Var}(C')=\emptyset]\}.$$

Here $\alpha\in\mathbb{R}_+$ is a parameter to be chosen s.t. $\tau_{\max}$ becomes minimal. [29] proved (in our interpretation) $\tau_{\max}(d^1,\,\mathscr{T}_{\mathrm{Sch}}(F_0))\leqslant\tau_{1,2,1}=1.5780\ldots$.

The advantage of m_1 over n is that an *increase* in the number of 2-clauses can *increase* Δm_1 (while Δn stays unchanged) and thus *decreases* $\tau_{\max}(\Delta m_1,\,\mathscr{T}_{\mathrm{S}}(F_0))$. The algorithm therefore must try to produce as *many new* 2-clauses as possible (for [23, 25] the mere *existence* of a 2-clause was sufficient) while not eliminating to many old 2-clauses, and in case more 2-clauses vanish than new 2-clauses arise, an increase of Δn (due to the special situation) must compensate this loss.

9.2. The generalized "approximation" of the number of 2-clauses

Definition 9.2. We generalize the measure z_1 to $\boldsymbol{z_k} : 3\text{-}\mathcal{CLS} \to \mathbb{N}_0$ for $k \in \mathbb{N}_0 \cup \{+\infty\}$:

$$\boldsymbol{z_k(F)} := \max\{|P|:\ P \subseteq F^{[2]} \wedge \rho(P) \leqslant k\}.$$

We consider $z_k(F)$ as an "approximation" of $z_\infty(F) = |F^{[2]}|$. We cannot use $z_\infty(F)$ itself because the number of (arbitrary) 2-clauses eliminated by (applying) a partial assignment φ cannot be bounded in $n(\varphi)$ only. By way of contrast the number of vanishing "z_k-clauses" is less than $k \cdot n(\varphi)$ (see Lemma 9.6).

Some basic properties of z_k (and ρ) are given in the next lemma.

Lemma 9.1. *For all $k, k' \in \mathbb{N}_0 \cup \{+\infty\}$, clauses C with $|C| \leqslant 3$ and $F, F' \in 3\text{-}\mathcal{CLS}$ the following holds*:
1. *ρ is monotonic*: $F \subseteq F' \Rightarrow \rho(F) \leqslant \rho(F')$;
2. *$z_k(F)$ is monotonic in k and F* : $k \leqslant k' \wedge F \subseteq F' \Rightarrow z_k(F) \leqslant z_{k'}(F')$;
3. $k \geqslant \rho(F) \Rightarrow z_k(F) = z_{\rho(F)}(F)$;
4. (a) $\rho(F) \leqslant \rho(F \cup \{C\}) \leqslant \rho(F) + 1$;
   (b) $z_k(F) \leqslant z_k(F \cup \{C\}) \leqslant z_k(F) + 1$;
5. (a) $z_k(F) < z_{k'}(F) \Leftrightarrow \min(\rho(F), k') \geqslant k + 1$;
   (b) $\rho(F) = \min\{k \in \mathbb{N}_0:\ z_k(F) = z_{k+1}(F)\}$;
6. (a) $\rho(F) \leqslant z_\infty(F) \leqslant \frac{1}{2} \cdot \rho(F) \cdot n(F)$;
   (b) $2 \cdot z_\infty(F)/n(F) \leqslant \rho(F) \leqslant \min(z_\infty(F), 4 \cdot (n(F) - 1))$ *for* $n(F) \geqslant 1$;
   (c) $\min(k, z_\infty(F)) \leqslant z_k(F) \leqslant \frac{1}{2} \cdot \min(k, \rho(F)) \cdot n(F)$.

Proof. Parts 1–4 are obvious from the definition. Part 5(a) "$\Rightarrow$" follows from parts 2 and 3, and "$\Leftarrow$" follows by part 4(a). Part 5(b) is an immediate consequence of 5(a). For part 6(a) note that no clause contains complementary literals and

$$z_\infty(F) = \tfrac{1}{2}\ell(F^{[2]}) \leqslant \tfrac{1}{2}\rho(F) \cdot n(F).$$

For $v \in \mathrm{Var}(F)$ we have

$$\rho(F) \leqslant |\{\{a, b\} \in \mathcal{CL}:\ \mathrm{Var}(a) = v \wedge \mathrm{Var}(b) \in \mathrm{Var}(F) \backslash \{v\}\}| = 2 \cdot 2 \cdot (n(F) - 1)$$

and hence by part (a) part (b) follows. Finally $\min(k, z_\infty(F)) \leqslant z_k(F)$ follows from $\rho(P) \leqslant |P|$ for $P \subseteq F^{[2]}$, and for $P \subseteq F^{[2]}$ s.t. $\rho(P) \leqslant k$ and $|P| = z_k(F)$ holds we have (by part (a)):

$$z_k(F) = |P| \leqslant \tfrac{1}{2}\rho(P) n(F) \leqslant \tfrac{1}{2}\min(k, \rho(F)) \cdot n(F). \qquad \square$$

Using the "approximation parameter" k we generalize the measure m_1:

Definition 9.3. For $F \in 3\text{-}\mathcal{CLS}$ and $\alpha \in \mathbb{R}_+$ with $\alpha < 2/k$ we define

$$\boldsymbol{m_k(F)} := n(F) - \alpha \cdot z_k(F).$$

By the distance function $d^2 := \Delta m_k = \Delta n - \alpha \cdot \Delta z_k$ now

- we can use an *optimal level k of approximation* of the number of 2-clauses (e.g. for our complexity bound $k = 2$ is optimal).

However, it is not the end of the story:

Our main problem is that new 2-clauses do not need to contribute to Δz_k, since z_k counts only *certain* 2-clauses and we do not know which.

An ad hoc approach used in [29] (for z_1) to overcome this difficulty is as follows: Consider a branch $w \to w'$ where some new 2-clause did not account to Δz_k. Due to Lemma 9.1 part 5(a) then a variable must occur more than k times in the 2-clauses of $F_0(w')$, and thus it is advantageous to form an aggregate τ–value out of these consecutive branchings,[34] since at w' we obtain a good Δn-value. But even for z_1 this job is tedious, causes a (too) complicated algorithm, and gives away all long-range effects.

An alternative definition of z_k by "maximal k-matchings". We define an undirected graph $G(F)$ for $F \in 3\text{-}\mathcal{CLS}$, where the set of nodes is $\operatorname{Var}(F)$ and every 2-clause $\{a, b\} \in F^{[2]}$ connects the nodes $\operatorname{Var}(a), \operatorname{Var}(b)$ (parallel edges are allowed).

Now $z_k(F)$ is the maximal size of a "k-matching" in $G(F)$, where a k-matching is a set M of edges such that every node of $G(F)$ is adjacent to at most k edges from M.

9.3. Using "budgets" for new 2-clauses

Here a general method is introduced for bringing new 2-clauses into account which (implicitly) takes the whole computation tree into consideration and completes (at least for the moment) our way of approximating the number of 2-clauses.

We introduce the distance function $\boldsymbol{d^3 = d^3(k, (h_i)_{i \in \mathbb{N}_0}, \alpha)}$ for $\mathcal{T}_{\mathcal{N}_3}(F_0)$, depending additionally on "*budgets*" (h_i) allowing

- to take for a branch $w \to w'$ up to $h_{\rho(w)}$ new 2-clauses definitely into account (but not more).

  We require $h_i = 0$ for $i > k$. Thus we divide the nodes w into two categories:

  – If $\rho(w) \leqslant k$ holds then an increase in the number of 2-clauses improves $\tau(d^3(w))$ over $\tau(\Delta n(w))$.

  – But if $\rho(w) > k$ holds then $\tau(d^3(w))$ gets worse (compared with $\tau(\Delta n(w))$) and indeed the worse the bigger the h_i are.

This partition is motivated by the fact that due to the branching rule of "branchingA" (see Section 6.3.1) an increasing $\rho(w)$ causes an increasing $\Delta n(w)$ (at least in the worst case).

More precisely we have the following case distinction in mind:

– The case $\rho(w) = 0$ is covered by br-Autarkness.

– In case $\rho(w) = 1$ in both branches created by "branchingB" (see Section 6.4) only one 2-clause vanishes and thus the number of new 2-clauses (at least one by

[34] According to Lemma 8.4 a "mean value" of the τ-values of the single branchings w and w'.

br-Autarkness) determines an increase in the number of 2-clauses: $\Delta_{w'} z_\infty(w) = v_{w'}(w) - 1 \geqslant 0$, where $v_{w'}(w) := z_\infty(w') - z_\infty(F_0(w') \cap F_0(w))$ is the number of new 2-clauses.

– For case $1 < \rho(w) \leqslant k$, belonging to "branchingA" (see Section 6.3), the number of eliminated 2-clauses (increasing with $\rho(w)$) in fact is greater than the number of new 2-clauses (at least in the worst case and for our analysis), thus $\tau(d^3(w)) > \tau(\Delta n(w))$, but the new 2-clauses provided by Generalized Autarkness prevent the τ–value from being too bad.
– For $\rho(w) > k$ only the negative contribution by the eliminated 2-clauses is counted, but, due to the restriction of d^3 to an *approximation* of z_∞, the case $\rho(w) = k+1$ is the worst.[35]

The cases $\rho(w) = 1$ and $\rho(w) = k+1$ are the overall worst cases, and for $\rho(w) = 1$ the weight α should be as large as possible while for $\rho(w) = k+1$ as small as possible. The more new 2-clauses we have in case $\rho(w) = 1$ the bigger is the optimal k (using an optimal α).

9.4. The definition of $d^3 = \Delta n - \alpha \cdot \zeta$

For our handling of the variation in the number of 2-clauses the following differentiation between eliminated and new 2-clauses is basic.

Definition 9.4. For $i \in \mathbb{N}_0 \cup \{+\infty\}$ and $F, F' \in 3\text{-}\mathscr{CLS}$ we define:

$$\boldsymbol{\delta z_i(F,F')} := z_i(F \setminus F')$$
$$\boldsymbol{v(F,F')} := z_\infty(F' \setminus F).$$

$\delta z_i(F,F')$ is the number of "z_i-clauses" eliminated by the transition $F \to F'$, and $v(F,F')$ is the (total) number of new clauses created by this transition. We have

$$\Delta z_\infty = \delta z_\infty - v.$$

That only for the calculation of eliminated clauses the "approximations" z_i are used, while for the calculation of new clauses all new clauses are considered, mirrors a fundamental asymmetry of our approach:

While the number of *eliminated* 2-clauses has to be restricted to make it depend on Δn, we want, on the other hand, to take an arbitrary number of *new* 2-clauses into account (at least in principal). (The restrictions we have to pose on the number of "acceptable" new clauses do not depend on Δn but on ρ.)

Definition 9.5. Assume $\boldsymbol{k} \in \mathbb{N}_0$, $\boldsymbol{h} : \mathbb{N}_0 \to \mathbb{N}_0$ and $\boldsymbol{\alpha} \in \mathbb{R}_+$ fulfilling:
(i) $\forall i \in \mathbb{N}_0\ [i > k \Rightarrow h_i = 0]$,
(ii) $\alpha < (k + \max_{i \in \{0,\dots,k\}} h_i)^{-1}$ $(\frac{1}{0} := +\infty)$.

[35] Also in these cases the new clauses provided by Generalized Autarkness are useful, but only for restricting the costs for providing the budgets.

We define $\boldsymbol{\zeta(k,h)(F,F')}$ and $\boldsymbol{d^3(k,h,\alpha)(F,F')}$ for $F,F' \in 3\text{-}\mathscr{CLS}$ as follows:

$$\begin{aligned}\boldsymbol{\zeta(k,h)(F,F')} &:= \max\Bigg(\min\left(\delta z_\infty(F,F'),\ \max_{i\in\{0,\dots,k\}}(h_i + \delta z_i(F,F'))\right) - v(F,F'),\\ &\qquad\qquad \delta z_k(F,F') - h_{\rho(F)}\Bigg)\\ \boldsymbol{d^3(k,h,\alpha)} &:= \Delta n - \alpha\cdot\zeta(k,h).\end{aligned}$$

Remarks. 1. In the following we use $\zeta = \zeta(k,h)$ and $d^3 = d^3(k,h,\alpha)$.
2. Because of $\alpha \geqslant 0$, for the sake of a good $\tau_{\max}(d^3, \mathscr{T}_{\mathcal{N}_3}(F_0))$-value ζ should be as small (negative) as possible and of course Δn should be as large as possible.
3. Comparing d^3 with d^2 we see that ζ replaces z_k. And the reader should furthermore note the structural similarity between the definition of ζ and the equation $\Delta z_\infty = \delta z_\infty - v$, that is, ζ is the maximum of two versions of that equation.
4. Contrary to the hitherto distance functions there is no *formula measure* $\mu : 3\text{-}\mathscr{CLS} \to \mathbb{R}$ s.t. $d^3 = \Delta\mu$ holds.[36]

The definition of ζ combines two cases, as shown in the next lemma.

Lemma 9.2. *For $F,F' \in 3\text{-}\mathscr{CLS}$ we have, using $\delta z_i = \delta z_i(F,F')$, $v = v(F,F')$ and $\zeta = \zeta(F,F')$:*
1. $\rho(F) \leqslant k \Rightarrow \zeta = \delta z_k - \min(v, h_{\rho(F)})$;
2. $\rho(F) > k \Rightarrow \zeta = \max(\min(\delta z_\infty, \max_{i\in\{0,\dots,k\}}(h_i + \delta z_i)) - v,\ \delta z_k)$.

Proof. The easy proofs are left as an exercise to the reader (for part 1 just note that because of $\rho \leqslant k$ we have $\delta z_\infty = \delta z_k$). $\square$

In order to prove that d^3 is a distance function for $\mathscr{T}_{\mathcal{N}_3}(F_0)$ we need an upper bound for δz_k.

9.5. A general upper bound for the numbers of eliminated clauses

The aim of this subsection is to give a general upper bound for the number of eliminated z_k-clauses (with arbitrary k) when applying partial assignments or the reduction operator r.

Lemma 9.3. *For $k \in \mathbb{N}_0 \cup \{+\infty\}$ we have*
1. *For $F,F',F'' \in 3\text{-}\mathscr{CLS}$: $\delta z_k(F,F') \leqslant \delta z_k(F,F'') + \delta z_{\min(k,\rho(F))}(F'',F')$.*
2. *And for a sequence $F_0,\dots,F_s$, $s \in \mathbb{N}_0$ of clause-sets $F_i \in 3\text{-}\mathscr{CLS}$:*

$$\delta z_k(F_0,F_s) \leqslant \sum_{i=0}^{s-1} \delta z_{\min(k,\rho(F_0),\dots,\rho(F_i))}(F_i,F_{i+1}).$$

[36] One reason is that δz_i does not fulfill the "triangle equality" $\delta z_i(F,F') = \delta z_i(F,F'') + \delta z_i(F'',F')$ as it would be the case for $\delta z_k = \Delta\mu$, but only the triangle *inequality* (see Lemma 9.3).

Proof. *Part* 1: The reader should be easily able to verify that (generally)

$$\delta z_k(F,F') = \max\{|P| - |P \cap F'|: \ P \subseteq F^{[2]} \wedge \rho(P) \leqslant k\}$$

holds. Now for $P \subseteq F^{[2]}$ with $\rho(P) \leqslant k$ we have the estimations

$$|P| - |P \cap F''| \leqslant \delta z_k(F, F''),$$

$$|P \cap F''| - |(P \cap F'') \cap F'| \leqslant \delta z_{\min(k,\rho(F))}(F'', F').$$

Adding the inequalities yields $|P| - |P \cap F'' \cap F'| \leqslant \delta z_k(F,F'') + \delta z_{\min(k,\rho(F))}(F'',F')$ and thus by $|P \cap F''| \geqslant |P \cap F'' \cap F'|$ the assertion follows.

Part 2: $s=0$: $0 \leqslant 0$.

$s>0$: By part 1 and the induction hypothesis we have:

$$\delta z_k(F_0, F_s) \leqslant \delta z_k(F_0, F_1) + \delta z_{\min(k,\rho(F_0))}(F_1, F_s) \leqslant \delta z_k(F_0, F_1) + \sum_{i=1}^{s-1} \delta z_{\min(k,\rho(F_0),\rho(F_1),\ldots,\rho(F_i))}(F_i, F_{i+1}). \qquad \square$$

Lemma 9.3 enables us to estimate $\delta z_k(F,F')$ by dividing the transformation $F \to F'$ into smaller steps $F = F_0 \to \cdots \to F_n = F'$ for which we have appropriate estimations. Note that a transformation $F_{i+1} = \varphi_i * F_i$ in the situation $\varphi_i = \varphi_i' \cup \varphi_i''$ can be further divided via $F_{i+1} = \varphi_i' * (\varphi_i'' * F)$.

