Electronic Colloquium on Computational Complexity, Report No. 55 (2007)

Constraint satisfaction problems in clausal form: Autarkies and minimal unsatisfiability

Oliver Kullmann*

Computer Science Department
University of Wales Swansea
Swansea, SA2 8PP, UK

e-mail: O.Kullmann@Swansea.ac.uk
http://cs.swan.ac.uk/~csoliver

May 22, 2007

Abstract

We consider the problem of generalising boolean formulas in conjunctive normal form by allowing *non-boolean variables*, where our goal is to maintain *combinatorial* properties. Requiring that a literal involves only a single variable, the most general form of literals is given by the well-known "signed literals", however we will argue that only the most restricted form of generalised clause-sets, corresponding to "sets of no-goods" in the AI literature, maintains the essential properties of boolean conjunctive normal forms. We start our investigations by building up a solid foundation for (generalised) clause-sets, including the notion of autarky systems, the interplay between autarkies and resolution, and basic notions of (DP-)reductions. As a basic combinatorial parameter of generalised clause-sets, we introduce the (generalised) notion of *deficiency*, which in the boolean case is the difference between the number of clauses and the number of variables. We obtain *fixed parameter tractability* (FPT) of satisfiability decision for generalised clause-sets, using as parameter the maximal deficiency (over all sub-clause-sets). Another central result in the boolean case regarding the deficiency is the classification of *minimally unsatisfiable clause-sets* with low deficiency (MU(1), MU(2), ...). We generalise the well-known characterisations of boolean MU(1). The proofs for FPT and MU(1) are not straight-forward, but are obtained by an interplay between suitable generalisations of techniques and notions from the boolean case, and exploiting *combinatorial* properties of the natural translation of (generalised) clause-sets into boolean clause-sets. Of fundamental importance here is *autarky theory*, and we concentrate especially on *matching autarkies* (based on matching theory). A natural question considered here is to determine the structure of *(matching) lean clause-sets*, which do not admit non-trivial (matching) autarkies. Special lean clause-sets are minimally unsatisfiable (generalised) clause-sets, and we consider the generalisation to *irredundant clause-sets*, so that also satisfiable clause-sets can be taken into account, with a special emphasise on *hitting clause-sets* (which are irredundant in a very strong sense) and the generalisation to *multihitting* clause-sets.

*Supported by EPSRC Grant GR/S58393/01

ISSN 1433-8092

Contents

1 Introduction

Satisfiability problems with constraint variables having more than two values occur naturally at many places, for example in colouring problems. Translations into boolean satisfiability problems are interesting and useful (see [18, 47, 2] for various techniques), and may even improve performance of SAT solving, however they hide to a certain degree the structure of the original problem, which causes these translations typically to be not very well suited for theoretical studies on the structure of the original problem. In this article[1)] we study non-boolean satisfiability problems closest to boolean conjunctive normal form, namely satisfiability of what is called *generalised clause-sets* (or sets of "no-goods"). Combining suitable generalisations of boolean techniques with suitable translations into the boolean case we obtain non-trivial generalisations of fundamental theorems on autarkies and minimally unsatisfiable formulas.

Three aspects of clauses (as combinations of literals) make processing of boolean clause-sets especially efficient:

(i) When the underlying variable of a literals gets a value, then the literal is either true or false (this enables efficient handling of literals).

(ii) Only by assigning a value to all the variables in a clause can we falsify the clause, and for each variable the value here is uniquely determined (this makes a tight connection between partial assignments and clauses, and enables "conflict learning" by clauses).

(iii) By giving just one variable a right value we are always able to satisfy a clause (this enables simple satisfaction-based heuristics).

Taking these properties as axiomatic, a "generalised clause" should be a disjunction of generalised literals, and a "generalised literal" should have exactly one possibility to become false, while otherwise it should evaluate to true. We arrive naturally at the following concept for generalised literals (the earliest systematic use seems to

[1)] based on [37]

be in [3]): A variable v has a domain D_v of values, and a literal is a pair (v, ε) of the variable and a value $\varepsilon \in D_v$ such that the literal becomes true under an assignment φ iff φ sets v to a value different than ε (i.e., $\varphi(v) \in D_v \setminus \{\varepsilon\}$); to express this interpretation, often when displaying formulas we will write "$v \neq \varepsilon$" for the literal (v, ε). In case of $D_v = \{0, 1\}$ variable v becomes an ordinary boolean variable with the literal $(v, 0)$ representing the positive literal. We remark here, that a fourth property of boolean clauses, namely that if all literals except of one are falsified, that then the value for the variable in the remaining literal is uniquely determined, which is the basis for the ubiquitous unit-clause propagation, is necessarily lost here. In this article we investigate the basic combinatorial properties of generalised clause-sets, concentrating on the *theory of autarkies* and on structural properties of *minimally unsatisfiable generalised clause-sets.*

1.1 Generalising the notion of deficiency

Using $c(F)$ for the number of clauses in a boolean clause-set, and $n(F)$ for the number of variables, in [17] the *deficiency* $\boldsymbol{\delta(F)} := c(F) - n(F)$ has been introduced and made fruitful for the study of minimally unsatisfiable boolean clause-sets as well as for the introduction of a new polynomial time decidable class of "matched" satisfiable (boolean) clause-sets:

- Let $\mathcal{MUSAT}$ denote the class of minimally unsatisfiable clause-sets (unsatisfiable clause-sets, where each strict sub-clause-set is satisfiable). For boolean $F \in \mathcal{MUSAT}$ the property $\forall F' \subset F : \delta(F') < \delta(F)$ has been shown; using $\delta^*(F) := \max_{F' \subseteq F} \delta(F')$ for the *maximal deficiency* we get $\delta^*(F) = \delta(F)$ as well as "Tarsi's lemma" $\delta(F) \geq 1$ (since for the empty clause-set $\top \subset F$ we have $\delta(\top) = 0$).
- Considering only boolean clause-sets (for now), let the class $\mathcal{MSAT}$ of "matching satisfiable" clause-sets F be defined by the condition $\delta^*(F) = 0$. All matching satisfiable clause-sets are in fact satisfiable, since by Hall's theorem the bipartite graph $B(F)$ contains a matching covering all variables, where the vertices of $B(F)$ are the clauses of F on the one side and the variables of F on the other side, while an edge joins a variable and a clause if that variable appears in the clause (positively or negatively). Or, using Tarsi's lemma, one argues that if $F \in \mathcal{MSAT}$ would be unsatisfiable, then F would contain some minimally unsatisfiable $F' \subseteq F$, for which $\delta(F') \geq 1$ would hold, contradicting $\delta^*(F) = 0$.

The study of the levels $\mathcal{MUSAT}(k)$ of minimally unsatisfiable boolean clause-sets F with $\delta(F) \leq k$ has attracted some attention. In [1] (where also Tarsi's lemma has been proven) the class $\mathcal{SMUSAT}$ of "strongly minimally unsatisfiable clause-sets" has been introduced, which are minimally unsatisfiable clause-sets such that adding any literal to any clause renders them satisfiable, and a nice characterisation of $\mathcal{SMUSAT}(1) = \{F \in \mathcal{SMUSAT} : \delta(F) = 1\}$ has been given (yielding polynomial time decision of $\mathcal{SMUSAT}(1)$). Then in [10] a (poly-time) characterisation of $\mathcal{MUSAT}(1)$ has been obtained, followed by a characterisation of $\mathcal{MUSAT}(2)$ in [5], while in [52] some subclasses of $\mathcal{MUSAT}(3)$ and $\mathcal{MUSAT}(4)$ have been shown to be poly-time decidable. For arbitrary (constant) $k \in \mathbb{N}$ it has been shown in [4] that for $F \in \mathcal{MUSAT}(k)$ there is a tree resolution refutation using at most $2^{k-1} \cdot n(F)^2$ steps, and thus the classes $\mathcal{MUSAT}(k)$ are in NP. In [4] it has been conjectured that in fact all classes $\mathcal{MUSAT}(k)$ are in P.

This conjecture has been proven true in [26], using tools from matroid theory. Actually the classes $\mathcal{SAT}(k)$, consisting of all satisfiable clause-sets F with $\delta^*(F) \leq k$, have been shown poly-time decidable, from which immediately poly-time decision of the classes $\mathcal{MUSAT}(k)$ and $\mathcal{SMUSAT}(k)$ follows. Regarding the method used, more precisely the classes $\mathcal{USAT}(k)$ of *unsatisfiable* clause-sets F with $\delta^*(F) \leq k$ have been shown poly-time decidable by improving the "splitting theorem" from [10], yielding tree resolution refutations for F using at most $2^{k-1} \cdot n(F)$ steps and of a simple recursive structure, so that these refutations can be found in polynomial time by means of enumeration of the circuits of the transversal matroid $T(F)$ associated to the bipartite graph $B(F)$, where the independent subsets of $T(F)$ are the matching satisfiable sub-clause-sets of F. Independently also in [15] poly-time decision of the classes $\mathcal{MUSAT}(k)$ has been derived by extending techniques from bipartite matching theory to *directed* bipartite graphs. Improving the proofs from [15], the present author joint the team in [14]. Actually refining the techniques from [26], in [49] fixed-parameter tractability of $\mathcal{SAT}(k)$ is shown (all this for the boolean case).

After setting syntax and semantics for generalised clause-sets, the first main task tackled in the present paper is to transfer these results regarding the deficiency to generalised clause-sets. After suitably generalising the notion of *deficiency* and *matching satisfiability* (which is not completely straight-forward; in Subsection 4.5 an earlier version is discussed, which doesn't seem to have the right properties), in Corollary 4.10 the "satisfiability-based" approach from [14] yields polynomial time satisfiability decision for generalised clause-sets with bounded maximal deficiency. Generalising fixed-parameter tractability turns out not to be straight-forward (again), and only by combining the generalised approach with a suitable translation into the boolean case we arrive in Theorem 5.5 at fixed parameter tractability also for generalised clause-sets. The general framework for our considerations is autarky theory as started in [29], with emphasise on *matching autarkies* as introduced in [31].

A key point for structural investigations in (generalised) clause-sets is to understand the effects of applying partial assignments (see for example [6, 8], where splitting of minimally unsatisfiable boolean clause-sets is studied in some depth), and in this paper we consider the basic questions regarding *irredundant* and *minimally unsatisfiable generalised clause-sets* (which leads in a natural way to the study of *hitting clause-sets* and generalisations). The well-known classifications of the simplest case of minimally unsatisfiable clause-sets, namely boolean clause-sets of deficiency 1, finds a natural generalisation in Theorem 6.16 (where again the proof is not straight-forward, caused by the breakdown of the "saturation method").

1.2 Examples for translations: Colourings and homomorphisms

Given a hypergraph G and a set C of "colours", a C-colouring of G is a map $f : V(G) \to C$ such that no hyperedge $H \in E(G)$ is "monochromatic" (that is, there must be vertices $v, w \in H$ with $f(v) \neq f(w)$). Translating this colouring problem into a generalised satisfiability problem $F_C(G)$ is straightforward[2)], using

[2)] This translation directly generalises the well-known translation of graph 2-colouring problems into boolean CNF; if we add the translation from generalised clause-sets into boolean clause-sets via the standard translation (see Section 5), then the given translation also generalises the well-known standard translation of (arbitrary) graph colouring problems into boolean CNF.

the vertices of G as variables with (uniform) domain C: For each hyperedge $H \in E(G)$ and each colour $\varepsilon \in C$ form the clause $\{v \neq \varepsilon : v \in H\}$, and $F_C(G)$ is the set of all these clauses (thus $n(F_C(G)) = |V(G)|$ and $c(F_C(G)) = |C| \cdot |E(G)|$). Obviously the C-colourings of G correspond 1-1 to the (total) satisfying assignments for $F_C(G)$. Interesting examples of hypergraph colouring problems are given by the diagonal van der Waerden problems and the diagonal Ramsey problems. Computing van der Waerden numbers has been considered in [12, 25, 21], and it seems that SAT solvers are performing quite well on them, and that SAT solvers can help to compute new van der Waerden numbers[3)], so here is the problem:

Consider natural numbers $k, m, n \in \mathbb{N}$, and let the hypergraph $\mathrm{WH}(m,n)$ have vertex set $\{1,\dots,n\}$, while the hyperedges of $\mathrm{WH}(m,n)$ are the subsets $H \subseteq \{1,\dots,n\}$ of size m which form an arithmetic progression (that is, for every H there exist $a, d \in \{1,\dots,n\}$ with $H = \{a + i \cdot d : i \in \{0,\dots,m-1\}\}$); now the van der Waerden number $\mathrm{N_W}(k,m)$ is the minimal n such that $\mathrm{WH}(m,n)$ is not k-colourable. The corresponding generalised clause-sets are $\mathrm{F_W}(k,m,n) := F_{\{1,\dots,k\}}(\mathrm{WH}(m,n))$, and if $\mathrm{F_W}(k,m,n)$ is satisfiable, then $\mathrm{N_W}(k,m) > n$, while if $\mathrm{F_W}(k,m,n)$ is unsatisfiable, then $\mathrm{N_W}(k,m) \leq n$; for $k = 2$ we obtain boolean clause-sets (I would like to point out how natural the translation is — no auxiliary variables are involved[4)]). The only known precise van der Waerden numbers (besides the trivial values for $k = 1$ or $m \leq 2$) are $\mathrm{N_W}(2,3) = 9$, $\mathrm{N_W}(2,4) = 35$, $\mathrm{N_W}(2,5) = 178$, $\mathrm{N_W}(3,3) = 27$ and $\mathrm{N_W}(4,3) = 76$, and all these numbers can be easily calculated using most current SAT solvers; recently in [24] finally by (extensive) SAT-computations $\mathrm{N_W}(2,6) = 1132$ has been confirmed (as conjectured by [25]). Directly expressing the problem instance as a generalised clause-set, in this way also the non-diagonal versions of van der Waerden- and Ramsey problems can be immediately translated into generalised clause-sets (see [42]). For further applications of the mapping $G \mapsto F_C(G)$ from hypergraphs to clause-sets see [37, 38].