The next lemma estimates the total number of 2-clauses eliminated by a partial assignment.

Lemma 9.4. *For $F,F' \in 3\text{-}\mathcal{CLS}$ with $F \subseteq F'$ and $V \subseteq \mathcal{VA}$, $\varphi \in \mathcal{PASS}$ the following holds*:
1. $F \setminus (\varphi * F') = \{C \in F: \ \mathrm{Var}(C) \cap \mathrm{Var}(\varphi) \neq \emptyset\}$ (*and thus* $F \setminus (\varphi * F') = F \setminus (\varphi * F)$);
2. $|\{C \in F^{[2]}: \ \mathrm{Var}(C) \cap V \neq \emptyset\}| \leqslant |V| \cdot \rho(F)$.

(*In words*: *The clauses, which are eliminated by φ, are those clauses which are effected* (*at all*) *by φ. And the number of* 2-*clauses, which are affected by φ, is less than or equal to the number of variables eliminated by φ times the maximal number $\rho(F)$ of occurrences of a variable in the* 2-*clauses of F.*)

Proof. *Part* 1 follows from $\varphi * \{C\} = \{C\}$ for $\mathrm{Var}(\varphi) \cap \mathrm{Var}(C) = \emptyset$, and $\mathrm{Var}(\varphi) \cap \mathrm{Var}(\varphi * F) = \emptyset$.

Part 2 : $|\{C \in F^{[2]}: \ \mathrm{Var}(C) \cap V \neq \emptyset\}| \leqslant \sum_{v \in V} (\#_v^2 + \#_{\overline{v}}^2)(F) \leqslant \sum_{v \in V} \rho(F) = |V| \cdot \rho(F)$. $\square$

The main result of this subsection now is given in the following lemma.

Lemma 9.5. *Assume $k \in \mathbb{N}_0 \cup \{+\infty\}$. Then we have*:
1. *For $F \in 3\text{-}\mathcal{CLS}$, $\varphi \in \mathcal{PASS}$*: $\delta z_k(F, \varphi * F) \leqslant n(\varphi) \cdot \min(k, \rho(F))$.
2. *Assume a sequence $F_0, \ldots, F_s$, $s \in \mathbb{N}_0$ of clause-sets $F_i \in 3\text{-}\mathcal{CLS}$.*

(a) *Assume furthermore that there is* $I \subseteq \{0, \ldots, s-1\}$ *and* $\varphi : I \to \mathscr{PASS}$ *s.t.*

$$\forall_{i \in I}[F_{i+1} = \varphi_i * F_i] \wedge \forall_{i \in \{0,\ldots,s-1\} \setminus I}[F_i^{[2]} \subseteq F_{i+1}^{[2]}].$$

Then: $\delta z_k(F_0, F_s) \leqslant \sum_{i \in I} n(\varphi_i) \cdot \min(k, \rho(F_0), \ldots, \rho(F_i))$.

(b) *If* $\forall_{i \in \{0,\ldots,s-1\}}[F_{i+1} = r(F_i) \vee \exists_{\varphi \in \mathscr{PASS}}[F_{i+1} = \varphi * F_i]]$ *holds, then*:

$$\delta z_k(F_0, F_s) \leqslant (n(F_0) - n(F_s)) \cdot \min(k, \rho(F_0)).$$

Proof. *Part* 1: Take $P \subseteq F^{[2]}$ with $\rho(P) \leqslant k$. Then $|P \setminus (\varphi * P)| \leqslant n(\varphi) \cdot \rho(P)$ (by Lemma 9.4) $\leqslant n(\varphi) \cdot \min(k, \rho(F))$.

Part 2(a) follows immediately by Lemma 9.3 part 2 and by part 1. And using Lemma 5.1 part 2 we get part 2(b) from part 2(a). □

Since $F_0(w')$ for a successor $w' \in \mathrm{ds}_T(w)$ of a node $w \in \mathrm{nds}(\mathscr{T}_{\mathcal{N}_3}(F_0))$ results from $F_0(w)$ by a series of applications of partial assignments and r, we can apply Lemma 9.5 part 2(b):

Lemma 9.6. *For* $w \in \mathrm{nds}(\mathscr{T}_{\mathcal{N}_3}(F_0))$, $w' \in \mathrm{ds}(w)$ *and* $k \in \mathbb{N}_0 \cup \{+\infty\}$ *the number of eliminated "z_k-clauses" in branch* $w \to w'$ *is less than or equal to the number of eliminated variables times the minimum of parameter k and the maximal number* $\rho(w)$ *of occurrences of a variable in the* 2*-clauses of* $F_0(w)$:

$$\delta_{w'} z_k(w) \leqslant \Delta_{w'} n(w) \cdot \min(k, \rho(w)).$$

(*We use* $\boldsymbol{\delta_{w'} z_k(w)} := (\delta z_k)_{w'}(w)$).

We conclude this section with some bounds on ζ and d^3 and proving that d^3 is indeed a distance function for $\mathscr{T}_{\mathcal{N}_3}(F_0)$.

Lemma 9.7. 1. *For* $w \in \mathrm{nds}(\mathscr{T}_{\mathcal{N}_3}(F_0))$ *and* $w' \in \mathrm{ds}(w)$ *we have*

(a) $\zeta_{w'}(w) \leqslant \delta_{w'} z_k(w) + \max_{i \in \{0,\ldots,k\}} h_i \leqslant k \cdot \Delta_{w'} n(w) + \max_{i \in \{0,\ldots,k\}} h_i$.

(b) $d^3_{w'}(w) \geqslant \Delta_{w'} n(w) \cdot (1 - k \cdot \alpha) - \alpha \cdot \max_{i \in \{0,\ldots,k\}} h_i$.

2. d^3 *is a distance function for* $\mathscr{T}_{\mathcal{N}_3}(F_0)$.

Proof. *Part* 1(a): For the first inequality we have

$$\begin{aligned}
&\max(\min(\delta_{w'} z_\infty(w), \max_{i \in \{0,\ldots,k\}}(h_i + \delta_{w'} z_i(w))) - v_{w'}(w), \delta_{w'} z_k(w) - h_{\rho(w)})\\
&\leqslant \max\left(\max_{i \in \{0,\ldots,k\}}(h_i + \delta_{w'} z_i(w)) - v_{w'}(w), \delta_{w'} z_k(w) - h_{\rho(w)}\right)\\
&\leqslant \max\left(\max_{i \in \{0,\ldots,k\}}(h_i + \delta_{w'} z_k(w)), \delta_{w'} z_k(w)\right) = \delta_{w'} z_k(w) + \max_{i \in \{0,\ldots,k\}} h_i.
\end{aligned}$$

The second inequality is a consequence of Lemma 9.6. Part 1(b) follows immediately from part (a), and part 2 follows from part 1(b), condition (ii) of Definition 9.5 and the fact $\Delta_{w'} n(w) \geqslant 1$. □

10. Bounding the sum of d^3

Consider $\zeta = \zeta(k, h)$ and $d^3 = d^3(k, h, \alpha)$ (see Definition 9.5). This section is devoted to the proof of

$$\sum d^3(\omega_0) \leqslant n(F_0) - \cdots,$$

where

Definition 10.1. $\omega_0 := \mathrm{root}(\mathscr{T}_{\mathcal{N}_3}(F_0))$ is the root of the computation tree,

and ... is some negligible quantity.

The idea is to study paths $(w_1, \ldots, w_t)$ in $\mathscr{T}_{\mathcal{N}_3}(F_0)$ with $\rho(w_1), \rho(w_t) \leqslant k$ and $\rho(w_i) > k$ for $1 < i < k$, that means at the endpoints w_1 and w_t the distance function Δn can be improved with the help of ζ while at the intermediate nodes $w_2, \ldots, w_{t-1}$ "for compensation" Δn is impaired by ζ. For those paths we show (Lemma 10.4):

$$\sum_{i=1}^{t-1} \zeta_{w_t}(w_i) \geqslant \sum_{i=1}^{t-1} \Delta_{w_t} z_k(w_i)$$

and thus

$$\sum_{i=1}^{t-1} d^3_{w_t}(w_i) = \sum_{i=1}^{t-1} \Delta_{w_t} n(w_i) - \alpha \cdot \sum_{i=1}^{t-1} \zeta_{w_t}(w_i) \leqslant \sum_{i=1}^{t-1} \Delta_{w_t} n(w_i) - \alpha \cdot \sum_{i=1}^{t-1} \Delta_{w_t} z_k(w_i)$$

$$= \sum_{i=1}^{t-1} \Delta_{w_t} m_k(w_i) = m_k(w_1) - m_k(w_t) = n(w_1) - \alpha \cdot z_k(w_1) - m_k(w_t).$$

This result is generalized to arbitrary paths by dividing them into sub-paths (up to a start and an end section). In order to emphasize the rather general nature of the proofs in this section, in fact we will not consider paths in $\mathscr{T}_{\mathcal{N}_3}(F_0)$ but arbitrary sequences $(F_1, \ldots, F_t)$ of 3-clause-sets.[37]

We start with some (further) basic properties of z_i, δz_i and ζ.

Lemma 10.1. *For $i \in \mathbb{N}_0$ and $F, F_1, F_2, F' \in 3\text{-}\mathscr{CLS}$ we have*
1. $z_i(F_1 \cup F_2) \leqslant z_i(F_1) + z_i(F_2)$;
2. (a) $\delta z_i(F, F')$ *is increasing in* i *and* F (*w.r.t.* $\subseteq$) *and decreasing in* F';
   (b) $\delta z_\infty(F, F') = \delta z_{\rho(F)}(F, F')$;
   (c) $\delta z_i(F, F') \geqslant \Delta z_i(F, F')$;
   (d) $\delta z_i(F, F') + z_i(F \cap F') \geqslant z_i(F)$.
3. (a) $\rho(F) > k \Rightarrow (\zeta - \delta z_k)(F, F') \geqslant 0$;
   (b) $(\zeta - \delta z_k)(F, F') \geqslant -\min(v(F, F'), h_{\rho(F)})$.

[37] Furthermore, the results of this section hold for example for an arbitrary $\rho : 3\text{-}\mathscr{CLS} \to \mathbb{R}_+$ which only has to fulfill monotonicity: $F \subseteq F' \Rightarrow \rho(F) \leqslant \rho(F')$. And the results also do not depend on the notion of (3-)clause-sets.

4. (a) $(\zeta - \Delta z_k)(F,F') \geqslant -h_{\rho(F)} + (z_k - z_{\rho(F)})(F')$;
   (b) $\rho(F) \leqslant k \Rightarrow (\zeta - \Delta z_k)(F,F') \geqslant -(z_\infty - z_k)(F')$.

Proof. Parts 1 and 2 follow directly from the definition of δz_i (for 2(c) and 2(d) use 1). Part 3(a) follows from Lemma 9.2 part 2, and part 3(b) follows from part 3(a) and Lemma 9.2 part 1.

Part 4(a): By part 3(b) we can estimate

$$\begin{aligned}(\zeta - \Delta)(F,F') &= ((\zeta - \delta) + (\delta - \Delta))(F,F')\\ &\geqslant -h_{\rho(F)} + \delta z_k(F,F') - z_k(F) + z_k(F').\end{aligned}$$

By part 2(d) we get $\delta z_k(F,F') + z_{\rho(F)}(F') \geqslant z_k(F)$, since $z_k(F \cap F') \leqslant z_{\rho(F)}(F \cap F') \leqslant z_{\rho(F)}(F')$. Replacing $\delta z_k(F,F') - z_k(F)$ by $-z_{\rho(F)}(F')$ now completes the proof of 4(a).

Part 4(b): Again by part 3(b) we can estimate:

$$\begin{aligned}(\zeta - \Delta)(F,F') &= ((\zeta - \delta) + (\delta - \Delta))(F,F')\\ &\geqslant -v(F,F') + \delta z_k(F,F') - z_k(F) + z_k(F').\end{aligned}$$

Furthermore, using $\rho(F) \leqslant k$ we get $v(F,F') + z_k(F) = z_\infty(F' \setminus F) + z_k(F) = z_\infty(F' \setminus F) + z_\infty(F) = z_\infty(F') - z_\infty(F \cap F') + z_\infty(F) = z_\infty(F \setminus F') + z_\infty(F') = z_k(F \setminus F') + z_\infty(F') = \delta z_k(F,F') + z_\infty(F')$. □

Lemma 10.2. *For a sequence $(F_1,\ldots,F_u)$ of 3-clause-sets $F_i \in 3\text{-}\mathcal{CLS}$ and for $0 \leqslant a \leqslant k$:*

$$z_a(F_u) \geqslant z_k(F_u) - (z_k - z_a)(F_1) - \sum_{i=1}^{u-1} (\delta z_k - \Delta z_k)(F_i, F_{i+1}).$$

Proof. By Lemma 10.1 part 2(a) and 2(c) we only strengthen the assertion when replacing δz_k by Δz_a. Now because of

$$\sum_{i=1}^{u-1} (\Delta z_a - \Delta z_k)(F_i, F_{i+1}) = (z_a(F_1) - z_a(F_u)) - (z_k(F_1) - z_k(F_u)),$$

we are done. □

The next lemma is of central importance for this section.

Lemma 10.3. *For a sequence $(F_1,\ldots,F_t)$ of 3-clause-sets $F_i \in 3\text{-}\mathcal{CLS}$ fulfilling $\rho(F_i) > k$ for all $1 \leqslant i \leqslant t-2$*

$$\begin{aligned}\sum_{i=1}^{t-1} (\zeta - \Delta z_k)(F_i, F_{i+1}) \geqslant \min\left(z_\infty(F_1), \max_{i \in \{0,\ldots,k\}} (h_i + z_i(F_1))\right)\\ - z_k(F_1) - (z_\infty - z_k)(F_t)\end{aligned}$$

holds.

Proof. Induction on t, considering three cases: $t=1$, $t=2$ and $t\geqslant 3$.

$t=1$: $\min(z_\infty(F_1), \max_{0\leqslant i\leqslant k}(h_i+z_i(F_1)))-z_k(F_1)-(z_\infty-z_k)(F_1)\leqslant z_\infty(F_1)-z_\infty(F_1)=0$.

Case $t=2$:

$$\begin{aligned}
&(\zeta-\Delta z_k)(F_1,F_2)\\
&\quad\geqslant \min\left(\delta z_\infty(F_1,F_2), \max_{0\leqslant i\leqslant k}(h_i+\delta z_i(F_1,F_2))\right)-v(F_1,F_2)-\Delta z_k(F_1,F_2)\\
&\quad=\min\left(\delta z_\infty(F_1,F_2), \max_{0\leqslant i\leqslant k}(h_i+\delta z_i(F_1,F_2))\right)-z_\infty(F_2)+z_\infty(F_1\cap F_2)-\Delta z_k(F_1,F_2)\\
&\quad=\min\left(\delta z_\infty(F_1,F_2)+z_\infty(F_1\cap F_2), \max_{0\leqslant i\leqslant k}(h_i+\delta z_i(F_1,F_2)+z_\infty(F_1\cap F_2))\right)-z_k(F_1)\\
&\qquad-(z_\infty-z_k)(F_2).\\
&\quad\geqslant \min\left(z_\infty(F_1), \max_{0\leqslant i\leqslant k}(h_i+z_i(F_1))\right)-z_k(F_1)-(z_\infty-z_k)(F_2).
\end{aligned}$$

(with Lemma 10.1 part 2(d)).

Case $t\geqslant 3$: Using case $t=2$ we get

$$\begin{aligned}
&\sum_{i=1}^{t-1}(\zeta-\Delta z_k)(F_i,F_{i+1})\\
&\quad=\sum_{i=1}^{t-2}(\zeta-\Delta z_k)(F_i,F_{i+1})\ +(\zeta-\Delta z_k)(F_{t-1},F_t)\\
&\quad\geqslant\sum_{i=1}^{t-2}(\zeta-\Delta z_k)(F_i,F_{i+1})+\min\left(z_\infty(F_{t-1}), \max_{0\leqslant i\leqslant k}(h_i+z_i(F_{t-1}))\right)-z_k(F_{t-1})\\
&\quad\sum(z_\infty-z_k)(F_t).
\end{aligned}\tag{1}$$

Sub-case (a): $z_\infty(F_{t-1})\leqslant \max_{0\leqslant i\leqslant k}(h_i+z_i(F_{t-1}))$.