For the more general *list-hypergraph colouring problem*, for each vertex v a list $L(v)$ of allowed colours is given; this can be translated into a generalised clause-set $F_C(G, L)$ by just restricting the domain of v to $L(v)$. At this point it is worth noticing that also the still more general *list-hypergraph-homomorphism problem* has a direct (structure-preserving) translation into a satisfiability problem for generalised clause-sets. Given two hypergraphs G_1, G_2 and for each vertex $v \in V(G_1)$ a non-empty set $L(v) \subseteq V(G_2)$ of allowed image vertices, the problem is to find a map $f : V(G_1) \to V(G_2)$ with $f(v) \in L(v)$ for all $v \in V(G_1)$ such that for each hyperedge $H \in E(G_1)$ we have $f(H) \in E(G_2)$. Note that if we take for G_2 the hypergraph G_C with vertex set C and hyperedges all subsets of C with at least two elements, then the homomorphisms from G_1 to G_2 are exactly the C-colourings for G_1. For the translation of the list-hypergraph-homomorphism problem we use the set $V(G_1)$ of vertices as the set of variables, while the domain of v is $D_v = L(v)$, and for each hyperedge $H \in E(G_1)$ and for each map $f : H \to V(G_2)$ such that for each $v \in H$ we have $f(v) \in L(v)$ and such that $f(H) \notin E(G_2)$ holds, we have a clause $C_{H,f} := \{v \neq f(v) : v \in H\}$. Now satisfying assignments of the generalised clause-set $F(G_1, G_2, L)$ consisting of all clauses $C_{H,f}$ are exactly the hypergraph homomorphisms from G_1 to G_2 respecting the restrictions given by L. Note that the translation of hypergraph colouring problems is a special case via

[3)] On the other hand, the problem sizes of formulas related to unknown Ramsey numbers are likely too big to be manageable by any (current) SAT solver.

[4)] The translation is the core of two translations discussed in [12] — the additional constraints used in [12] just express the structural property of a generalised clause-set, that every variable gets exactly one value of its domain.

$F_C(G, L) = F(G, G_C, L)$.

The colourings considered above are also called "weak hypergraph colourings", to distinguish them from *strong hypergraph colourings* of hypergraph G by a set C of colours, which is a map $f : V(G) \to C$ such that for all hyperedges $H \in E(G)$ and all $v, w \in H$, $v \neq w$, we have $f(v) \neq f(w)$. Collecting all such binary clauses $\{f(v) \neq \varepsilon, f(w) \neq \varepsilon\}$ for $\varepsilon \in C$ we obtain a generalised clause-sets whose satisfying (total) assignments correspond 1-1 to the strong colourings of G. To conclude our list of translations for colouring problems we mention that *mixed hypergraph colouring* as studied in [51] takes a pair G_1, G_2 of hypergraphs with $V(G_1) = V(G_2)$, and the "mixed colourings" using colour-set C are (weak) colourings of G_1 which are *not* strong colourings of G_2; the most natural translation of this problem seems to consist of a pair of a generalised CNF (the translation of the (weak) colouring problem) and a generalised DNF (the negation of the translation of the strong colouring problem), using "monosigned literals" (see the next subsection), that is, allowing for inequalities "$v \neq \varepsilon$" as well as equalities "$v = \varepsilon$".

In the same vein as for hypergraph homomorphisms we can also translate *homomorphism problems for relational structures*: Let $\mathcal{A} = (A, (R_i)_{i \in I})$ and $\mathcal{B} = (B, (R'_i)_{i \in I})$ be two compatible finite relational structures, that is, A, B as well as I are finite sets, the R_i are relations (of arbitrary arity) on A and the R'_i are relations on B, while R_i has the same arity as R'_i. We want to express the set of homomorphisms $f : A \to B$, defined by the property that for $i \in I$ and all $\vec{x} \in R_i$ we have $f(\vec{x}) \in R'_i$, where f is applied componentwise to $\vec{x}$. For this we choose A as the set of variables, which all have the same domain B, and for each $i \in I$ and each $\vec{x} \in R_i$ and each $\vec{y} \in B^m \setminus R'_i$, where m is the arity of R_i, we have the clause $C_{i,\vec{x},\vec{y}} := \{\vec{x}_i \neq \vec{y}_i : i \in \{1, \dots, m\}\}$. We obtain the generalised clause-set $F(\mathcal{A}, \mathcal{B})$ by collecting all these clauses. The size of $F(\mathcal{A}, \mathcal{B})$ is polynomial in the sizes of A, B together with the number of tuples in R_i and the number of tuples *not* in R'_i. The requirement that the homomorphism f is injective can be encoded by the binary clauses $\{a \neq b, a' \neq b\}$ for $a, a' \in A$ with $a \neq a'$ and for $b \in B$.[5] We note that the translations $F(G_1, G_2, L)$ as well as $F(\mathcal{A}, \mathcal{B})$ are "direct" (homomorphisms are directly encoded as assignments) and "negative" (we use forbidden value combinations).

If we wish to have $F(\mathcal{A}, \mathcal{B})$ polynomial in the number of tuples in R'_i, then we can use an "indirect" and "positive" translation as follows: Variables are pairs $(i, \vec{x})$ for $i \in I$ and $\vec{x} \in R_i$, where the domain of variable $(i, \vec{x})$ is R'_i; so instead of mapping elements of A to elements of B, where constraints forbid that allowed tuples are mapped to disallowed tuples, here now we directly map tuples of relations in $\mathcal{A}$ to tuples in the corresponding relation in $\mathcal{B}$, and the constraints will ensure that this mapping actually is induced by some mapping from A to B. The constraints are the unit clauses $\{(i, \vec{x}) \neq \vec{y}\}$ for variables $(i, \vec{x})$ and values $\vec{y}$ such that indices k, k' exist with $\vec{x}_k = \vec{x}_{k'}$ but $\vec{y}_k \neq \vec{y}_{k'}$, and the binary clauses $\{(i, \vec{x}) \neq \vec{y}, (i', \vec{x}') \neq \vec{y}'\}$ for variables $(i, \vec{x}), (i', \vec{x}')$ and values $\vec{y}, \vec{y}'$, such that an index k exists with $\vec{x}_k = \vec{x}'_k$ but with $\vec{y}_k \neq \vec{y}'_k$.

[5] On the other hand, to formulate surjectivity of f requires an exponential number of clauses; one sees that for the good combinatorial properties which can be exploited for problems expressed in the language of (generalised) clause-sets (which is quite restricted from the constraint programming point of view) we have to pay the price that some natural problems do not have succinct representations.

1.3 Signed formulas and resolution

Are there still more general versions of "generalised conjunctive normal forms" suitable in our context? The most general form of variable-based literals allows literals of the form "$v \in S$" for some $S \subseteq D_v$ (generalised literals (v, ε) correspond to $S = D_v \setminus \{\varepsilon\}$); see [2], where S is called a "sign", while literals of the form "$v \in S$" are called "signed literals", and clause-sets made of signed literals "signed CNF formulas", or see [18] (using the same class for formulas, but calling them "nb-formulas"). Our generalised clause-sets are "negative monosigned CNF formula" in the language of [2], while "monosigned CNF formula" allow signs of the form $S = D_v \setminus \{\varepsilon\}$ as well as $S = \{\varepsilon\}$.

So the closest generalisation of our "clause-sets" are "monosigned CNF formulas". Considering this extension is also motived by the fact, that these formulas correspond exactly to their boolean counterpart via the natural translation. However, monosigned formulas seem to lack the good combinatorial properties which "negative monosigned formulas" have, which can be seen for example by the fact, that the boolean translation of monosigned CNF formulas need the "AMO" clauses (expressing that every (original) variable gets at most one value), making the translated formula unwieldy, while the AMO clauses are not needed for negative monosigned CNF (here we can just select some value, if an original variable gets several values, without destroying the satisfaction relation, which is not possible in the presence of literals demanding that a variable gets some fixed value).

An important point has been raised in [45], where it has been shown, that splitting on the boolean translation of generalised clause-sets can have an *exponential speed-up* over the (wide) splitting only available when splitting on the original ("negative") literals, where one considers $|D_v|$ many branches for splitting on a variable v, each branch fixing a value of v [6]. This seems to be an inherent weakness of using generalised clause-sets for SAT solving, but actually our model of generalised clause-sets allows the form of binary splitting corresponding to splits on the boolean translation: Our literals can express only "$v \neq \varepsilon$", but since we allow arbitrary variable domains, we can have a binary splitting with a domain collapse $D_v \mapsto \{\varepsilon\}$ in one branch (splitting on the positive literal "$v = \varepsilon$") and a domain restriction $D_v \mapsto D_v \setminus \{\varepsilon\}$ in the other branch (i.e., splitting on the negative literal "$v \neq \varepsilon$"): In the first branch all literals with variable v would become true or false, while in the second branch possibly the literal stays, and only the domain of v is restricted (globally).[7] Actually, if $(D_i)_{i \in I}$ is a partition of D_v then we can split into $|I|$ branches where in branch i variable v gets the new domain D_i; if for a literal (v, ε) we have $\varepsilon \notin D_i$, then the literal (and thus the clause) becomes true, while if $D_i = \{\varepsilon\}$, then the literal becomes false, and otherwise just the domain of v is restricted (globally). The splitting trees for (generalised) clause-sets with domain-splittings "$(\{\varepsilon\}, D_v \setminus \{\varepsilon\})$" correspond exactly to the splitting trees for the natural boolean translations. The price we have to pay however for this more powerful branching is, that if we stick with (generalised) clause-sets, then we cannot have (full) clause learning — if we want to use clause learning, in this way reflecting the search process within the "clause-database", then at least for recording the learned clauses we need monosigned clauses to record these binary splittings (and signed literals for more general domain splittings); this is the reason why in the upcom-

[6] the corresponding form of resolution has been studied in some depth in [36] (generalised there through the use of oracles)

[7] That is, since in the second branch we do not assign a value to variable v, we do not get rid off v in the second branch. As a consequence, we need a global domain management.

ing `OKlibrary`, a generic library for generalised SAT solving, a distinction is made between "input logic" (which might use (generalised) clause-sets) and "branching logic" (which might use an extension like monosigned clause-sets).[8)]

1.4 Overview and main results

In **Section 2** we present some preliminaries for our study of generalised clause-sets: Partial assignments for non-boolean variables, and fundamental notions and notations for graphs. Then in **Section 3** generalised clause-sets are introduced and the main operations associated with them. Autarkies and autarky systems for generalised clause-sets are reviewed in Subsection 3.5 (a useful result here is Lemma 3.1, showing how to actually find a non-trivial autarky when just given an oracle deciding whether a non-trivial autarky exists or not), while resolution for generalised clause-sets is the subject of Subsection 3.7 (in Theorem 3.2 it is proven, that a clause can be used in some resolution refutation iff it cannot be satisfied by some autarky; computation of the lean kernel via "intelligent backtracking solvers" follows). The most basic polynomial time reductions for generalised clause-sets are presented in Subsection 3.8, and finally in Subsection 3.9 the conflict graph and related notions are introduced.

Section 4 on matching autarkies for generalised clause-sets is central for this paper, and some of the main results are contained in here. First in Subsection 4.1 the notion of matching satisfiable clause-sets (introduced in [17] under the name of "matched clause-sets") is generalised in a natural way to generalised clause-sets, based on the generalised notion of deficiency. From Theorem 4.7 in Subsection 4.2, the first main result, guaranteeing always the existence of satisfying assignments "close enough" to matching satisfying assignments, we can derive in Corollary 4.10 poly-time satisfiability decision for generalised clause-sets with bounded maximal deficiency, generalising and strengthening the approach from [14] (while proving fixed parameter tractability with respect to the maximal deficiency has to wait until the next section, where further tools are provided; however we believe that the approach based on Theorem 4.7 has its own merits). Then in Subsection 4.3 matching autarkies for generalised clause-sets are introduced, and the main properties are proven. A typical result here is the generalisation of "Tarsi's Lemma" in Corollary 4.22 (every generalised minimally unsatisfiable clause-set has deficiency at least one). In Subsection 4.4 the notions of "expansion" and "surplus" are transferred from matching theory, yielding a simplified proof of FPT for SAT decision w.r.t. the parameter $\delta^*(F)$ in the boolean case (however we do not get a proof for the general case). Finally, in Subsection 4.5 we review the notion of matching autarkies introduced here, comparing it with an earlier version.