With induction hypothesis we conclude:

$$\begin{aligned}
(1)&\geqslant\min\left(z_\infty(F_1), \max_{0\leqslant i\leqslant k}(h_i+z_i(F_1))\right)-z_k(F_1)-(z_\infty-z_k)(F_{t-1})\\
&\quad+z_\infty(F_{t-1})-z_k(F_{t-1})-(z_\infty-z_k)(F_t)\\
&=\min\left(z_\infty(F_1), \max_{0\leqslant i\leqslant k}(h_i+z_i(F_1))\right)-z_k(F_1)-(z_\infty-z_k)(F_t).
\end{aligned}$$

Sub-case (b): $z_\infty(F_{t-1})\geqslant \max_{0\leqslant i\leqslant k}(h_i+z_i(F_{t-1}))$.

Here the induction hypothesis is useless:

$$(1)=\sum_{i=1}^{t-2}(\zeta-\Delta z_k)(F_i,F_{i+1})\ +\max_{0\leqslant i\leqslant k}(h_i+z_i(F_{t-1}))-z_k(F_{t-1})-(z_\infty-z_k)(F_t).$$

Choose $a \in \{0,\dots,k\}$ such that $h_a + z_a(F_1)$ is maximal. By Lemma 10.2 we continue:

$$\begin{aligned}(1) &\geqslant \sum_{i=1}^{t-2}(\zeta - \Delta z_k)(F_i, F_{i+1}) \ + h_a + z_a(F_{t-1}) - z_k(F_{t-1}) - (z_\infty - z_k)(F_t)\\
&\geqslant \sum_{i=1}^{t-2}(\zeta - \Delta z_k)(F_i, F_{i+1}) \ + h_a + z_k(F_{t-1}) - (z_k - z_a)(F_1)\\
&\quad -\sum_{i=1}^{t-2}(\delta z_k - \Delta z_k)(F_i, F_{i+1}) - z_k(F_{t-1}) - (z_\infty - z_k)(F_t)\\
&= \sum_{i=1}^{t-2}(\zeta - \delta z_k)(F_i, F_{i+1}) \ + h_a + z_a(F_1) - z_k(F_1) - (z_\infty - z_k)(F_t).\end{aligned}$$

Finally applying Lemma 10.1 part 3(a) we conclude:

$$\begin{aligned}(1) &\geqslant h_a + z_a(F_1) - z_k(F_1) - (z_\infty - z_k)(F_t)\\
&\geqslant \min\left(z_\infty(F_1), \max_{0\leqslant i\leqslant k}(h_i + z_i(F_1))\right) - z_k(F_1) - (z_\infty - z_k)(F_t). \qquad \square\end{aligned}$$

Lemma 10.4 applies the general bound of Lemma 10.3 to the case that at the beginning and the end of the sequence of 3-clause-sets we have $\rho \leqslant k$.

Lemma 10.4. *For a sequence* $(F_1,\dots,F_t)$ *of* 3-*clause-sets* $F_i \in 3\text{-}\mathscr{CLS}$ *fulfilling* $\rho(F_1)$, $\rho(F_t) \leqslant k$ *and* $\rho(F_i) > k$ *for* $2 \leqslant i \leqslant t-2$

$$\sum_{i=1}^{t-1}(\zeta - \Delta z_k)(F_i, F_{i+1}) \geqslant 0$$

holds.

Proof. The case $t = 1$ is trivial. So consider the case $t > 1$. Applying first Lemma 10.1 parts 4(a) and 4(b), and then Lemma 10.3 (using $\rho(F_t) \leqslant k$) we get

$$\begin{aligned}&\sum_{i=1}^{t-1}(\zeta - \Delta z_k)(F_i, F_{i+1})\\
&\quad = (\zeta - \Delta z_k)(F_1, F_2) + \sum_{i=2}^{t-1}(\zeta - \Delta z_k)(F_i, F_{i+1})\\
&\quad \geqslant \max(-h_{\rho(F_1)} + (z_k - z_{\rho(F_1)})(F_2),\ -(z_\infty - z_k)(F_2)) + \sum_{i=2}^{t-1}(\zeta - \Delta z_k)(F_i, F_{i+1})\\
&\quad = \max(-h_{\rho(F_1)} + (z_k - z_{\rho(F_1)})(F_2),\ -(z_\infty - z_k)(F_2))\\
&\qquad + \min\left(z_\infty(F_2), \max_{0\leqslant i\leqslant k}(h_i + z_i(F_2))\right) - z_k(F_2). \qquad (2)\end{aligned}$$

In case $z_\infty(F_2) \leqslant \max_{0\leqslant i\leqslant k}(h_i + z_i(F_2))$ obviously (2) $\geqslant 0$ holds, and in the other case $z_\infty(F_2) \geqslant \max_{0\leqslant i\leqslant k}(h_i + z_i(F_2))$ use $i := \rho(F_1) \leqslant k$. $\square$

Finally we are able to prove the main result of this section:

Theorem 3. *For every sequence* $(F_1,\ldots,F_s)$ *of 3-clause-sets* $F_i \in 3\text{-}\mathscr{CLS}$ *we have*

$$\sum_{i=1}^{s-1} \zeta(F_i,F_{i+1}) \geqslant z_k(F_1) - z_k(F_s) - \max_{i\in\{0,\ldots,k\}} h_i.$$

Proof. The assertion is equivalent to

$$\sum_{i=1}^{s-1} (\zeta - \Delta z_k)(F_i,F_{i+1}) + \max_{i\in\{0,\ldots,k\}} h_i \geqslant 0.$$

Consider indices $0 =: n_0 < n_1 < \cdots < n_p < n_{p+1} := s+1$ with $p \geqslant 0$, $\rho(F_{n_1}),\ldots,\rho(F_{n_p}) \leqslant k$ and $\rho(F_j) > k$ for $0 \leqslant i \leqslant p$ and $n_i + 1 \leqslant j \leqslant n_{i+1} - 1$. We reformulate again the assertion:

$$\sum_{j=1}^{n_1-1} (\zeta - \Delta z_k)(F_j,F_{j+1}) + \sum_{i=1}^{p} \sum_{j=n_j}^{n_{i+1}-1} (\zeta - \Delta z_k)(F_j,F_{j+1}) + \max_{i\in\{0,\ldots,k\}} h_i \geqslant 0.$$

By Lemma 10.1 part 3(a) the first part $\sum_{j=1}^{n_1-1}$ is non-negative, and by Lemma 10.4 all $\sum_{j=n_j}^{n_{i+1}-1}$ for $1 \leqslant i \leqslant p-1$ are also non-negative. It remains to show

$$\sum_{j=n_p}^{s-1} (\zeta - \Delta z_k)(F_j,F_{j+1}) + \max_{i\in\{0,\ldots,k\}} h_i \geqslant 0.$$

In case $n_p = s$ we are done, since all h_i are non-negative. Otherwise, using Lemma 10.1 parts 3(a) and 4(a), we conclude:

$$\begin{aligned}\sum_{j=n_p}^{s-1} (\zeta - \Delta z_k)(F_j,F_{j+1}) + \max_{i\in\{0,\ldots,k\}} h_i &\geqslant (\zeta - \Delta z_k)(F_{n_p},F_{n_p+1}) + \max_{i\in\{0,\ldots,k\}} h_i \\ &\geqslant -h_{\rho(F_{n_p})} + (z_k - z_{\rho(F_{n_p})})(F_{n_p+1}) + \max_{i\in\{0,\ldots,k\}} h_i \geqslant 0. \qquad \square\end{aligned}$$

Applying Theorem 3 to $\mathscr{T}_{\mathscr{N}_3}(F_0)$ we get

Lemma 10.5. $\forall F_0 \in 3\text{-}\mathscr{CLS}_r : \sum d^3(\omega_0) \leqslant n(F_0) + \alpha \cdot \max_{i\in\{0,\ldots,k\}} h_i - \alpha \cdot z_k(\omega_0) - \min_{w\in \text{lvs}(\mathscr{T}_{\mathscr{N}_3}(F_0))} m_k(w)$.

11. The two worst cases and the special choices for the parameters

Combining the τ-Lemma 8.2 with Lemma 9.7, part 2 and Lemma 10.5 we have established our basis for the analysis of algorithm $\mathscr{N}_3$. In this section we motivate and define the special parameter values k, $(h_i)_{0\leqslant i\leqslant k}$ and α needed for $d^3 = d^3(k,(h_i),\alpha)$.

First we introduce some further notions for the computation tree $\mathscr{T}_{\mathscr{N}_3}(F_0)$ used frequently in the sequel.

Definition 11.1. For $w \in \mathrm{nds}_1(\mathscr{T}_{\mathcal{N}_3}(F_0))$ we define the following predicates:

$$\textbf{“D”}(\boldsymbol{w}) :\Leftrightarrow \text{the branching is degenerated: } |I(w)| = 1;$$

$$\textbf{“A”}(\boldsymbol{w}) :\Leftrightarrow \neg\text{“D”}(w) \text{ and case (A) holds: } \rho(w) \geqslant 2;$$

$$\textbf{“AA”}(\boldsymbol{w}) :\Leftrightarrow \text{“A”}(w) \text{ and case (3a) of “branchingA” holds;}$$

$$\textbf{“AN”}(\boldsymbol{w}) :\Leftrightarrow \text{“A”}(w) \text{ and case (3b) of “branchingA” holds;}$$

$$\textbf{“B”}(\boldsymbol{w}) :\Leftrightarrow \neg\text{“D”}(w) \text{ and case (B) holds: } \rho(w) = 1 \wedge F_0(w)^{[2]} \backslash B(w) \neq \emptyset;$$

$$\textbf{“BN”}(\boldsymbol{w}) :\Leftrightarrow \text{“B”}(w) \text{ and case (3) of “branchingB” holds;}$$

$$\textbf{“BB}_0\textbf{”}(\boldsymbol{w}) :\Leftrightarrow \text{“B”}(w) \text{ and case (4a) of “branchingB” holds;}$$

$$\textbf{“BB}_1\textbf{”}(\boldsymbol{w}) :\Leftrightarrow \text{“B”}(w) \text{ and case (4b) of “branchingB” holds;}$$

$$\textbf{“C”}(\boldsymbol{w}) :\Leftrightarrow \neg\text{“D”}(w) \text{ and case (C) holds: } \rho(w) \leqslant 1 \wedge F_0(w)^{[2]} \backslash B(w) = \emptyset.$$

Let $\boldsymbol{\hat{\varphi}(w)}$ be defined by branching$(F_0(w)) = (\hat{\varphi}(w), F(w))$, and $\boldsymbol{I(w)} := I(\hat{\varphi}(w))$.

Let $\boldsymbol{l(w)}$ be the result of branchLitA$(F_0(w))$, branchLitB$(F_0(w))$ or branchLitC$(F_0(w))$ in case “A”(w), “B”(w) or “C”(w). And let

$$\boldsymbol{\#_0(w)} := \#^2_{l(w)}(F(w)), \qquad \boldsymbol{\#_1(w)} := \#^2_{\overline{l(w)}}(F(w)).$$

New clauses in branch $i \in I(w)$ are denoted by

$$\boldsymbol{N_i(w)} := N(\hat{\varphi}_i(w), F(w)),$$
$$\boldsymbol{N_i^r(w)} := N_r(\hat{\varphi}_i(w), F(w)).$$

For $i \in I(\hat{\varphi}(w))$ let $\boldsymbol{w^i}$ be the root of subtree $\mathscr{T}_{\mathcal{N}_3}(r(\hat{\varphi}_i(w) * F(w)))$.

Finally for any edge labeling d of $\mathscr{T}_{\mathcal{N}_3}(F_0)$, for $w \in \mathrm{nds}(\mathscr{T}_{\mathcal{N}_3}(F_0))$ and for $i \in I(w)$ we use $\boldsymbol{d_i(w)} := d_{w^i}(w)$.

Note that for $w \in \mathrm{nds}_1(\mathscr{T}_{\mathcal{N}_3}(F_0))$ with $\neg$“D”(w) and $\neg$“AA”(w) we have $I(w) = \{0, 1\}$, and that the test assignments are given by

$$\hat{\varphi}_i(w) = [\langle l(w) \to i \rangle]_{F(w)}$$

for $i \in I(w)$. Furthermore in cases “A”(w), “BN”(w) and “C”(w) we have $F(w) = F_0(w)$, while in general $F_0(w) \subseteq F(w)$ holds (hence $N(\hat{\varphi}_i(w), F_0(w)) \subseteq N_i(w)$), and $F(w)^{[2]} = F_0(w)^{[2]}$.

In order to estimate $d^3(w)$ we have to give lower bounds on $\Delta_i n(w)$ and $v_i(w)$, and upper bounds on $\delta_i z_j(w)$ for $0 \leqslant j \leqslant k$. The next lemma gives the structural facts for $\Delta_i n$ and v_i:

Lemma 11.1. *The basic facts for the estimation of Δn and v*
For $w \in \mathrm{nds}_1(\mathcal{T}_{\mathcal{N}_3}(F_0))$ and $i \in I(w)$ we have
1. $\Delta_i n(w) \geqslant n(\hat{\varphi}_i(w)) \geqslant 1$;
2. (a) $v_i(w) = |N_i^r(w)|$;
   (b) $\Delta_i n(w) = n(\hat{\varphi}_i(w)) \Rightarrow N_i(w) \subseteq N_i^r(w)$;
   (c) $\neg$"D"$(w) \Rightarrow N_i(w), N_i^r(w) \neq \emptyset$.

Proof. For part 2(b) use property 3(d) of Lemma 5.1, and for part 2(c) use Lemma 6.1, part 3. □

The lower bounds for Δn and v we need in this section are given in the next lemma.

Lemma 11.2. *Lower bounds for Δn and v in the cases "AN", "B(N)" and "C"*
Consider $w \in \mathrm{nds}(\mathcal{T}_{\mathcal{N}_3}(F_0))$ with "AN"$(w) \vee$ "B"$(w) \vee$ "C"(w).
1. (a) $n(\hat{\varphi}_i(w)) \geqslant \#_i(w) + 1$ *for* $i \in I(w)$,
   (b) $\#_0(w) + \#_1(w) = \rho(w)$,
   (c) $\neg$"C"$(w) \Rightarrow \#_0(w) \geqslant 1$.
2. *In case* "AN"(w):
   (a) $\Delta_1 n(w) = 1 \Rightarrow |N_0(w)| \geqslant 2$,
   (b) $\Delta_0 n(w) = \Delta_1 n(w) = 2 \Rightarrow v_0(w), v_1(w) \geqslant 2$.
3. *In case* "BN"(w): $|N_0(w)|, |N_1(w)| \geqslant 3$.

Proof. Part 1 follows from the special choices for $l(w)$ (see Sections 6.3.1 and 6.4.1).
For part 2 see case (3a) in "branchingA" (and for 2(b) also use part 1 and Lemma 11.1, parts 2(a) and 2(b)).
And for part 3 see case (3) in "branchingB". □

For the cases "AN," "B" and "C" we improve the upper bounds for δz_k from Lemma 9.5 as follows.

Lemma 11.3. *Consider $F \in 3\text{-}\mathcal{CLS}$, $l \in \mathcal{LIT}$ and $j \in \mathbb{N}_0 \cup \{+\infty\}$. We define the partial assignment φ as $\langle l \to 0 \rangle$ extended by the immediately following 1-clause-eliminations*:

$$\varphi := \langle l \to 0 \rangle \cup \langle x \to 1 : \{l, x\} \in F^{[2]} \rangle.$$

We assume that φ is indeed a partial assignment ($\varphi \in \mathcal{PASS}$), i.e., F does not contain clauses $\{l, x\}$ and $\{l, \bar{x}\}$ at the same time, which is fulfilled for $F \in 3\text{-}\mathcal{CLS}_r$ (see Property 4(c) *from Lemma* 5.1).

By Lemma 9.5 *we can estimate the number of eliminated z_j-clauses by*

$$\delta z_j(F, \varphi * F) \leqslant \min(j, \rho(F)) \cdot n(\varphi) = \min(j, \rho(F)) \cdot (\#_l^2(F) + 1).$$

However the special structure of φ we did not use yet. For example, if we have $\varphi = \langle l \to 0, x \to 1 \rangle$ for $\{l, x\} \in F^{[2]}$, where all 2-clauses of F are variable-disjoint ($\rho(F) = 1$), then $\delta z_j(F, \varphi * F) = 1$ (for $j \geqslant 1$) holds, while the above bound gives only $\delta z_j(F, \varphi * F) \leqslant 2$.