In **Section 5** the canonical translation of generalised clause-sets into boolean clause-sets is studied under the point of view of structure preservation, taking advantage of the fact, that due to the restriction to "negative literals" we do not need

[8)] This discussion shows in my opinion a major reason, why generalising boolean reasoning proved to be difficult in the past, and (boolean) SAT solvers have an edge over constraint solvers: Either we restrict ourselves to wide branching, which is inherently inefficient, or we use more powerful branching, and then we have to use a more complicated domain management than in the boolean case (where there is none), and also finding out whether a literal actually became true or false becomes considerably more complicated (while it's trivial in the boolean case). Furthermore, if we want to use learning, which seems of importance for many "real-world" problems, then we have to use more complicated literal structures, and domain and literal (occurrence) management gets further complicated.

the AMO clauses (incorporating them would destroy the structures the translation should preserve). Besides preservation of satisfiability, minimal unsatisfiability and leanness, in Subsection 5.3 we show that also a good deal of the matching structure is preserved by the translation (including for example the deficiency). Equipped with these tools, in Theorem 5.5 then we obtain FPT for SAT decision in the maximal deficiency.

In **Section 6** we turn to the study of generalised clause-sets which are minimally unsatisfiable. Considering the larger class of irredundant generalised clause-sets (no clause is implied by the others), we study the question when irredundancy is preserved by applying partial assignments. The class of irredundant clause-sets which stay irredundant for all partial assignments is characterised in Corollary 6.6 as the class of hitting clause-sets, while in Lemma 6.8 we consider the bigger class of multihitting (generalised) clause-sets and show, that they have a unique minimally unsatisfiable core (if they are unsatisfiable). In Subsection 6.3 we then discuss the process of "saturation" as introduced [16]; for generalised clause-sets we have to face a considerably more complicated situation here than in the boolean case, and thus it seems that for generalised clause-sets saturation does not play the role it does for boolean clause-sets. Without the saturation tool, proving the basic Lemma 6.15 for the characterisation of $\mathcal{MUSAT}_{\delta=1}$ needs a different trick; we use the good properties of the boolean translation. The main result of Subsection 6.4 then follows in Theorem 6.16 (the characterisation of minimally unsatisfiable clause-sets of deficiency 1), and its two corollaries (the characterisation of saturated and marginal minimally unsatisfiable clause-sets of deficiency 1). A short review on properties related to minimal variable occurrences then is given in Subsection 6.5.

Finally we present a collection of open problems in Section 7.

2 Preliminaries

We use $\mathbb{N} = \mathbb{Z}_{\geq 1}$ and $\mathbb{N}_0 = \mathbb{Z}_{\geq 0}$. For a set X by S_X the group of all bijections from X to X is denoted, while for $n \in \mathbb{N}_0$ we set $S_n := S_{\{1,\dots,n\}}$.

2.1 Variables and partial assignments

Fundamental for our considerations is the **monoid** $(\mathcal{PASS}, \circ, \emptyset)$ **of partial assignments** as introduced in Subsection 2.1 of [36], where the reader can find more information. Here we just recall the basic definitions.

The universe of variables is denoted by the infinite set $\boldsymbol{\mathcal{VA}}$, while the universe of domain elements is the infinite set $\boldsymbol{\mathcal{DOM}}$; a **(value-)domain** is a finite non-empty subset of $\mathcal{DOM}$, and for each variable $v \in \mathcal{VA}$ we denote the associated (value-)domain by $\boldsymbol{D_v}$; thus variables have fixed (value-)domains, and change of domain (for example removal of values) must be performed by renaming. For a domain D by $\boldsymbol{\mathcal{VA}_D}$ the set of all variables with domain D is denoted; to avoid running out of variables and to ease renaming, we make the assumption, that for all domains D the set $\boldsymbol{\mathcal{VA}_D}$ has the same cardinality as $\mathcal{VA}$ itself. A variable $v \in \mathcal{VA}$ is called **boolean** if $D_v = \{0, 1\}$ (and thus $\mathcal{VA}_{\{0,1\}}$ is the set of all boolean variables).

A **partial assignment** is a map φ with finite domain $\mathbf{var}(\boldsymbol{\varphi}) := \operatorname{dom}(\varphi) \subseteq \mathcal{VA}$, such that for all $v \in \operatorname{var}(\varphi)$ we have $\varphi(v) \in D_v$. The domain size of a partial assignment φ is denoted by $\boldsymbol{n(\varphi)} := |\operatorname{var}(\varphi)| \in \mathbb{N}_0$. A special partial assignment is the empty partial assignment $\emptyset$. The set of all partial assignments is denoted by $\boldsymbol{\mathcal{PASS}}$, while for some set $\mathcal{VA}' \subseteq \mathcal{VA}$ of variables we denote by $\boldsymbol{\mathcal{PASS}(\mathcal{VA}')} := \{\varphi \in \mathcal{PASS} : \mathrm{var}(\varphi) \subseteq \mathcal{VA}'\}$ the set of partial assignments for variables from $\mathcal{VA}'$ (thus $\mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ is the set of partial assignments for boolean variables). We use the notation $\langle v_1 \to \varepsilon_1, \dots, v_m \to \varepsilon_m \rangle$ to denote the partial assignment φ with $n(\varphi) = m$ and $\varphi(v_i) = \varepsilon_i$.

For two partial assignments $\varphi, \psi \in \mathcal{PASS}$ their **composition $\boldsymbol{\varphi \circ \psi}$** is defined as the partial assignment $\varphi \circ \psi$ with domain $\mathrm{var}(\varphi \circ \psi) = \mathrm{var}(\varphi) \cup \mathrm{var}(\psi)$ such that first ψ is evaluated and then φ, i.e., $(\varphi \circ \psi)(v) = \psi(v)$ if $v \in \mathrm{var}(\psi)$ while otherwise $(\varphi \circ \psi)(v) = \varphi(v)$. It is $(\mathcal{PASS}, \circ, \emptyset)$ a monoid. An alternative representation of this structure is obtained as follows: Make each D_v a ("right zero") semigroup $(D_v, \cdot)$ by defining $\varepsilon_1 \cdot \varepsilon_2 := \varepsilon_2$ for $\varepsilon_1, \varepsilon_2 \in D_v$. Adjoin an identity element "$*$" to each D_v, obtaining monoids D_v^*. Now $\mathcal{PASS}$ is isomorphic to the direct sum $\sum_{v \in \mathcal{VA}} D_v^*$ of the monoids (the sub-monoid of the direct product $\prod_{v \in \mathcal{VA}} D_v^*$ given by those elements where only finitely many components are different from $*$), where $\varphi \in \mathcal{PASS}$ corresponds to the map $\varphi^* \in \prod_{v \in \mathcal{VA}} D_v^*$ with $\varphi(v) = \varphi^*(v)$ for $v \in \mathrm{var}(\varphi)$ and $\varphi^*(v) = *$ for $v \in \mathcal{VA} \setminus \mathrm{var}(\varphi)$. This representation of partial assignments as total maps with distinguished "undefined" value $*$ actually has certain advantages over representation using partial maps, since working with total maps is often easier than working with partial maps, and we get a somewhat richer algebraic structure; however in this article we stick to the above representation of partial assignments.

2.2 Graphs and matching

A (finite) *graph* G here is a pair $G = (V, E)$ with finite vertex set $V(G) = V$ and edge set $E(G) = E \subseteq \binom{V}{2}$, where for a set M and $k \in \mathbb{N}_0$ by $\binom{M}{k}$ we denote the set of all subsets $T \subseteq M$ with $|T| = k$. So graphs here have no parallel edges and no loops. A graph G' is a *subgraph* of a graph G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$; G' is called a *partial subgraph* of G if G' is a subgraph of G and $V(G') = V(G)$. A graph G is called a *forest* if G contains no cycle. A graph G is *complete* if all distinct vertices $v, w \in V(G)$ are adjacent. G is *bipartite*, if the chromatic number of G is at most 2, while G is *complete bipartite* if G is bipartite and addition of any edge to G either destroys the graph property (i.e., creates a loop or a parallel edge) or the bipartiteness property. More generally, G is called *complete k-partite* for $k \in \mathbb{N}_0$ if the chromatic number of G is at most k, and addition of any edge to G either destroys the graph property or increases the chromatic number. It is G complete k-partite iff G is the union of at most k independent sets, such that each pair of vertices from different independent sets is adjacent (equivalently, iff the complement of G is the disjoint union of at most k cliques).

A function $f : S \to \mathbb{R}$, where S is some set system stable under union and intersection, is called *submodular* resp. *supermodular* if for all $A, B \in S$ we have $f(A \cup B) + f(A \cap B) \le f(A) + f(B)$ resp. $f(A \cup B) + f(A \cap B) \ge f(A) + f(B)$, while f is called **modular** if f is submodular and supermodular. A prototypical example for a modular function is $A \subseteq X \mapsto f(A) := |A|$, where X is some finite set. For a graph G and a vertex set $A \subseteq V(G)$ the *(closed) neighbourset* $\Gamma_G(A)$ is defined as the set of vertices adjacent to at least one element of A. The function $A \subseteq V(G) \mapsto |\Gamma(A)|$ is a prototypical example for a submodular function, while the *deficiency* $\delta(A) := |A| - |\Gamma(A)| \in \mathbb{Z}$ is a supermodular function (as the difference of a modular function and a submodular function).

A *matching* M in a graph G is a set $M \subseteq E(G)$ of edges such that two distinct elements of M are non-adjacent. A matching in G which is of maximal size is called a *maximum matching*, and the size of a maximum matching M of G is denoted by $\nu(G) := |M|$. If G is a bipartite graph with bipartition (A, B) (also called "colour classes"), then $\nu(G) = |A| - \delta^*(A) = |B| - \delta^*(B)$, where for any $S \subseteq V(G)$ we set $\delta^*(S) := \max_{S' \subseteq S} \delta(S')$.[9] A maximum matching in a bipartite graph can be computed in time $O(\sqrt{\nu(G)} \cdot |E(G)|) \leq O(\sqrt{|V(G)|} \cdot |E(G)|)$ (see Theorem 16.4 in [48]).[10]

A *maximal matching* M in a graph G is one which can not be extended (that is, there is no matching M' in G with $M \subset M'$), while the vertices *covered* by a matching M are the vertices incident to one of the edges in M. An *M-augmenting path* for a matching M in G is a path P of odd length with endpoints not covered by M and whose edges are alternately out of M and in M (so necessarily start edge and end edge (which might coincide) are out of M). A new matching M^+ is obtained by adding the edges from P to M, which were not in M, while removing the other edges of P from M; we then have $|M^+| = |M| + 1$. A matching M in a graph G has an augmenting path if and only if M is not maximum (see for example Theorem 16.1 in [48]). If G is bipartite, then deciding whether M has an augmenting path and finding one if existent can by done by breadth-first search in the directed bipartite graph naturally associated with the notion of augmenting paths (see Section 16.3 in [48]), and so this process takes time $O(|E(G)|$. Using this process to construct a maximum matching, starting with the empty matching, would take time $O(\nu(G) \cdot |E(G)|)$ which is worse than the bound given above, however if a matching M is given of "reasonable size", then the time $O((\nu(G) - |M|) \cdot |E(G)|)$ it takes to construct a maximum matching, starting with M, might be better.

A *vertex cover* of a graph G is a set $T \subseteq V(G)$ of vertices such that every edge of G is incident with (at least) one of the vertices in T; a vertex cover of minimal size is called a *minimum vertex cover*, the size of a minimum vertex cover of G is denoted by $\tau(G)$. For bipartite graphs G we have $\tau(G) = \nu(G)$, and given a maximum matching of G, in time $O(|E(G)|)$ a minimum vertex cover can be computed (see Theorem 16.6 in [48]). A witness for $\nu(G) \geq k$ can always be given by a matching M in G with $|M| \geq k$, while for bipartite graphs G a witness for $\nu(G) \leq k$ can always be given by a vertex cover T with $|T| \leq k$. For our applications, witnesses using the notion of deficiency are more useful (they will yield autarkies in Lemmas 4.25 and 4.32): Given a bipartite graph G with bipartition (A, B) and vertex cover T of G, the set $A' := A \setminus T$ has deficiency $\delta(A') \geq |A| - |T|$ (while given $A' \subseteq A$, the vertex cover $\Gamma(A') \cup (A \setminus A')$ has size $|A| - \delta(A')$).

3 Generalised (multi-)clause-sets

In this section we review the notion of generalised multi-clause-sets and the basic facts about them regarding autarkies and resolution.

In Subsection 3.1 we introduce the notion of "generalised multi-clause-sets" and "generalised clause-sets", while in Subsection 3.2 (partial) assignments and their operation on (multi-)clause-sets is discussed. This introduction into "syntax and

[9] See for example Theorem 22.2 in [48], where the notion of "transversals" or "systems of distinct representatives" of a set system is used (not to be mixed up with "transversals" in hypergraphs), and where the set system is $(\Gamma_G(\{a\}))_{a \in A}$ resp. $(\Gamma_G(\{b\}))_{b \in B}$.

[10] We won't dwell here on the details of graph representations; however when stating complexity results for (multi-)clause-sets we will be more precise.

semantics of generalised clause-sets" is completed in Subsection 3.4 with the discussion of various operations on (multi-)clause-sets F regarding their variable structure (that is, disregarding the different "polarities", i.e., disregarding the literal structure).