The reason for this difference is that because of $\#^2_{\bar{l}}(F)=0$ *all* 2*-clauses eliminated by* $\langle l \to 0\rangle$ *are already covered by the* 2*-clauses eliminated by* $\langle x \to 1 : \{l,x\} \in F^{[2]}\rangle$.

For the general case we easily obtain:

$$\text{(i)} \quad \delta z_j(F, \varphi * F) \leqslant \min(j, \#^2_{\bar{l}}(F)) + \min(j, \rho(F)) \cdot \#^2_{l}(F).$$

If $\varphi * F$ *is anew transformed into* $F' \in 3\text{-}\mathscr{CLS}$ *such that no new variables are introduced, and every eliminated* 2*-clause contains also an eliminated variable* (*as for* $\varphi * F \to r(\varphi * F)$), *then by the triangle inequality from Lemmas* 9.3 *and* 9.5, *part* 2(b) *we finally get*

$$\text{(ii)} \quad \delta z_j(F, F') \leqslant \min(j, \#^2_{\bar{l}}(F)) + \min(j, \rho(F)) \cdot (\Delta n(F, F') - 1).$$

Applied to $\mathscr{T}_{\mathscr{N}_3}(F_0)$ we obtain:

Lemma 11.4. *For a node* $w \in \text{nds}(\mathscr{T}_{\mathscr{N}_3}(F_0))$ *with* "AN"$(w) \vee$ "B"$(w) \vee$ "C"(w), *for* $i \in I(w)$ *and for* $j \in \mathbb{N}_0 \cup \{+\infty\}$ *we have*

$$\delta_i z_j(w) \leqslant \min(j,\, \#_{\bar{i}}(w)) + (\Delta_i n(w) - 1) \cdot \min(j,\, \rho(w)).$$

Now we consider the two worst cases of $\mathscr{N}_3$ mentioned in Section 9.3, leading us to the numerical definitions of the parameters k, h and α.

11.1. The worst case for "AN"

Consider $w \in \text{nds}(\mathscr{T}_{\mathscr{N}_3}(F_0))$ with "AN"(w) and $\rho(w) = k+1$ fulfilling

$$\#_0(w) = \rho(w) = k+1 \ (\Rightarrow \#_1(w) = 0),$$

$$\Delta_0 n(w) = \#_0(w) + 1 = k+2,$$

$$\Delta_1 n(w) = \#_1(w) + 1 = 1.$$

Lemma 11.4 yields:

$$\delta_0 z_j(w) \leqslant (k+1) \cdot j \quad \text{for } j \in \{0, \ldots, k\},$$

$$\delta_1 z_k(w) \leqslant k, \quad \delta_1 z_\infty(w) \leqslant k+1.$$

Furthermore, applying Lemma 11.2 part 2 and Lemma 11.1 part 2:

$$v_0(w) \geqslant 2, \quad v_1(w) \geqslant 1.$$

Thus assuming

$$\textbf{(Ah):} \quad \forall_{j \in \{0,\ldots,k\}} [\, h_j \leqslant 2 + (k+1) \cdot (k-j) \,]$$

we obtain the following upper bounds for $\zeta_i(w)$ (see Lemma 9.2, part 2):

$$\zeta_0(w) \leqslant \max\left(\min(\delta_0 z_\infty(w), \max_{j\in\{0,\dots,k\}}(h_j + \delta_0 z_j(w))) - v_0(w),\ \delta_0 z_k(w)\right)$$

$$\leqslant \max\left(\max_{j\in\{0,\dots,k\}}(h_j + j\cdot(k+1)) - 2,\ k\cdot(k+1)\right) = k\cdot(k+1).$$

$$\zeta_1(w) \leqslant \max\left(\min(\delta_1 z_\infty(w), \max_{j\in\{0,\dots,k\}}(h_j + \delta_1 z_j(w))) - v_1(w),\ \delta_1 z_k(w)\right)$$

$$\leqslant \max(\delta_1 z_\infty(w) - v_1(w),\ \delta_1 z_k(w)) \leqslant k.$$

We conclude:

$$d^3(w) \geqslant (k + 2 - (k+1)\cdot k\cdot\alpha,\ 1 - k\cdot\alpha). \tag{1}$$

11.2. The worst case for "BN"

Consider $w \in \mathrm{nds}(\mathscr{T}_{\mathcal{N}_3}(F_0))$ with "BN"(w), $\Delta_0 n(w) = 2$ and $\Delta_1 n(w) = 1$. We assume that lower bounds for the number of new 2-clauses are given: $v_i(w) \geqslant q_i \in \mathbb{N}$ for $i \in I(w)$.

Lemma 11.4 gives

$$\delta_0 z_\infty(w),\ \delta_1 z_\infty(w) \leqslant 1 \quad (\text{in fact } = 1)$$

and hence, assuming $\max(q_0, q_1) \leqslant h_1$, Lemma 9.2, part 1 gives for $i \in \{0, 1\}$:

$$\zeta_i(w) \leqslant 1 - q_i.$$

Thus:

$$d^3(w) \geqslant (2 + (q_0 - 1)\cdot\alpha,\ 1 + (q_1 - 1)\cdot\alpha). \tag{2}$$

11.3. The basis of the bound on the number of leaves of the computation tree, depending on the approximation parameter k and the number of new clauses in case "BN"

We define $\boldsymbol{\tau_{k,q_0,q_1}}$ as the minimal τ-value reachable w.r.t. the branching tuples (1) and (2):

Definition 11.2. For $k \in \mathbb{N}_0$ and $q_0, q_1 \in \mathbb{N}$ we define:

$$\tau_{k,q_0,q_1} := \min_{0\leqslant\alpha<1/k} \max\Big(\tau(k + 2 - (k+1)\cdot k\cdot\alpha,\ 1 - k\cdot\alpha),$$
$$\tau(2 + (q_0 - 1)\cdot\alpha,\ 1 + (q_1 - 1)\cdot\alpha)\Big).$$

Trivially we have $\tau_{0,q_0,q_1} = \tau(2, 1) = 1.618$.

For $k \geqslant 1$ the function $\tau(k+2-(k+1)\cdot k\cdot\alpha,\ 1-k\cdot\alpha)$ of $0 \leqslant \alpha < 1/k$ is strictly decreasing, starting at $\tau(k+2,1)$, while $\tau(2+(q_0-1)\cdot\alpha,\ 1+(q_1-1)\cdot\alpha)$ is increasing (for $q_0, q_1 \geqslant 1$), starting at $\tau(2,1) < \tau(k+2,1)$, and hence there is exactly one $0 < \alpha < 1/k$ where the two functions coincide:

Definition 11.3. For $k, q_0, q_1 \in \mathbb{N}$ let $\boldsymbol{\alpha_{k,q_0,q_1}}$ be the solution $\alpha \in \mathbb{R}$, $0 < \alpha < 1/k$ of

$$\tau(k+2-(k+1)\cdot k\cdot\alpha,\ 1-k\cdot\alpha), = \ \tau(2+(q_0-1)\cdot\alpha,\ 1+(q_1-1)\cdot\alpha).$$

Now:

$$\begin{aligned}\tau_{k,q_0,q_1} &= \tau(k+2-(k+1)\cdot k\cdot\alpha_{k,q_0,q_1},\ 1-k\cdot\alpha_{k,q_0,q_1})\\ &= \tau(2+(q_0-1)\cdot\alpha_{k,q_0,q_1},\ 1+(q_1-1)\cdot\alpha_{k,q_0,q_1}).\end{aligned}$$

Lemma 11.2 part 3 motivates to choose $q_0 = q_1 = 3$. Using $\min_{k\in\mathbb{N}_0} \tau_{k,3,3} = \tau_{2,3,3}$[38] we are now ready to define the numerical values of the parameters k, h and α:

Definition 11.4. The numerical values of the parameters:

$$\boldsymbol{k} := 2, \quad \boldsymbol{h_0} := 0, \quad \boldsymbol{h_1} := 5, \quad \boldsymbol{h_2} := 2, \quad \boldsymbol{h_i} := 0 \quad \text{for } i \geqslant 3, \quad \boldsymbol{\alpha} := \alpha_{2,3,3}.$$

The values of h_1, h_2 are maximally chosen s.t. condition (Ah) is fulfilled. Numerical calculation yield $\alpha_{2,3,3} = 0.123931..$ and $\tau_{2,3,3} = 1.504432..$

12. Local balancings of distances

In this section we present our method of handling the situation where the τ-value of one node is "bad" but in the neighbourhood there are nodes with "good" τ-value which can compensate the "bad" value.

A possibility to handle this situation is to build composite branching tuples in the way of Lemma 8.4. We prefer to shift "surplus" since it is easier to handle[39] and allows to share surplus' between different nodes.

Definition 12.1. For a branching tuple $t \in \mathscr{BT}$ and an envisaged τ-value $\beta > 1$ (in case t has breadth one also $\beta = 1$ is possible) we define the *surplus* $\boldsymbol{\sigma_\beta(t)}$ of t w.r.t. β as

$$\boldsymbol{\sigma_\beta(t)} := \sup\{\,\lambda \in \mathbb{R} : t - \lambda \in \mathscr{BT} \wedge \tau(t-\lambda) \leqslant \beta\,\},$$

[38] We will not *prove* the optimality of our choice of the parameters since it does not matter for the bound.

[39] For a "bad" node w with a "good" successor w' we are not able to fix the type of w' (for example, if we know $\rho(w') \geqslant 2$, then nevertheless we can have "AN"(w), "AA"(w) or "D"(w)(!)) and thus, when using composite branching tuples we had to consider all compositions of the branching tuple at w with the branching tuples of all the different cases at w', since replacing in a composite branching tuple one of the component tuples by a tuple with a better τ-value can nevertheless impair the total τ-value(!) – however when shifting surplus' we only have to guarantee that in all cases for w' the needed surplus is available, which can be done by looking at those cases alone (no combination with the node which needs "support" is needed), and thus can be used also for different types of nodes w.

where $t - \lambda$ for $t = (t_1, \dots, t_s)$ is $t = (t_1 - \lambda, \dots, t_s - \lambda)$.

Some remarks: (the breadth of t we denote by $|t|$)

1. Using $\min(t) := \min_{1 \leqslant i \leqslant s} t_i$ for $t = (t_1, \dots, t_s)$ we have $t - \lambda \in \mathscr{BT} \Leftrightarrow \lambda < \min(t)$ (in case of $s = 1$ also $\lambda = \min(t)$ is possible).
2. If $|t| > 1$ then $\sigma_\beta(t) < \min(t)$ holds, while otherwise we have $\sigma_\beta(t) = \min(t) = t$.
3. For $|t| > 1$ we have
$$\lim_{\lambda \nearrow \min(t)} \tau(t - \lambda) = +\infty \quad \text{and} \quad \lim_{\lambda \to -\infty} \tau(t - \lambda) = 1$$
and thus indeed $\sigma_\beta(t)$ is well defined, and since τ is continuous we could define σ by using the "max" instead of the "sup."
4. (a) $\sigma_\beta(t) \geqslant 0 \Leftrightarrow \tau(t) \leqslant \beta$,
   (b) $\tau(t - \sigma_\beta(t)) = \beta$ for $|t| > 1$.

Lemma 12.1. *Consider a tree T with distance function d, nodes $w_0, w \in \mathrm{nds}_1(T)$ with $w \in \mathrm{ds}_T(w_0)$ and a real number $s \in \mathbb{R}$.*

We define the edge labeling $d[w_0, w, s]$ as equal to d except for the edges (w_0, w) and (w, v_i), where $\mathrm{ds}_T(w) = \{v_1, \dots, v_m\}$:

$$d[w_0, w, s](w_0, w) := d(w_0, w) + s,$$

$$d[w_0, w, s](w, v_i) \ := d(w, v_i) - s.$$

$d[w_0, w, s]$ has the same maximal sum as d:

$$\sum d[w_0, w, s](\mathrm{root}\,(T)) = \sum d(\mathrm{root}\,(T)).$$

If $-d(w_0, w) < s \leqslant \sigma_\beta(d(w))$ for $\beta > 1$ holds, then $d[w_0, w, s]$ is a distance function for T with

$$\tau(d[w_0, w, s](w)) \leqslant \beta.$$

"Bad" cases with $\tau(d^3(w)) > \tau_{2,3,3}$ for $w \in \mathrm{nds}(\mathscr{T}_{\mathscr{N}_3}(F_0))$ occur, at least according to our analysis, in the following three situations:

(1) In case "C"(w) the branching at w is a poor one, but for compensation we have "br-autarkness" (compare Lemma 6.1, part 4), guaranteeing that the immediate predecessor w_0 of w has property "D"(w_0), and so we can replace d^3 by $d^3[w_0, w, \sigma_{\tau_{2,3,3}}(w)]$. (See Lemma 13.4.)
If w is the root of $\mathscr{T}_{\mathscr{N}_3}(F_0)$, then we introduce a "virtual" predecessor ω_0^*, increasing the maximal sum $\sum d^3(\omega_0^*)$ only by a constant. See Lemma 13.14.

(2) Under certain circumstances in case "BB$_1$"(w) one new 2-clause is "missing," but then we know that for one direct successor w' of w there is a variable occurring in both signs in the 2-clauses, which assures a surplus at w' of at least $1 \cdot \alpha$ (compensating the "missing" clause). So we replace d^3 by $d^3[w, w', \alpha]$.

(3) And also under certain circumstances in case "AN"(w) we need a surplus from one direct successor.

Some difficulties arise with (2) and (3): Although we know that we have a favourable situation at w', which “should” yield a surplus of α, the algorithm must deviate from what we expect in the following three cases:
(i) the branching rule behaves different (for example “AA”(w) may happen); or
(ii) the branching at w' is in fact degenerated, i.e., “D”(w') holds;
(iii) or w' even is a leaf: $w' \in \mathrm{lvs}(\mathscr{T}_{\mathcal{N}_3}(F_0))$.
(i) is handled by showing that in all possible cases the surplus α is guaranteed. See Lemma 13.9.

In case (ii) we have to be careful, since the surplus at w' is possibly also used for (1). Fortunately we need only the amount α in (2) and (3), and so it suffices to show $\alpha - \sigma_{\tau_{2,3,3}}(w) \leqslant d^3(w_0)$ for the situation in (1). See Lemma 13.4.

And to handle case (iii) we just increase the maximal sum $\sum d^3$ by α. See Lemma 13.14.

13. The estimation of $\tau_{\max}(d^3, \mathscr{T}_{\mathcal{N}_3}(F_0))$

In this section we finally conclude the proof of Theorem 2. In order to do so, we have to show that for all branching tuples t in $\mathscr{T}_{\mathcal{N}_3}(F_0)$, induced by the distance function d^3 (in some cases modified according to the foregoing section), the estimation $\tau(t) \leqslant \tau_{2,3,3}$ holds. The values $\tau(t)$ are computed by numerical calculations.

The reader who wants to verify $\tau(t) \leqslant \tau_{2,3,3}$ can do this by simply checking that

$$\sum_{i=1}^{m} \tau_{2,3,3}^{-t_i} \leqslant 1$$

holds, where $t = (t_1, \ldots, t_m)$.

We start by a general lower bound for d^3, easily obtained from the general δz_k-bound of Lemma 9.6 when using the special values for h_0, h_1, h_2:

Lemma 13.1. *Consider a node $w \in \mathrm{nds}_1(\mathscr{T}_{\mathcal{N}_3}(F_0))$ and $i \in I(w)$. Using the abbreviations $\Delta n := \Delta_i n(w)$ and $v := v_i(w)$ we have the following general bounds*:
1. (a) $\zeta_i(w) \leqslant 2 + \Delta n + \max(3, \Delta n) - \min(v, 2)$,
   (b) $\neg$“D”$(w) \Rightarrow \zeta_i(w) \leqslant 1 + \Delta n + \max(3, \Delta n)$.
2. (a) $d_i^3(w) \geqslant \Delta n \cdot (1 - 2\alpha) - \alpha \cdot (\max(5 - \Delta n, 2) - \min(v, 2))$,
   (b) $\neg$“D”$(w) \Rightarrow d_i^3(w) \geqslant \Delta n \cdot (1 - 2\alpha) - \alpha \cdot \max(4 - \Delta n, 1)$.

Before obtaining in the next lemma (as an easy application) a criterion for nodes with a surplus of at least α, we introduce the following convention:

Definition 13.1. For tuples $t = (t_1, \ldots, t_p)$ and $t' = (t'_1, \ldots, t'_q)$ the relation $\boldsymbol{t} \geqslant \boldsymbol{t'}$ is fulfilled iff $p = q$ holds and for all $1 \leqslant i \leqslant p$ we have $t_i \geqslant t'_i$.

Lemma 13.2. *For a node $w \in \mathrm{nds}(\mathscr{T}_{\mathcal{N}_3}(F_0))$ with $|I(w)| = 2$, $\Delta n(w) \geqslant (2, 2)$ and $\sum_{i \in I(w)} \Delta_i n(w) \geqslant 6$ we have $\sigma_{\tau_{2,3,3}}(d^3(w)) \geqslant \alpha$.*

Proof. Apply Lemma 13.1, part 2(b) to the cases $\Delta n(w) \geqslant (3,3)$ and $\Delta n(w) \geqslant (2,4)$, getting $d^3(w) \geqslant (3-7\alpha) \cdot (1,1)$ resp. $d^3(w) \geqslant (2-6\alpha, 4-9\alpha)$. Numerical calculations yield $\tau((3-8\alpha)\cdot(1,1)) = 1.41..$ and $\tau(2-7\alpha, 4-10\alpha) = 1.46..$ $\square$

13.1. The case "D" of degeneration, and case "C"

Immediately from Lemma 13.1 we get the following estimation of the surplus at a node belonging to case "D" (using the numerical value for α):

Lemma 13.3. *Consider* $w \in \mathrm{nds}(\mathcal{T}_{\mathcal{N}_3}(F_0))$ *with* "D"(w).
1. $\sigma_{\tau_{2,3,3}}(d^3(w)) \geqslant \alpha$.
2. *If* $\Delta n(w) \geqslant 2$ *holds, then we have* $\sigma_{\tau_{2,3,3}}(d^3(w)) \geqslant 2 - 7\alpha$.

Lemma 13.4. *For a node* $w \in \mathrm{nds}(\mathcal{T}_{\mathcal{N}_3}(F_0))$ *with* "C"(w), *which is not the root of* $\mathcal{T}_{\mathcal{N}_3}(F_0)$ *and thus has a direct predecessor* $w_0 \in \mathcal{T}_{\mathcal{N}_3}(F_0)$, *we have the following*:
1. $\sigma_{\tau_{2,3,3}}(d^3(w_0)) \geqslant 2 - 7\alpha$;
2. $\tau(d^3[w_0, w, -(2-8\alpha)](w)) \leqslant \tau(3-9\alpha, 3-9\alpha) = 1.44\ldots < \tau_{2,3,3}$.

Proof. By Lemma 11.1, part 2(c) we know that "D"(w_0) holds.

Furthermore, Lemma 6.1, part 4 gives us $\Delta n(w_0) \geqslant 2$, since the additional assumption in Lemma 6.1, part 4, called $(*)$ here, is fulfilled as the following argumentation shows:

> Assume $\Delta n(w_0) = 1$. Thus $F_0(w_0) \neq F(w_0)$ must hold (otherwise we could use $E := \emptyset$ for $(*)$).
>
> We conclude that $\mathcal{N}_3$ on input $F_0(w_0)$ must have passed instructions (4a) or (4b) in "branchingB", because these are the only places in $\mathcal{N}_3$ where the input is changed.
>
> Furthermore $\hat{\varphi}'_i = \hat{\varphi}(w_0)$ from Lemma 6.1, part 4 corresponds to variable $\hat{\varphi}_1$ from "branchingB", i.e., $\hat{\varphi}(w_0) = \langle l \to 1 \rangle$ for $l := \mathrm{branchLitB}(F_0(w_0))$, since $n(\hat{\varphi}_0) \geqslant 2$ holds (there is a 2-clause $C \in F_0^{[2]}(w_0)$ with $l \in C$, and thus the 1-clause-closure, performed by "eval" in instruction (2) of "branchingB", adds (at least) one additional variable to $\langle l \to 0 \rangle$).
>
> Now if "branchingB" has passed (4a) we can set $E := B_{\bar{l}}^{[3]}(F_0(w_0))$, and in case (4b) we set $E := \emptyset$.

The previous Lemma 13.3 now gives $\sigma_{\tau_{2,3,3}}(d^3(w_0)) \geqslant 2 - 7\alpha$. From that surplus we can use $2 - 8\alpha$, since 1α must be saved for a (possible) "negative surplus" at the predecessor of w_0.

Because of $\rho(w) \leqslant 1$ we immediately get from Lemma 9.2, part 1 and Lemma 9.6 the estimation

$$d^3(w) \geqslant (1-\alpha) \cdot (\Delta_0 n(w), \Delta_1 n(w)) \geqslant (1-\alpha, 1-\alpha). \qquad \square$$

13.2. The case "AA"

Definition 13.2. For $w \in \mathrm{nds}(\mathcal{T}_{\mathcal{N}_3}(F_0))$ with "AA"(w) let $\boldsymbol{\varphi(w)}$ be the parameter φ of procedure "GenAut" (see Section 6.3.2) at the last call, so that there are literals a, b

with $N(\varphi(w), F_0(w)) = \{\{a,b\}\}$ and

$$\hat{\varphi}_0(w) = [\langle a \to 0, b \to 0\rangle]_{F_0(w)},$$

$$\hat{\varphi}_1(w) = [\langle a \to 0, b \to 1\rangle \cup (w)]_{F_0(w)},\ \ \hat{\varphi}_2(w) = [\langle a \to 1\rangle \cup \varphi(w)]_{F_0(w)}.$$

(In case $I(w)| = 2$ one of these assignments is in fact not present in $\hat{\varphi}(w)$.)

The general estimation of δz_k from Lemma 9.6 we have to refine for the "Generalized Autarkness branching" (remember Section 4.3) in those cases, where applying the 1-clause-closure (Definition 5.2) has "no additional effect." In fact we prove the following general result:

Consider a "test-system" $(\psi_1, \ldots, \psi_p)$ of partial assignments created by binary branching (corresponding to a binary tree where each inner node is labeled by the "branching variable").

Extend the ψ to ψ_i' such that the additional variables are taken from a fixed set V of variables (apart from that the extensions are arbitrary).

If now for each ψ_i' the 1-clause-closure (with respect to a certain clause-set F) does not add new variables, and also the empty clause is not created, then each ψ_i' only effects 2-clauses which contain some variables from V.

Lemma 13.5. *Consider $F \in 3\text{-}\mathscr{CLS}$ and $V \subseteq \mathscr{VA}$.*

Consider a "test system" $(\psi_1, \ldots, \psi_p)$ of partial assignments generated by the following rule, starting with the trivial test system $(\emptyset)$:

- *If $(\psi_1, \ldots, \psi_m)$ already has been generated, choose $1 \leqslant i \leqslant m$ and generate*

$$(\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots, \psi_m, \psi_i \cup \langle v \to 0\rangle, \psi_i \cup \langle v \to 1\rangle)$$

for a variable $v \notin \mathrm{Var}(\psi_i)$.

Consider (finally) any system $(\psi_1', \ldots, \psi_p')$ of partial assignments fulfilling

$$\forall i \in \{1, \ldots, p\} : \psi_i \subseteq \psi_i' \wedge \mathrm{Var}(\psi_i') \backslash \mathrm{Var}(\psi_i) \subseteq V.$$

Assume

$$(*) \quad \forall i \in \{1, \ldots, p\} : (\psi_i' * F)^{[0,1]} = \emptyset.$$

Then for all $i \in \{1, \ldots, p\}$ we have:

(i) $\{C \in F^{[2]} : \mathrm{Var}(\psi_i') \cap \mathrm{Var}(C) \neq \emptyset\} \subseteq \{C \in F^{[2]} : \mathrm{Var}(C) \cap V \neq \emptyset\}$.

(ii) *If especially $V = \mathrm{Var}(\varphi)$ for some $\varphi \in \mathscr{PASS}$ holds: $\delta z_k(F, \psi_i' * F) \leqslant \delta z_k(F, \varphi * F)$.*

Proof. Part (ii) follows immediately from part (i) by Lemma 9.4, part 1.

Assume there is a 2-clause $C \in F^{[2]}$ with $\mathrm{Var}(\psi_i') \cap \mathrm{Var}(C) \neq \emptyset$ and $\mathrm{Var}(C) \cap V = \emptyset$. It follows $\mathrm{Var}(\psi_i) \cap \mathrm{Var}(C) \neq \emptyset$.

Let $C = \{l, x\}$ with $\mathrm{Var}(l) \in \mathrm{Var}(\psi_i)$.

Due to the construction of $(\psi_1,\dots,\psi_p)$ there is $\psi_{i'}$ with $\psi_{i'}(l)=0$. Because of assumption $(*)$ (concerning 1-clauses) we have $\psi'_{i'}(x)=1$, and thus also $\psi_{i'}(x)=1$ holds (using $\mathrm{Var}(x)\notin V$).

Now by the same argumentation (but with x instead of l) there must also be $\psi_{i''}$ fulfilling $\psi_{i''}(x)=0$ and $\psi_{i''}(l)=1$.

Due to the construction of the test-system $(\psi_1,\dots,\psi_p)$ the following is true:

> If there are ψ_{j_1} and ψ_{j_2} with $\psi_{j_1}(a)=0$, $\psi_{j_1}(b)=1$ and $\psi_2(a)=1$, $\psi_2(b)=0$ for any literals a,b, $a\neq\overline{b}$, then there must be ψ_{j_3} with $\psi_{j_3}(a)=\psi_{j_3}(b)=0$.

Hence we get a contradiction to $(*)$ (concerning the 0-clause). □

Lemma 13.6. *For a node $w\in\mathrm{nds}(\mathscr{T}_{\mathcal{N}_3}(F_0))$ with* "AA"(w) *we have $\sigma_{\tau_{2,3,3}}(d^3(w))\geqslant\alpha$.*

Proof. To start with, first note that due to instruction (3) in "branchingA" we have $n(\varphi(w))\geqslant 2$.

Let $N(\varphi(w),F_0(w))=\{\{a,b\}\}$. There is $\{a,b,c\}\in F_0(w)^{[3]}$ with $\varphi(c)=0$ and

$$[\langle a\to 0,b\to 0\rangle]_{F_0(w)}\supseteq\langle a\to 0,b\to 0,c\to 1\rangle.$$

Thus in case $|I(w)|=2$ we know $\Delta n(w)\geqslant(3,3)$ and therefore we can apply Lemma 13.2, yielding the assertion.

For the remainder of the proof assume $|I(w)|=3$ (and so $I(w)=\{0,1,2\}$).

With respect to Δn and v we have the following estimations (see instructions (2) and (3b) of "GenAut", and apply Lemma 11.1, part 2(b)):

$$\Delta_0 n(w)\geqslant 3, \tag{1}$$

$$\Delta_1 n(w)\geqslant 2+n(\varphi(w))\geqslant 4, \tag{2}$$

$$\Delta_2 n(w)\geqslant 1+n(\varphi(w))\geqslant 3, \tag{3}$$

$$\Delta_0 n(w)=3\wedge n(\varphi(w))=2\Rightarrow v_0(w)\geqslant 2, \tag{4}$$

$$\Delta_1 n(w)=2+n(\varphi(w))\Rightarrow v_1(w)\geqslant 2, \tag{5}$$

$$\Delta_2 n(w)=1+n(\varphi(w))\Rightarrow v_2(w)\geqslant 2. \tag{6}$$

In case $n(\varphi(w))\geqslant 3$ Lemma 13.1, parts 2(a) and 2(b), and (5), (6) give

$$\begin{aligned}\tau(d^3(w)-\alpha)\leqslant\tau\big((&3\cdot(1-2\alpha)-\alpha\cdot(2-1),5\cdot(1-2\alpha)-\alpha\cdot(2-2),\\ &4\cdot(1-2\alpha)-\alpha\cdot(2-2))-\alpha\big)=1.48..<\tau_{2,3,3}.\end{aligned}$$

The case $n(\varphi(w))=2$ remains.

In this case procedure "GenAut" cannot have entered the "loop" in (3b) but must have executed (3a) immediately.

Furthermore

$$\varphi(w)=\langle l\to 0,x\to 1\rangle$$

holds for some 2-clause $\{l,x\} \in F_0(w)^{[2]}$ with $\#^2_l(w) = \#^2_{\bar{l}}(w) = 1$ and $\rho(w) = 2$, due to instruction (3) in "branchingA" and Lemma 11.2, parts 1(a) and 1(b) (branchLitA$(F_0(w)) \in \{l, \bar{l}\}$).

Because of $\rho(w) = 2$, Lemma 9.2, part 1 yields for $i \in I(w)$

$$\zeta_i(w) = \delta_i z_k(w) - \min(v_i(w), 2). \tag{7}$$

Sub-case I: $\Delta n(w) = (3,4,3)$.

Here we can apply Lemma 13.5, part (ii) (and Lemma 11.1, part 2(a) as well as Lemma 9.3, part 2) and get

$$\delta_i z_k(w) \leqslant \delta z_k(F_0(w), \varphi(w) * F_0(w)),$$

while by Lemma 11.3 we obtain

$$\delta z_k(F_0(w), \varphi(w) * F_0(w)) \leqslant 1 + 2 \cdot 1 = 3.$$

Altogether we have (using (4)–(6)):

$$\begin{aligned}\tau(d^3(w) - \alpha) &\leqslant \tau((3 - \alpha \cdot (3-2), 4 - \alpha \cdot (3-2), 3 - \alpha \cdot (3-2)) - \alpha)\\ &= 1.43.. < \tau_{2,3,3}.\end{aligned}$$

For the rest of the proof the estimation $\delta_i z_k(w) \leqslant 2 \cdot \Delta_i n(w)$ from Lemma 9.6 suffices for (7).

Sub-case II: $\Delta n(w) \geqslant (3,5,3)$.

$$\begin{aligned}\tau(d^3(w) - \alpha) &\leqslant \tau((3 - \alpha \cdot (6-2), 5 - \alpha \cdot (10-1), 3 - \alpha \cdot (6-2)) - \alpha)\\ &= 1.48.. < \tau_{2,3,3}.\end{aligned}$$

The (better) sub-cases $\Delta n(w) \geqslant (3,4,4)$ and $\Delta n(w) \geqslant (4,4,3)$ are handled analogously. □

Before turning to case "AN" we complete in the next subsection our list of cases which yield a surplus of at least α.

13.3. *Sufficient conditions for a surplus of at least α*

The next lemma combines the estimations from Lemmas 9.2 and 11.4 (omitting the straight-forward proofs).

Lemma 13.7. *For $w \in \mathrm{nds}(\mathcal{T}_{\mathcal{N}_3}(F_0))$ with "AN"(w) and $i \in I(w) = \{0,1\}$ we have*
1. $\rho(w) = 2 \Rightarrow \zeta_i(w) \leqslant \#_{\bar{i}}(w) + 2\Delta_i n(w) - 2 - \min(v_i(w), 2)$,
2. $\rho(w) = 3 \wedge \Delta_i n(w) \geqslant 2 \Rightarrow \zeta_i(w) \leqslant \#_{\bar{i}}(w) + 3\Delta_i n(w) - 4$.

Lemma 13.8. *For $w \in \mathrm{nds}(\mathcal{T}_{\mathcal{N}_3}(F_0))$ with "AN"(w) and $\Delta n(w) \geqslant (2,2)$ we have $\sigma_{\tau_{2,3,3}}(d^3(w)) \geqslant \alpha$.*

Proof. Because of Lemma 13.2 we assume $\Delta_0 n(w) + \Delta_1 n(w) \leqslant 5$. Lemma 11.2 part 1 gives $\#_i(w) \leqslant \Delta_i n - 1$ for $i \in I(w)$.

Case I: $\Delta n(w) = (2,2)$.

By Lemma 11.2 part 2(b) we have $v_0(w), v_1(w) \geqslant 2$. Thus Lemma 13.7 part 1 gives

$$\tau(d^3(w) - \alpha) \leqslant \tau((2,2) - 2\alpha) = 1.48.. < \tau_{2,3,3}.$$

Case II: $\Delta_0 n(w) + \Delta_1 n(w) = 5$.