Of central importance to our work is Subsection 3.5, where the notion of *autarkies* (special partial assignments, which satisfy parts of the formula, and leave the rest untouched) and *autarky systems* (allowing to tailor the notion of autarkies for special purposes) for multi-clause-sets are introduced. In Subsection 3.7 then resolution for generalised clause-sets is discussed, while in Section 3.8 we give the most basic reductions for generalised clause-sets. Finally in Subsection 3.9 some very basic notions regarding the conflict structure of generalised clause-sets are introduced.

For more background information, see [36, 30] for a general, axiomatic framework for "generalised satisfiability problems", while in Subsection 2.3 of [36] generalised clause-sets are discussed, and in Section 2 of [33] boolean multi-clause-sets are considered (see also [32] for more information). In this paper, when we speak of "clause-sets" then we always mean "generalised clause-sets", while clause-sets in the "traditional" sense are always qualified as "boolean clause-sets"; however in lemmas, corollaries and theorems we often speak of "generalised clause-sets" to ease independent access.

3.1 Syntax: The notion of "multi-clause-sets"

A **literal** is a pair (v, ε) of a variable $v \in \mathcal{VA}$ and a value $\varepsilon \in D_v$; we write $\mathbf{var}(\boldsymbol{v}, \boldsymbol{\varepsilon}) := v$ and $\mathbf{val}(\boldsymbol{v}, \boldsymbol{\varepsilon}) := \varepsilon$. The set of all literals is denoted by $\boldsymbol{\mathcal{LIT}}$, and for any $\mathcal{VA}' \subseteq \mathcal{VA}$ we write $\boldsymbol{\mathcal{LIT}(\mathcal{VA}')} := \{x \in \mathcal{LIT} : \mathrm{var}(x) \in \mathcal{VA}'\}$ for the set of literals with variables from $\mathcal{VA}'$ (thus $\mathcal{LIT}(\mathcal{VA}_{\{0,1\}})$ is the set of boolean literals). For a partial assignment $\varphi \in \mathcal{PASS}$ and a literal (v, ε) with $v \in \mathrm{var}(\varphi)$ we set $\varphi((v, \varepsilon)) = 1$ if $\varphi(v) \neq \varepsilon$, while we set $\varphi((v, \varepsilon)) = 0$ if $\varphi(v) = \varepsilon$; thus a literal (v, ε) has the meaning "v shall *not* get value ε". Accordingly a literal (v, ε) is often denoted by "$v \neq \varepsilon$".

A **clause** C is a finite set of literals not containing "clashing literals", that is for literals $x, y \in C$ with $x \neq y$ we have $\mathrm{var}(x) \neq \mathrm{var}(y)$. The set of all clauses is denoted by $\boldsymbol{\mathcal{CL}}$. For a clause C we set $\mathrm{var}(C) := \{\mathrm{var}(x) : x \in C\}$, and for a set $\mathcal{VA}' \subseteq \mathcal{VA}$ we write $\boldsymbol{\mathcal{CL}(\mathcal{VA}')} := \{C \in \mathcal{CL} : \mathrm{var}(C) \subseteq \mathcal{VA}'\}$ for the set of clauses with variables from $\mathcal{VA}'$ (thus $\mathcal{CL}(\mathcal{VA}_{\{0,1\}})$ is the set of boolean clauses). The empty clause is denoted by $\bot \in \mathcal{CL}$.

Given a clause C, we obtain the corresponding partial assignment $\boldsymbol{\varphi_C} \in \mathcal{PASS}$ as the partial assignment φ with $\mathrm{var}(\varphi) = \mathrm{var}(C)$ and $\varphi(v) = \varepsilon$ for $(v, \varepsilon) \in C$; on the other hand, given a partial assignment φ, we obtain the corresponding clause $\boldsymbol{C_\varphi} \in \mathcal{CL}$ as the clause C with $\mathrm{var}(C) = \mathrm{var}(\varphi)$ such that for $\varphi(v) = \varepsilon$ we have $(v, \varepsilon) \in C$. Using the representation of maps as ordered pairs of arguments and values, actually $\varphi_C = C$ and $C_\varphi = \varphi$ (and thus $\mathcal{CL} = \mathcal{PASS}$); explicitely said, a clause corresponds to the partial assignment which sets exactly the literals in the clause to false.[11)]

[11)]The motivation is, that with a partial assignment φ we restrict the search space, and in case the partial assignment φ is inconsistent with the clause-set F, then the clause C_φ can be learned (i.e., follows from F).

A (finite) **multi-clause-set** is a map $F : \mathcal{CL} \to \mathbb{N}_0$ (assigning to each clause its number of occurrences) such that only for finitely many $C \in \mathcal{CL}$ we have $F(C) \neq 0$, while a (finite)**clause-sets** is a finite subset of $\mathcal{CL}$. Clause-sets F can be implicitly converted to multi-clause-sets by setting $F(C) := 1$ for $C \in F$ and $F(C) := 0$ otherwise, while for a multi-clause-set F the **underlying clause-set $\hat{\mathrm{t}}(F)$** is defined as $\hat{\mathrm{t}}(F) = \{C \in \mathcal{CL} : F(C) \neq 0\}$, and this conversion is only performed if necessary to apply a definition. We have $\boldsymbol{C \in F}$ for a (multi-)clause-set F iff $F(C) > 0$. For a (multi-)clause-set F we set $\mathrm{var}(F) := \bigcup\{\mathrm{var}(C) : C \in F\}$. For a (multi-)clause-set F and a variable $v \in \mathcal{VA}$ we define $\mathbf{val}_{\boldsymbol{v}}\boldsymbol{(F)} := \{\varepsilon \in D_v \mid \exists\, C \in F : (v, \varepsilon) \in C\}$. We have $\mathrm{var}(F) = \{v \in \mathcal{VA} : \mathrm{val}_v(F) \neq \emptyset\}$. Finally the empty clause-set as well as the empty multi-clause-set is denoted by $\top$.

A clause-set F has a **uniform domain**, if $\forall\, v, w \in \mathrm{var}(F) : D_v = D_w$ holds. Boolean clause-sets have uniform domain, and every (generalised) clause-set has a "domain-uniformisation" by using the union of all relevant domains and adding unit-clauses to forbid unwanted domain elements; see Subsection 3.3 for details.