By Lemma 13.7 parts 1 and 2 we get $\zeta_i(w) \leqslant \#_{\bar{i}}(w) + 3\Delta_i n(w) - 4$. Altogether:

$$\tau(d^3(w) - \alpha) \leqslant \tau(2 - 5\alpha, 3 - 7\alpha) = 1.49.. < \tau_{2,3,3}. \qquad \square$$

Using Lemmas 13.3, 13.6, 13.8, and the selection condition for the branching literal in “branchLitA” (see Section 6.3.1) we immediately get:

Lemma 13.9. *For* $w \in \mathrm{nds}_1(\mathscr{T}_{\mathcal{N}_3}(F_0))$ *s.t. there is a variable v occurring maximally often and in every sign at least once, i.e., fulfilling*

$$\exists v \in \mathrm{Var}(w) : (\#_v^2 + \#_{\bar{v}}^2)(w) = \rho(w) \wedge \min(\#_v^2(w), \#_{\bar{v}}^2(w)) \geqslant 1$$

we can infer $\sigma_{\tau_{2,3,3}}(d^3(w)) \geqslant \alpha$.

13.4. The case “AN”

We complete our estimations for ζ_i in case “AN” as follows, again combining Lemma 9.2 and Lemma 11.4 (and again omitting the easy proofs).

Lemma 13.10. *For* $w \in \mathrm{nds}(\mathscr{T}_{\mathcal{N}_3}(F_0))$ *with* “AN”(w), $\rho(w) \geqslant 3$ *and* $\Delta_1 n(w) = 1$ *we have*
1. $\zeta_0(w) \leqslant 2\Delta_0 n(w) - \min(v_0(w), 2)$,
2. $\zeta_1(w) \leqslant \min(\rho(w), 6) - 1$.

Lemma 13.11. *Consider* $w \in \mathrm{nds}(\mathscr{T}_{\mathcal{N}_3}(F_0))$ *with* “AN”(w). *Assume* w^1 *is not a leaf of* $\mathscr{T}_{\mathcal{N}_3}(F_0)$. *Then*

$$\tau(d^3[w, w^1, \max(\min(\sigma_{\tau_{2,3,3}}(d^3(w^1)), \alpha), 0)](w)) < \tau_{2,3,3}$$

holds.

Proof (*Outline*). We consider two main cases: $\rho(w) = 2$ (case I) and $\rho(w) \geqslant 3$ (case II).

Since in case II “enough” variables disappear, it is the easier case and can be handled by means of Lemma 13.10.

If in case I not “enough” variables vanish, and not “enough” new 2-clauses are created in the branch $w \to w_1$, then we have to look more carefully at the special situation (in cases I.1, I.2, I.3):

We have two occurrences $\{l(w), a\}, \{l(w), b\} \in F_0(w)^{[2]}$ of the branching literal $l(w)$ in the 2-clauses of $F_0(w)$. If these two clauses are the only occurrences of $\mathrm{Var}(a)$ and

$\mathrm{Var}(b)$, then we can improve our (general) bound $\delta_0 z_k(w) \leqslant 4$ in branch $w \to w^0$ by $\delta_0 z_k(w) = 2$ (case I.1).

Otherwise (I.2, I.3) we can apply Lemma 13.9 to w^1 (since the one new 2-clause brings a complementary literal into play), getting a surplus $\sigma_{\tau_{2,3,3}}(d^3(w^1)) \geqslant \alpha$.

Proof. Due to Lemma 13.8 we assume $\Delta_1 n(w) = 1$, and thus by Lemmas 11.1 and 11.2 we have

$$\hat{\varphi}_1(w) = \langle l(w) \to 1 \rangle, \qquad \#_1(w) = 0, \quad N_1(w) \subseteq N_1^r(w)$$
$$\#_0(w) = \rho(w), \qquad \Delta_0 n(w) \geqslant \rho(w) + 1.$$

Case I: $\rho(w) = 2$. Lemma 13.7, part 1 and Lemma 11.2, part 2(a) give

$$\zeta_0(w) \leqslant 2\Delta_0 n(w) - 2 - \min(v_0(w), 2), \tag{1}$$

$$\zeta_1(w) \leqslant 2 - \min(v_1(w), 2) \leqslant 1, \tag{2}$$

$$d_0(w) \geqslant \Delta_0 n(w) \cdot (1 - 2\alpha) + (2 + \min(v_0(w), 2)) \cdot \alpha \tag{3}$$

$$\geqslant 3(1 - 2\alpha) + 4\alpha = 3 - 2\alpha, \tag{4}$$

$$d_1(w) \geqslant 1 - \alpha. \tag{5}$$

If $\Delta_0 n(w) \geqslant 4$, then (3) and (5) give $d^3(w) \geqslant (4 - 5\alpha, 1 - \alpha)$ and thus $\tau(d^3(w)) < \tau_{2,3,3}$.

Thus we assume $\Delta_0 n(w) = 3$ in the following (and hence in fact $v_0(w) \geqslant 2$ is the case).

If $v_1(w) \geqslant 2$, then by (2) and (4) we have

$$\tau(d^3(w)) \leqslant \tau(3 - 2\alpha, 1) = 1.494.. < \tau_{2,3,3}.$$

So let us assume $v_1(w) = 1$ for the rest of case I. Now we have to consider $F_0(w^1)$ in detail. To begin with just note (remember Lemma 5.1, part 3(d))

$$N_1(w) = N_1^r(w), \qquad F_0(w^1)^{[2]} = (\hat{\varphi}_1(w) * F_0(w))^{[2]}.$$

Using $l := l(w)$ there are 2-clauses $\{l, a\}, \{l, b\} \in F_0(w)$ with

$$\hat{\varphi}_0(w) = \langle l \to 0, a \to 1, b \to 1 \rangle.$$

Lemma 5.1, part 4(g) yields

$$\#_{\bar{l}}(F_0(w)) = \#_{\bar{l}}^2(F_0(w)) = v_1(w) = 1.$$

Let $\{\bar{l}, x, y\} \in F(w_0)$ the $\bar{l}$-occurrence in $F_0(w)$. Lemma 5.1, part 4(f)i gives

$$\{l, \bar{x}\}, \{l, \bar{y}\} \in F_0(w)^{[2]}, \Rightarrow \{\bar{l}, x, y\} = \{\bar{l}, \bar{a}, \bar{b}\},$$

and thus

$$F_0(w^1)^{[2]} = (F_0(w)^{[2]} \setminus \{\{l, a\}, \{l, b\}\}) \cup \{\{\bar{a}, \bar{b}\}\},$$

from which we conclude $\rho(w^1) \leqslant 2$.

Case I.1: $(\#^2_a + \#^2_{\overline{a}})(F_0(w)) = (\#^2_b + \#^2_{\overline{b}})(F_0(w)) = 1$.
Now

$$F_0(w^0)^{[2]} \supseteq F_0(w)^{[2]} \setminus \{\{l,a\},\{l,b\}\}$$

holds. It follows $\delta_0 z_k(w) = 2$, yielding (recall Lemma 9.2)

$$\zeta_0(w) \leqslant \delta_0 z_k(w) - \min(v_0(w), 2) = 0 :$$

$$\tau(d^3(w)) \leqslant \tau(3, 1-\alpha) = 1.497.. < \tau_{2,3,3}.$$

Case I.2: $(\#^2_a + \#^2_{\overline{a}})(F_0(w)) = 2$.

Due to the selection condition in "branchLitA" we have $\#_{\overline{a}}(F_0(w)) = 0, \Rightarrow \#_a(F_0(w)) = 2$. Hence there is a 2-clause $\{a,u\} \in F_0(w)$ with $\mathrm{Var}(u) \neq \mathrm{Var}(l)$, and we know

$$\{a,u\}, \{\overline{a},\overline{b}\} \in F_0(w^1)$$

with $\rho(w^1) = 2$. Now we are enabled to apply Lemma 13.9, getting $\sigma_{\tau_{2,3,3}}(d^3(w)) \geqslant \alpha$.
Finally, using (4), (5):

$$\tau(d^3[w,w^1,\alpha](w)) \leqslant \tau(3-2\alpha, 1-\alpha+\alpha) < \tau_{2,3,3}.$$

Case I.3: $(\#^2_b + \#^2_{\overline{b}})(F_0(w)) = 2$. Analogously to case I.2.
Case II: $\rho(w) \geqslant 3$.
By Lemma 13.10 and Lemmas 11.1, 11.2 we obtain the estimations

$$\begin{aligned} d^3_0(w) &\geqslant \Delta_0 n(w) \cdot (1-2\alpha) + \alpha \cdot \min(v_0(w), 2) \\ &\geqslant (\rho(w)+1) \cdot (1-2\alpha) + 2\alpha, \\ d^3_1(w) &\geqslant 1 - \alpha \cdot \min(\rho(w)-1, 5). \\ &\Rightarrow \quad d^3(w) \geqslant t(\rho(w)) := ((\rho(w)+1)\cdot(1-2\alpha)+2\alpha, 1-\alpha\cdot\min(\rho(w)-1,5)). \end{aligned}$$

Calculating $\tau(t(3)) = \tau_{2,3,3}$, $\tau(t(4)) = 1.46..$, $\tau(t(5)) = 1.44..$, $\tau(t(6)) = 1.44..$ and $(\rho(w) \geqslant 7 \Rightarrow t(\rho(w)) > t(6))$, the proof of Lemma 13.11 is finished. □

13.5. The case "B"

The last part in the series of Lemmas 13.7 and 13.10 is

Lemma 13.12. *For* $w \in \mathrm{nds}(\mathscr{T}_{\mathscr{N}_3}(F_0))$ *with* "B"(w) *and for* $i \in I(w)$

$$\zeta_i(w) \leqslant \Delta_i n(w) + i - 1 - \min(v_i(w), 5)$$

holds.

Lemma 13.13. *Consider* $w \in \mathrm{nds}(\mathscr{T}_{\mathscr{N}_3}(F_0))$ *with* "B"(w). *Assume* w^1 *is not a leaf of* $\mathscr{T}_{\mathscr{N}_3}(F_0)$. *Then we have*

$$\tau(d^3[w, w^1, \max(\min(\sigma_{\tau_{2,3,3}}(d^3(w^1)), \alpha), 0)](w)) \leqslant \tau_{2,3,3}.$$

Proof ((*Rough*) *Outline*). Since the “normal case” “BN”(w) has been constitutional for the bound (recall Section 11), the critical cases are “BB$_0$”(w) and “BB$_1$”(w).

In these cases $\mathcal{N}_3$ adds blocked 3-clauses with the aim to create additional new 2-clauses after branching (two new 2-clauses are already assured by the reduction process (remember Section 5) and the choice of the branching literal).

In fact these new 3-clauses become eliminated after branching (in the branch where only two new 2-clauses have been created) or become 2-clauses (in the other branch(!)) since they all contain the branching literal.

If all these new 3-clauses (really) yield new and additional 2-clauses we have no problems. More difficult are the cases where these 2-clauses either are already present (i.e., are not “new”), or do coincide with those two new 2-clauses mentioned above or among one another (i.e., they are not “additional”).

In these “overlapping” cases (finally) the surplus established by Lemma 13.9 compensates the missing new 2-clause.

Proof. First we consider the combinatorial situation in some detail.

By Lemma 5.1, part 4(e) we see, using $l := l(w)$:

$$\hat{\varphi}_0 := \hat{\varphi}_0(w) = \langle l \to 0, x \to 1\rangle, \quad \hat{\varphi}_1 := \hat{\varphi}_1(w) = \langle l \to 1\rangle,$$

where

$$\{l, x\} \in F_0 := F_0(w)$$

is the l-occurrence in $F_0^{[2]}$. Due to Lemma 5.1, part 4(f)i there is also

$$\{l, y, z\} \in F_0^{[3]},$$

where by Lemma 5.1, parts 4(e) and 4(g) $\mathrm{Var}(x) \notin \mathrm{Var}(\{y, z\})$ holds.

Since $\{l, x\}$ is not blocked for x w.r.t. F_0 (see Section 6.4.1), there is

$$\{\bar{x}, u, v\} \in F_0^{[3]}$$

with $\mathrm{Var}(l) \notin \mathrm{Var}(\{u, v\})$, and Lemma 5.1, part 4(h) gives $\{u, v\} \neq \{y, z\}$. Hence

$$N(\hat{\varphi}_0, F_0) \supseteq \{\{u, v\}, \{y, z\}\}, \quad |N(\hat{\varphi}_0, F_0)| \geqslant 2. \tag{1}$$

Now let us look at the $\bar{l}$-occurrences in F_0. Because of $\rho(F_0) = 1$ and using Lemma 5.1, part 4(f)ii we have $\#_{\bar{l}}(F_0) \geqslant 2$, and thus there exist

$$\{\bar{l}, a, b\}, \{\bar{l}, c, d\} \in F_0^{[3]}$$

with $\{a, b\} \neq \{c, d\}$. Lemma 5.1, part 4(e) yields $x \notin \{a, b, c, d\}$ (however $\bar{x} \in \{a, b, c, d\}$ is possible). We conclude

$$N(\hat{\varphi}_1, F_0) \supseteq \{\{a, b\}, \{c, d\}\}, \quad |N(\hat{\varphi}_1, F_0)| \geqslant 2. \tag{2}$$

(By the way, (of course) Lemma 5.1, part 4(g) is also used here.)

Now we turn to the estimation of $d^3(w)$. Lemma 13.12 gives

$$d_0^3(w) \geqslant \Delta_0 n \cdot (1-\alpha) + \alpha \cdot (1 + \min(v_0, 5)) \tag{3}$$

$$\geqslant 2 \cdot (1-\alpha) + \alpha \cdot (1+2) = 2 + \alpha, \tag{4}$$

$$d_1^3(w) \geqslant \Delta_0 n \cdot (1-\alpha) + \alpha \cdot \min(v_1, 5) \tag{5}$$

$$\geqslant 1 \cdot (1-\alpha) + \alpha \cdot 2 = 1 + \alpha, \tag{6}$$

where $v_{0/1} := v_{0/1}(w)$ and $\Delta_{0/1} n := \Delta_{0/1} n(w)$.

If $\Delta_0 \geqslant 3$, then (3) and (6) yield

$$\tau(d^3(w)) \leqslant \tau(3 \cdot (1-\alpha) + \alpha \cdot (1+1), 1+\alpha) = \tau(3-\alpha, 1+\alpha) = 1.45.. < \tau_{2,3,3}.$$

Furthermore, in case $\Delta_1 \geqslant 2$ we obtain by (4) and (5)

$$\tau(d^3(w)) \leqslant \tau(2+\alpha, 2 \cdot (1-\alpha) + \alpha) = \tau(2+\alpha, 2-\alpha) = 1.41.. < \tau_{2,3,3}.$$

Thus we assume $\Delta_0 = 2$ and $\Delta_1 = 1$ in the sequel. By Lemma 11.1, part 2(b) it follows

$$N_i(w) \subseteq N_i^r(w)$$

for $i \in I(w)$. Moreover (see Lemma 5.1, part 3(d)):

$$F_0(w^i) = (F_0 \setminus \{\{l, x\}\}) \cup N_i^r(w).$$

In case "BN"(w) (3) and (5) just give

$$\tau(d^3(w)) \leqslant \tau(2 + 2\alpha, 1 + 2\alpha) = \tau_{2,3,3}.$$

So let us assume "BB$_0$"(w) or "BB$_1$"(w) is the case. Note that in our situation we can describe these cases as follows:

$$\text{“BB}_0\text{”}(w) \Leftrightarrow N(\hat{\varphi}_0, F_0) = \{\{u, v\}, \{y, z\}\}, \tag{7}$$

$$\text{“BB}_1\text{”}(w) \Leftrightarrow N(\hat{\varphi}_0, F_0) \supset \{\{u, v\}, \{y, z\}\} \wedge N(\hat{\varphi}_1, F_0) = \{\{a, b\}, \{c, d\}\}. \tag{8}$$

Case I: "BB$_0$"(w). Here we have

$$F := F(w) = F_0 \cup B_{\bar{l}}(F_0)^{[3]}.$$

If $\#_l(F_0) \geqslant 3$ would be the case, then there would be $\{l, x', y'\} \in F^{[3]}$ with $\{x', y'\} \neq \{x, y\}$ and $\{x', y'\} \in N(\hat{\varphi}_0, F_0)$, and by Lemma 5.1, part 4(h) also $\{x', y'\} \neq \{u, v\}$ would hold, contradicting case I (see (1)).

So we conclude $\#_l(F_0) = 2$, and therefore

$$B_{\bar{l}}(F_0) = B_{\bar{l}}(F_0)^{[3]} = \{\{\bar{l}, \bar{x}, \bar{y}\}, \{\bar{l}, \bar{x}, \bar{z}\}\}.$$

Because of $\rho(F_0)=1$ and $\{l,x\}\in F_0$ we have $\{\overline{x},\overline{y}\},\{\overline{x},\overline{z}\}\notin F_0$, and thus

$$N_1(w)\supseteq\{\{a,b\},\{c,d\},\{\overline{x},\overline{y}\},\{\overline{x},\overline{z}\}\}.$$

Now Lemma 5.1, part 4(i) (stating the absence of blocked 3-clauses in F_0) gives

$$|N_1(w)|\geqslant 4, \Rightarrow v_1\geqslant 4,$$

and thus by (4), (5) we get

$$\tau(d^3(w))\leqslant\tau(2+\alpha,1+3\alpha)=1.496..<\tau_{2,3,3}.$$

The remaining case is

Case II: "BB_1"(w). We have

$$F=F_0\cup B_l(F_0)^{[3]}.$$

If ("just by chance") $v_1\geqslant 3$ holds, then (again) $\tau(d^3(w))\leqslant\tau(2+2\alpha,1+2\alpha)=\tau_{2,3,3}$ is the case. So we assume $v_1=2$, i.e.,

$$N_1^r(w)=\{\{a,b\},\{c,d\}\}.$$

Hence $\#_{\overline{l}}(F_0)=2$ must hold, and we infer (recall (2)):

$$B_l(F_0)=\{\{l,\overline{a},\overline{c}\},\{l,\overline{a},\overline{d}\},\{l,\overline{b},\overline{c}\},\{l,\overline{b},\overline{d}\}\}\cap\mathscr{CL}. \tag{9}$$

Here two things could happen:

(i) in case $\{a,b\}\cap\{c,d\}\neq\emptyset$ some of these clauses would in fact be 2-clauses; and

(ii) in case $\{a,b\}\cap\overline{\{c,d\}}\neq\emptyset$ some of these clauses would in fact be no clauses (!) (therefore we added "$\cap\mathscr{CL}$").

Case (i) is easy to handle, since in our situation we simply have $\{a,b\}\cap\{c,d\}=\emptyset$: *Assume* $\{a,b\}\cap\{c,d\}\neq\emptyset$. W.l.o.g.: $a=c$.

Now Lemma 5.1, part 4(f)i gives $\{l,\overline{a}\}\in F_0$, thus $\{l,\overline{a}\}=\{l,x\}$, but this is impossible since $\{l,x\}$ is not blocked for l w.r.t. F_0 (see Section 6.4.1) – *contradiction.*

Case (ii) is more complicated. We will show that now either $v_0\geqslant 5$ with

$$\tau(d^3(w))\leqslant\tau(2+4\alpha,1+\alpha)=1.497..<\tau_{2,3,3}$$

or $\sigma_{\tau_{2,3,3}}(w)\geqslant\alpha$ with

$$\tau(d^3[w,w^1,\alpha](w))\leqslant\tau(2+2\alpha,1+\alpha+\alpha)=\tau_{2,3,3}$$

(or both) must hold. To that end assume $v_0\leqslant 4$ and $\sigma_{\tau_{2,3,3}}(w)<\alpha$.

Now Lemma 13.9 assures $\{a,b\}\cap\overline{\{c,d\}}\neq\emptyset$, and thus

$$B_l(F_0)=B_l(F_0)^{[3]},\qquad |B_l(F_0)|=4.$$

Furthermore, Lemma 13.9 yields

$$\{\overline{a},\overline{b},\overline{c},\overline{d}\}\cap\mathrm{Lit}\,(F_0\setminus\{\{l,x\}\})=\emptyset,$$

since otherwise we had, say, $\{\overline{a},e\}\in F_0$ and thus $\{a,b\},\{\overline{a},e\}\in F_0(w^1)$.

We conclude (recall $x \notin \{a,b,c,d\}$)

$$\hat{\varphi}_0 * B_l(F_0) \subseteq N_0(w).$$

In case $x \notin \{\overline{a},\overline{b},\overline{c},\overline{d}\}$ we had $|\hat{\varphi}_0 * B_l(F_0)| = 4$. Lemma 5.1, part 4(i) gives $\{y,z\} \notin \hat{\varphi}_0 * B_l(F_0)$, and thus we had $|N_0(w)| \geqslant 4+1=5$ contradicting $v_0 \leqslant 4$.

So, finally, $x \in \{\overline{a},\overline{b},\overline{c},\overline{d}\}$ must hold. W.l.o.g.: $x = \overline{a}$, i.e.,

$$\{l,x\} = \{l,\overline{a}\}$$

and

$$\{\{\overline{b},\overline{c}\},\{\overline{b},\overline{d}\}\} \subseteq N_0(w).$$

Since we assumed $|N(\hat{\varphi}_0,F_0)| \geqslant 3$ and $v_0 \leqslant 4$, one of $\{\overline{b},\overline{c}\}$, $\{\overline{b},\overline{d}\}$ must already be present in $N(\hat{\varphi}_0,F_0)$:

$$\{\{\overline{b},\overline{c}\},\{\overline{b},\overline{d}\}\} \cap N(\hat{\varphi}_0,F_0) \neq \emptyset.$$

Since by Lemma 5.1, part 4(i) we have $\{l,\overline{b},\overline{c}\},\{l,\overline{b},\overline{d}\} \notin F_0$, it follows

$$\{\overline{x},\overline{b},\overline{c}\} \in F_0 \vee \{\overline{x},\overline{b},\overline{d}\} \in F_0.$$

W.l.o.g.: $\{\overline{x},\overline{b},\overline{c}\} \in F_0$. Thus

$$\{\overline{x},\overline{b},\overline{c}\},\{\overline{x},b\} \in \hat{\varphi}_1 * F$$

($\{\overline{x},b\} = \hat{\varphi}_1 * \{\overline{l},a,b\}$), and now Lemma 5.1, part 6 establishes

$$\{\overline{x},\overline{c}\} \in F_0(w^1) \vee \{\overline{b},\overline{c}\} \in F_0(w^1).$$

However we have $\{c,d\} \in F_0(w^1)$ as well, contradicting Lemma 13.9. □

13.6. The final step

Lemma 13.14. *For all $F_0 \in 3\text{-}\mathscr{CLS}_r$ there is a distance function d for $\mathscr{T}_{\mathscr{N}_3}(F_0)$ with*
1. $\tau_{\max}(d,\mathscr{T}_{\mathscr{N}_3}(F_0))) \leqslant \tau_{2,3,3}$,
2. $\sum d(\mathrm{root}(\mathscr{T}_{\mathscr{N}_3}(F_0))) \leqslant n(F_0) - c$, *where* $c := 1-\alpha$.

Proof. Lemma 10.5 yields (using $\omega_0 = \mathrm{root}(\mathscr{T}_{\mathscr{N}_3}(F_0))$):

$$\sum d^3(\omega_0) \leqslant n(F_0) + \alpha \cdot 5 - \alpha \cdot 0 - 3 \cdot (1-\alpha) = n(F_0) + 8\alpha - 3,$$

where $m_k(w) \geqslant n(w) - \alpha \cdot n(w) = n(w) \cdot (1-\alpha) \geqslant 3 \cdot (1-\alpha)$ for $w \in \mathrm{nds}(\mathscr{T}_{\mathscr{N}_3}(F_0))$ follows by Lemma 9.1, part 6(c) and Lemma 5.1, part 4(a).

Performing the balancings of d^3 according to Lemmas 13.4, 13.11 and 13.13 (see Lemma 12.1), getting d, increases the maximal sum by at most $(2-8\alpha)+\alpha = 2-7\alpha$. Thus

$$\sum d(\omega_0) \leqslant n(F_0) + 8\alpha - 3 + 2 - 7\alpha = n(F_0) - (1-\alpha).$$

Lemmas 13.3, 13.4, 13.6, 13.11 and 13.13 together with 12.1 give $\tau_{\max}(d, \mathscr{T}_{\mathscr{N}_3}(F_0)) \leqslant \tau_{2,3,3}$. $\Box$

Now the τ-Lemma 8.2 together with Lemma 13.14 and $2^{-c} = 0.5448..$ proves Theorem 2.

14. Final remarks

14.1. Blocked clauses and Extended Resolution

Extended Resolution, introduced in [32], allows the extension of the set F_0 of premises by repeated applications of $F_i \to F_{i+1} = F_i \cup \{\{\bar{l}, \bar{a}, \bar{b}\}, \{l, a\}, \{l, b\}\}$ for $\operatorname{Var}(l) \notin \operatorname{Var}(F_i)$. For the known "difficult" formulas or clause-sets (e.g. pigeonhole formulas and formulas corresponding to graphs via the Tseitin-construction [32]) Extended Resolution admits polynomial-size proofs and hence till today no super-polynomial lower bound for Extended Resolution is known.

By observing that the clauses $\{\bar{l}, \bar{a}, \bar{b}\}, \{l, a\}, \{l, b\}$ are just blocked clauses (for l respectively $\bar{l}$ and for every order of addition), we can easily generalize this concept by allowing the addition of *arbitrary blocked clauses*. In this way we get a more symmetric concept of extensions, and also it is possible now to add (blocked) new clauses *without new variables* (as used in $\mathscr{N}_3$).

Definition 14.1. $E \in \mathscr{CLS}$ is called a *simple blocked extension* for $F \in \mathscr{CLS}$ if there is an order $E = \{C_1, \dots, C_m\}$ such that

$$\forall\, 1 \leqslant i \leqslant m : C_i \text{ is blocked w.r.t. } F \cup \{C_1, \dots, C_{i-1}\}$$

holds. A *simple* $(\boldsymbol{K}, \boldsymbol{K'})$*-resolution proof* $\mathscr{P}$ of C from $F \in \mathscr{CLS}$ for $K, K' \in \mathbb{N}_0 \cup \{+\infty\}$ and $C \in \mathscr{CL}$ is a resolution proof of C from $F \cup E$, where E is a simple blocked extension for F with clause-length at most K, and with at most K' new variables (in total):
- $E \in K\text{-}\mathscr{CLS}$,
- $|\operatorname{Var}(E) \setminus \operatorname{Var}(F)| \leqslant K'$.

The length of $\mathscr{P}$ is its ordinary length as resolution proof. $\mathscr{P}$ is called tree-like if it is tree-like as resolution proof.

The correctness of this concept of proofs for the deduction of $\perp$ follows from Lemma 3.1 (but note that simple blocked extensions are not conservative with respect to *general* implication).

In [18] this concept is studied in a more general form, called "Generalized Extended Resolution"[40] which allows also (implicit) eliminations of blocked clauses, making the notion of extension independent of the order of addition. Among others an exponential

[40] "(K, K')-resolution" (without the adjective "simple") refers to this more general concept.

lower bound for $(\infty, 0)$-resolution proofs is shown for the Pigeonhole Principle. To obtain from that a (sub)exponential lower bound for $\mathcal{N}_3$, two difficulties must be overcome:

– blocked clauses are introduced in $\mathcal{N}_3$ at all nodes of the computation tree, not only at the root,
– and furthermore the dynamical addition *and* elimination of clauses (allowing to add more blocked clauses) has to be handled.

A lower bound for (∞, ∞)-resolution seems to be intractable at this time, since simple $(3, \infty)$-resolution already contains Extended Resolution.[41]

We conclude this subsection by showing that simple tree-like $(3, 0)$-resolution can not be simulated by resolution:

Lemma 14.1. *Simple treelike* $(3,0)$*-resolution cannot* (*even for* 3*-clause-sets*) *be simulated polynomially by resolution.*

Proof. Let $(F_n)_{n \in \mathbb{N}}$ be any sequence of unsatisfiable 3-clause-sets such that there is no polynomial bound on the length of resolution proofs for (F_n), but there are polynomially bounded extended resolution proofs for (F_n). (For example consider the Pigeonhole Formulas transformed into 3-$\mathcal{CLS}$ in the standard way.)

Since tree-like Extended Resolution polynomially simulates Extended Resolution,[42] for each n there is a sequence $(E_i^n)_{i \in \{1,\dots,m_n\}}$ of applications of the extension rule, i.e., for all $1 \leqslant i \leqslant m_n$

$$E_i^n = \{\{x_i^n, a_i^n, b_i^n\}, \{\overline{x_i^n}, \overline{a_i^n}\}, \{\overline{x_i^n}, \overline{b_i^n}\}\}$$

$$\operatorname{Var}(x_i^n) \notin \operatorname{Var}(F_n) \cup \operatorname{Var}\left(\bigcup_{k<i} E_k^n\right),$$

$$\operatorname{Var}(\{a_i^n, b_i^n\}) \subseteq \operatorname{Var}(F_n) \cup \operatorname{Var}\left(\bigcup_{k<i} E_k^n\right)$$

holds, such that there are polynomial (in $\ell(F_n)$) tree-like resolution proofs of $\perp$ from $F_n \cup E_n$, where $E_n := \bigcup_{i \leqslant m_n} E_i^n$.

Take a variable y not contained in any $F_n \cup E_n$, and define

$$E_n' := \{\{x_i^n, y\}\}_{1 \leqslant i \leqslant m_n}.$$

$F_n' := F_n \cup E_n'$ has the same complexity w.r.t. resolution as F_n. On the other hand F_n' has a short simple tree-like $(3,0)$-resolution proof:

The simple blocked extension E_n'' for F_n' is obtained as follows:

1. First add $\{y\}$ (blocked for y).

[41] In fact they are (polynomially) equivalent as shown in [18].

[42] Lemma 4.4.8 from [15] proves that tree-like Frege Systems polynomially simulate Frege Systems, while Lemma 4.5.8 from [15] says that Extended Resolution and Extended Frege Systems polynomially simulate each other. Now it is not hard to see that the proof of Lemma 4.5.8 also shows that tree-like Extended Resolution polynomially simulates tree-like Extended Frege Systems, and that the construction of Lemma 4.4.8 also gives that Extended tree-like Frege Systems polynomially simulate Extended Frege Systems.

2. (a) Add $\{x_1^n, a_1^n, b_1^n\}$ (blocked for x_1^n).
   (b) Add $\{\overline{x_1^n}, \overline{a_1^n}, \bar{y}\}$ and $\{\overline{x_1^n}, \overline{b_1^n}, \bar{y}\}$ (blocked for $\overline{x_1^n}$).
3. In the same way add all $\{x_i^n, a_i^n, b_i^n\}$, $\{\overline{x_i^n}, \overline{a_i^n}, \bar{y}\}$ and $\{\overline{x_i^n}, \overline{b_i^n}, \bar{y}\}$ for $2 \leqslant i \leqslant m_n$.

E_n'' does not contain new variables due to the dummy variables in E_n'.

And by Unit-resolution we obtain E_n from E_n''.[43] □

14.2. Comments to our method for estimating the size of trees

The next lemma motivates that for the purpose of estimating

$$(*) \quad \#\mathrm{lvs}(T) \leqslant \tau_{\max}(d, T)^{\sum d(\mathrm{root}(T))}$$

(recall Section 8) this upper bound is the better the closer the τ-values are lying together and thus for a good estimation the job is to balance the different τ-values over the tree:

Lemma 14.2. *For any tree T up to a positive factor there is exactly one distance function d_T with $\#\mathrm{lvs}(T) = \tau_{\max}(d_T, T)^{\sum d_T(\mathrm{root}(T))}$. d_T is characterized by the condition that for every path from the root to a leaf the sum of distances is the same, and that $\tau(d(w))$ is constant on inner nodes.*

For the choice of the τ-function we have the following justification:

Lemma 14.3. *The τ-function is the (pointwise) minimal function $\tau : \mathscr{BT} \to [1, +\infty)$ such that for all trees T and all distance functions d for T formula $(*)$ is valid.*

Finally we note that in our handling of ζ we used in fact only very general properties of the z_k-measures, so that the distance function d^3 in a generalized framework could be useful also for other applications.

14.3. General complexity of SAT-decision

Here we consider the general time complexity of SAT-decision for subclasses of $\mathscr{CLS}$, more strictly speaking the exponential (or "mathematical") part of the time complexity, abstracting from polynomial factors.

Definition 14.2. For $\mathscr{K} \subseteq \mathscr{CLS}$ and $\mu : \mathscr{K} \to \mathbb{R}_+$ we define the *power coefficient* $\boldsymbol{\alpha(\mathscr{K}, \mu)}$ as the infimum of exponents α which bound (up to a polynomial in $\ell(F)$) by

[43] If the reader is disturbed by the fact that F_n' contains blocked clauses, he may use instead: $E_n' := \{\{x_i^n, y\}, \{\overline{x_i^n}, y\}, \{a_i^n, b_i^n, y\} : 1 \leqslant i \leqslant m_n\} \cup \{\{\bar{y}, v\}, \{\bar{v}, w\}, \{\bar{w}, y\}\}$. To see that $F_n' = F_n \cup E_n'$ has the same complexity for resolution as F_n, apply the autark assignment $\langle y \to 1, v \to 1, w \to 1\rangle$ and note that clauses eliminated by some autark assignment can be eliminated from any resolution proof of $\bot$.
The simple blocked extension E_n'' for F_n' is now $E_n'' := \{\{x_i^n, \bar{y}\}, \{\overline{x_i^n}, \overline{a_i^n}, \bar{y}\}, \{\overline{x_i^n}, \overline{b_i^n}, \bar{y}\} : 1 \leqslant i \leqslant m_n\}$.

$2^{\alpha\cdot\mu}$ the time complexity of deciding SAT for $\mathscr{K}$ (by multi-tape Turing-machines):

$$\boldsymbol{\alpha(\mathscr{K}, m)} := \inf\{\alpha \in \mathbb{R}_+ : \exists_{\text{TM }\mathscr{M}} \exists_{\text{polynomial } P}[\mathscr{M} \text{ decides SAT for } \mathscr{K} \text{ with time-bound } (P(\ell(F)) \cdot 2^{\alpha\cdot\mu(F)})_{F\in\mathscr{K}}]\}.$$[44]

Due to the polynomial factor $P(\ell(F))$ power coefficients do not depend on the special coding and the special model of computation (when considering at least 2-tape TMs (cf. [26], 2.8.9)). In contrast to the notion of "power indices" from [14], the notion of power coefficients is designed for the consideration of those classes $\mathscr{K}$, for which there is a polynomial time and *linear size* reduction $t : \mathscr{CLS} \to \mathscr{K}$, i.e., t is computable in polynomial time w.r.t. $\ell(F)$ and there is a constant c such that for all $F \in \mathscr{CLS}$

$$t(F) \overset{\text{sat}}{\equiv} F \quad \text{and} \quad \ell(t(F)) \leqslant c \cdot \ell(F)$$

holds.

14.3.1. The known upper bounds for power coefficients

1. (a) $\alpha(\mathscr{CLS}\text{-}3, n) \leqslant \log_2 \tau(9,9,9) = 0.17610..$, where $\mathscr{CLS}$-3 is the class of clause-sets with every variable occurring at most three times [22];
   (b) $\alpha(\mathscr{CLS}\text{-}(1,\infty), n) \leqslant \log_2 \tau(3,3,3) = 0.52832..$, where $\mathscr{CLS}\text{-}(1,\infty)$ is the class of clause-sets in which every variable appears in one sign at most once (for the other sign there are no restrictions) [22, 23].
2. (a) $\alpha(3\text{-}\mathscr{CLS}, n) \leqslant \log_2 \tau_{2,3,3} = 0.58921..$ (Theorem 2), which can be improved somewhat to $\alpha(3\text{-}\mathscr{CLS}, n) \leqslant \log_2 1.49625.. = 0.58135..$ by refining the case "BN" [20, 30];
   (b) $\alpha(p\text{-}\mathscr{CLS}, n) \leqslant \log_2 \tau(1, \ldots, p-1) < 1 + \log_2(1 - 2^{-p})$ for $p \geqslant 2$. ([23, 25]; the generalization of $\mathscr{N}_3$ (and of the improved bound!) to p-$\mathscr{CLS}$ should be possible but is not an obvious task);
   (c) $\alpha(p\text{-}\mathscr{CLS}, n) \leqslant 1 - \varepsilon(p)/p$ ([27]), where $0 < \varepsilon(p) < 1/2$ is the solution of $1 - \varepsilon/p = H_2(\varepsilon)$ for the binary entropy function $H_2(\varepsilon) = -\varepsilon \cdot \log_2 \varepsilon - (1-\varepsilon) \cdot \log_2(1-\varepsilon)$, which is better than (b) for $p \geqslant 5$ ($\lim_{p\to\infty} \varepsilon(p) = 1/2$).
3. Using $c(F) := |F|$, $c^i(F) := c(F^{[i]})$ and $c^*(F) := \sum_{i=3}^{n(F)} (i-2) \cdot c^i(F)$ it is known that
   (a) $\alpha(\mathscr{CLS}, c^*) = \alpha(3\text{-}\mathscr{CLS}, c^*) \leqslant \log_2 \tau(1,4) = 0.46495..$ ([1] and Lemma 14.6);
   (b) $\alpha(\mathscr{CLS}, c) \leqslant \log_2 \tau(6,7,6,7) = 0.30896..$ [12].
4. $\alpha(\mathscr{CLS}, \ell) \leqslant 1/10$ [22].

14.3.2. Two variants for the number n of variables, counting only "hard" variables

By well-known techniques we obtain the following two lemmas.

Lemma 14.4. $\alpha(\mathscr{CLS}, n) = \alpha(3\text{-}\mathscr{CLS}, n') = \alpha(\mathscr{CLS}, n')$, *where for* $F \in \mathscr{CLS}$ *we define* $n'(F) := |\{v \in \operatorname{Var}(F) : \#_v(F) \cdot \#_{\overline{v}}(F) \geqslant 2\}|$ *as the number of variables which occur in one sign at least once in F and in the other sign at least twice.*

[44] $\inf \emptyset = +\infty$.

Lemma 14.4 gives $\alpha(\mathcal{CLS}, n') \leqslant 1$, generalizing the polynomial decidability of $\mathcal{CLS}$-2. Analogously, the following lemma generalizes the polynomial decidability of 2-$\mathcal{CLS}$:

Lemma 14.5. $\alpha(3\text{-}\mathcal{CLS}, n^*) = \alpha(3\text{-}\mathcal{CLS}, n)$, *where for* $F \in 3\text{-}\mathcal{CLS}$ *we define* $n^*(F)$ *as the number of variables of* F *occurring in both signs in the* 3*-clauses of* F.

14.3.3. 3-coloring and "(3,2)-SSS"

Here we discuss (shortly) the close relation between 3-SAT and 3-coloring.

In [2] the notion of "(a,b)-*SSS*" ("SSS" stands for "*Symbol System Satisfiability*") is introduced. An (a,b)-SSS instance consists of:
- a set V of *vertices* with $n := |V|$ elements,
- for each vertex a list of $\leqslant a$ *colors*,
- and a set of *constraints*, where a constraint is a list $((v_1, i_1), \ldots, (v_r, i_r))$ of pairs of vertices and colors with $r \leqslant b$ components.

The meaning of such a constraint is: vertex v_1 is not colored with color i_1 or … or vertex v_r is not colored with color i_r. The problem is to decide whether there is a coloring of the vertices fulfilling all constraints. The class of these (decision) problems is denoted by "(a,b)-SSS".

$(2, p)$-SSS is (up to renaming) identical with p-$\mathcal{CLS}$,[45] while $(k,2)$-SSS is a natural (and useful) generalization of the k-coloring problem. As in Definition 14.2 we define the *power coefficients* $\boldsymbol{\alpha((a, b)\textbf{-SSS}, n)}$, using the number n of vertices in (a,b)-SSS instances.

Lemma 14.6. $\alpha((3,2)\text{-}SSS, n) = \alpha(3\text{-}\mathcal{CLS}, c^3)$ (*recall* 3 *in Section* 14.3.1). *The measure* c^3 *is extended to whole* $\mathcal{CLS}$ *by* c^*, *for which we obtain*: $\alpha(3\text{-}\mathcal{CLS}, c^*) = \alpha(\mathcal{CLS}, c^*)$.

Proof. For $\alpha((3,2)\text{-SSS}, n) \leqslant \alpha(3\text{-}\mathcal{CLS}, c^*)$ we transform a (3,2)-SSS instance in the natural way into a 3-clause-set (the variables are the pairs (v,i)), and observe that there are at most n clauses of length 3.

And for the other direction transform a 3-clause-set according to Lemma 1 of [1] into a (3,2)-SSS instance (the nodes are the clauses, the colors the position of the literals in the clauses) and eliminate all nodes corresponding to $\leqslant$2-clauses via Lemma 2 of [1].

Finally c^* is invariant w.r.t. the standard transformation $\mathcal{CLS} \to 3\text{-}\mathcal{CLS}$. $\square$

14.4. Leaving 3-$\mathcal{CLS}$ to obtain faster 3-SAT-decision when the variables occur on the average less than 5.9 times

With the help of the algorithm $\mathcal{L} : \mathcal{CLS} \to \{0,1\}$ from [22], realizing $\alpha(\mathcal{CLS}, \ell) \leqslant 1/10$, we are able to improve the bound 1.5045^n from Theorem 2 for all $F \in 3\text{-}\mathcal{CLS}$ fulfilling $\ell(F)/n(F) \leqslant c_0 := 10 \cdot \log_2 \tau_{2,3,3} = 5.8921..$ as follows.

[45] Of course with respect to $\mathcal{SAT}$.

Denote by $\mathscr{N}_3' : 3\text{-}\mathscr{CLS} \to \{0,1\}$ the algorithm from Section 6, completed by reducing the input $(F_0 \mapsto r(F_0))$ and excluding the trivial case $\bot \in r(F_0)$.

Then our combined algorithm $\mathscr{C}_3$ is given by

```
PROCEDURE $\mathscr{C}_3(F_0 \in 3\text{-}\mathscr{CLS}) : \{0,1\}$;
BEGIN
  IF $F = \top$ THEN RETURN 1
  ELSE IF $\bot \in F$ THEN Return 0
  ELSE IF $\ell(F)/n(F) \leqslant c_0$ THEN
    RETURN $\mathscr{L}(F)$
  ELSE
    RETURN $\mathscr{N}_3'(F)$
END $\mathscr{C}_3$.
```

Theorem 4. *The 3-SAT-algorithm $\mathscr{N}_3' : 3\text{-}\mathscr{CLS} \to \{0,1\}$ decides 3-SAT with the following time bound* ($F_0 \in 3\text{-}\mathscr{CLS}$, $\ell := \ell(F_0)$, $n := n(F_0)$):

$$\log_2 \#\mathrm{lvs}(\mathscr{T}_{\mathscr{C}_3}(F_0)) \leqslant \min(\log_2 \tau_{2,3,3} \cdot n, 1/10 \cdot \ell) = \begin{cases} 1/10 \cdot \ell & \textit{for } \ell/n \leqslant c_0 \\ \log_2 \tau_{2,3,3} \cdot n & \textit{else} \end{cases}.$$

Acknowledgements

I am very obliged to Henri Morel and Paul W. Purdom for encouraging me, for their help in improving the presentation of the paper, and for valuable discussions.

References

[1] R. Beigel, D. Eppstein, 3-coloring in time O(1.3446^n): a no-MIS algorithm, in: 36th IEEE Symp. on Foundations of Computer Science, 1995, pp. 444–452.

[2] R. Beigel, R. Floyd, The Language of Machines: An Introduction of Computability and Formal Languages, Computer Science Press, New York, 1993.

[3] E. Dantsin, Two systems for proving tautologies, based on the split method, J. Sov. Math. 22 (1983) 1293–1305.

[4] M. Davis, G. Logemann, D. Loveland, A machine program for theorem-proving, Comm. ACM 5 (1962) 394–397.

[5] M. Davis, H. Putnam, A computing procedure for quantification theory, J. ACM 7 (1960) 201–215.

[6] D. Du, J. Gu, P.M. Pardalos (Eds.), Satisfiability Problem: Theory and Applications (DIMACS Workshop March 11–13, 1996), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 35, American Mathematical Society, Providence, RI, 1997.

[7] J. Franco, G. Gallo, H. Büning, E. Speckenmeyer, C. Spera, (Eds.), Workshop on the Satisfiability Problem, Universität zu Köln, Report No. 96-230, Siena, April 29–May 3 1996.

[8] A.V. Gelder, A satisfiability tester for non-clausal propositional calculus, Inform. and Comput. 79 (1988) 1–21.

[9] A.V. Gelder, Propositional search with k-clause introduction can be polynomially simulated by resolution, in: 5th Internat. Symp. on Artificial Intelligence and Mathematics, January 1998.

[10] A.V. Gelder, Y.K. Tsuji, Satisfiability testing with more reasoning and less guessing, in: D.S. Johnson, M. Trick, (Eds.), Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, American Mathematical Society, Providence, RI, 1996, pp. 559–586.

[11] J. Gu, P.W. Purdom, J. Franco, B.W. Wah, Algorithms for the satisfiability (SAT) problem: A survey, in: D. Du, J. Gu, P.M. Pardalos (Eds.), Satisfiability Problem: Theory and Applications (DIMACS Workshop March 11–13, 1996), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 35, American Mathematical Society, Providence, RI, 1997.

[12] E. Hirsch, Two new upper bounds for SAT, Proc. SODA'98, to appear.

[13] J.N. Hooker, V. Vinay, Branching rules for satisfiability, J. Autom. Reasoning 15 (1995) 359–383.

[14] H.B. Hunt, R.E. Stearns, Power indices and easier hard problems, Math. Systems Theory 23 (1990) 209–225.

[15] J. Krajíček, Bounded Arithmetic, Propositional Logic, and Complexity Theory, Cambridge University Press, Cambridge, 1995.

[16] O. Kullmann, Methods for 3-SAT-decision in less than $2^{0.59 \cdot n}$ steps, Bull. Symbol. Logic 1 (1) (1995) 96–97. Abstracts of contributed papers of the Logic Colloquium' 93, Keele, England, July 20–29, 1993.

[17] O. Kullmann, A note on a generlization of extended resolution, in: [7, pp. 73–95]. Revised: [18].

[18] O. Kullmann, On a generalization of extended resolution, Discrete Appl. Math., submitted (special edition on the satisfiability problem); revised version of [17], 1996.

[19] O. Kullmann, Heuristics for SAT algorithms: A systematic study. SAT'98, Second Workshop on the Satisfiability Problem, May 10–14, 1998, Eringerfeld, Germany.

[20] O. Kullmann, Worst-case analysis, 3-SAT decision and lower bounds: approaches for improved SAT algorithms, in: D. Du, J. Gu, P.M. Pardalos (Eds.), Satisfiability Problem: Theory and Applications (DIMACS Workshop March 11–13, 1996), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 35, American Mathematical Society, Providence, RI, 1997, pp. 261–313.

[21] O. Kullmann, The combination of extended resolution and DPLL-algorithms, forthcoming.

[22] O. Kullmann, H. Luckhardt, Deciding propositional tautologies: algorithms and their complexity, Inform. and Comput. submitted.

[23] H. Luckhardt, Obere Komplexitätsschranken für TAUT-Entscheidungen, in: Frege Conf. 1984, Schwerin, Akademie-Verlag, Berlin, 1984, pp. 331–337.

[24] B. Monien, E. Speckenmeyer, 3-satisfiability is testable in $O(1.62^r)$ steps, Technical Report Bericht Nr. 3/1979, Universität-Gesamthochschule-Paderborn, 1979, Reihe Theoretische Informatik.

[25] B. Monien, E. Speckenmeyer, Solving satisfiability in less than 2^n steps, Discrete Appl. Math. 10 (1985) 287–295.

[26] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA; 1994.

[27] R. Paturi, P. Pudlak, F. Zane, Satisfiability coding lemma, in: Proc. 38th Annual Symp. Foundations of Computer Science (FOCS 97), 1997, pp. 566–574.

[28] P.W. Purdom, Solving satisfiability with less searching, IEEE Trans. Pattern Anal. Machine Intell. 6 (4) (1984) 510–513.

[29] I. Schiermeyer, Solving 3-satisfiability in less than 1.579^n steps, in: Selected paper from Computer Science Logic '92, Lecture Notes Computer Science, vol. 702, Springer, Berlin, 1992, pp. 379–394.

[30] I. Schiermeyer, Pure literal look ahead: an $O(1,497^n)$ 3-satisfiability algorithm, in: [7, pp. 127–136]. Siena, April 29–May 3 (1996).

[31] G.M.N. Stalmarck, Method and apparatus for checking propositional logic theorems in system analysis, 1990, European Patent Specification; application: December 19, 1990, Bulletin 90/51; publication: June 28, 1995, Bulletin 95/26.

[32] G.S. Tseitin, On the complexity of derivation in propositional calculus, in: Seminars in Mathematics. V.A. Steklov Mathematical Institute, Leningrad, vol. 8, 1968. (English translation 1970; A.O. Slisenko (Ed.))

[33] W. Zhang, Number of models and satisfiability of sets of clauses, Theoret. Comput. Sci. 155 (1996) 277–288.