We use the following complexity measures for multi-clause-sets F of clauses:

1. $\boldsymbol{\#_{(v,\varepsilon)}(F)} := \sum_{C \in F, (v,\varepsilon) \in C} F(C) \in \mathbb{N}_0$ measures the number of occurrences of a literal;
2. $\boldsymbol{\#_v(F)} := \sum_{\varepsilon \in D_v} \#_{(v,\varepsilon)}(F) = \sum_{C \in F, v \in \mathrm{var}(C)} F(C) \in \mathbb{N}_0$ measures the number of occurrences of a variable;
3. $\boldsymbol{s_{(v,\varepsilon)}(F)} := \sum_{\varepsilon' \in D_v \setminus \{\varepsilon\}} \#_{(v,\varepsilon)}(F) = \#_v(F) - \#_{(v,\varepsilon)}(F) \in \mathbb{N}_0$ measures the number of occurrences of literals with variable v and value different from ε (this is the number of satisfied clauses when assigning value ε to v; see below);
4. $\boldsymbol{n(F)} := |\mathrm{var}(F)| \in \mathbb{N}_0$ measures the number of variables;
5. $\boldsymbol{c(F)} := \sum_{C \in F} F(C) \in \mathbb{N}_0$ measures the number of clauses;
6. $\boldsymbol{\ell(F)} := \sum_{C \in F} F(C) \cdot |C| = \sum_{v \in \mathrm{var}(F)} \#_v(F) \in \mathbb{N}_0$ measures the number of literal occurrences.

And for multi-clause-sets F_1, F_2 we use the following operations and relations:

1. the multi-clause-set $\boldsymbol{F_1 + F_2}$ is defined by $(F_1 + F_2)(C) := F_1(C) + F_2(C)$ for clauses C;
2. the multi-clause-set $\boldsymbol{F_1 \cup F_2}$ resp. $\boldsymbol{F_1 \cap F_2}$ is given by setting $(F_1 \cup F_2)(C) := \max(F_1(C), F_2(C))$ resp. $(F_1 \cap F_2)(C) := \min(F_1(C), F_2(C))$ for clauses C; if F_1, F_2 are clause-sets, then these operations coincide with the ordinary set operations;
3. if F_2 is a clause-set, then the multi-clause-set $\boldsymbol{F_1 \setminus F_2}$ is defined by setting $(F_1 \setminus F_2)(C) := 0$ for $C \in F_2$, while otherwise $(F_1 \setminus F_2)(C) := F_1(C)$; if also F_1 is a clause-set, then $F_1 \setminus F_2$ is the ordinary set operation;
4. the relation $\boldsymbol{F_1 \leq F_2}$ holds if for all clauses C we have $F_1(C) \leq F_2(C)$; we use $\boldsymbol{F' \lneq F}$ for $F' \leq F \wedge F' \neq F$; if F_1, F_2 are clause-sets, then $F_1 \leq F_2 \Leftrightarrow F_1 \subseteq F_2$;
5. F_1 is called a **sub-multi-clause-set** of F_2 if $F_1 \leq F_2$ holds, while F_1 is called an **induced sub-multi-clause-set** of F_2 if $F_1 \leq F_2$ and $\forall\, C \in F_1 : F_1(C) = F_2(C)$ holds; every sub-clause-set of a clause-set is induced;
6. if F_2 is a sub-multi-clause-set of F_1, then the multi-clause-set $\boldsymbol{F_1 - F_2}$ is defined via $(F_1 - F_2)(C) := F_1(C) - F_2(C)$ for clauses C.

The set of all multi-clause-sets is denoted by $\boldsymbol{\mathcal{MCLS}}$, the set of all clause-sets by $\boldsymbol{\mathcal{CLS}}$, while for a set $\mathcal{VA}' \subseteq \mathcal{VA}$ of variables we use $\boldsymbol{\mathcal{MCLS}(\mathcal{VA}')} := \{F \in \mathcal{MCLS} : \mathrm{var}(F) \subseteq \mathcal{VA}'\}$ and $\boldsymbol{\mathcal{CLS}(\mathcal{VA}')} := \{F \in \mathcal{CLS} : \mathrm{var}(F) \subseteq \mathcal{VA}'\}$ (thus $\mathcal{MCLS}(\mathcal{VA}_{\{0,1\}})$ is the set of boolean multi-clause-sets, and $\mathcal{CLS}(\mathcal{VA}_{\{0,1\}})$ is the set of boolean clause-sets). If $\mathcal{C}$ is a set of multi-clause-sets and $f : \mathcal{C} \to \mathbb{R}$, then by $\boldsymbol{\mathcal{C}_{f \le b}}$ for some $b \in \mathbb{R}$ we denote the set of all $F \in \mathcal{C}$ with $f(F) \le b$; analogously we define $\mathcal{C}_{f=b}$, $\mathcal{C}_{f \ge b}$ and so on. A special function usable here is $\mathrm{sat} : \mathcal{CLS} \to \{0,1\}$ with $\mathrm{sat}(F) = 1 \Leftrightarrow F \in \mathcal{SAT}$ (that is, sat is the characteristic function of the set of satisfiable clause-sets defined below); we can combine several such indices, and for typographical reasons we may use then for example $\mathcal{MCLS}_{f \le b}^{g \le b'}$.

3.2 Semantics: The operation of partial assignments

Now we define the operation $* : \mathcal{PASS} \times \mathcal{MCLS} \to \mathcal{MCLS}$ of $\mathcal{PASS}$ on multi-clause-sets, and the (derived) operation $* : \mathcal{PASS} \times \mathcal{CLS} \to \mathcal{CLS}$ on clause-sets, which in both cases have the meaning of substituting values for variables and carrying out the resulting simplifications (viewing a clause as a disjunction of its literals, and a (multi-)clause-set as a conjunction of its clauses), with the only difference that in the case of clause-sets contractions in the result are carried out (distinct clauses can become equal after a substitution). The case of clause-sets is reduced to the case of multi-clause-sets, using the explicit transformation $\check{\mathrm{t}} : \mathcal{CLS} \to \mathcal{MCLS}$ of clause-sets into multi-clause-sets. For $F \in \mathcal{MCLS}$ and $\varphi \in \mathcal{PASS}$ we define $\boldsymbol{\varphi * F} \in \mathcal{MCLS}$ by

$$(\varphi * F)(C) = \sum_{\substack{C' \in \mathcal{CL} \\ \varphi * \{C'\} = \{C\}}} F(C'),$$

for $C \in \mathcal{CL}$, where for a clause C we set $\varphi * \{C\} := \top \in \mathcal{CLS}$ if there exists a literal $x \in C$ with $\varphi(x) = 1$, while otherwise we set $\varphi * \{C\} := \{C \setminus C_\varphi\} \in \mathcal{CLS}$ (i.e., we remove the falsified literals from C). And for $F \in \mathcal{CLS}$ we define $\varphi * F \in \mathcal{CLS}$ as

$$\varphi * F := \hat{\mathrm{t}}(\varphi * \check{\mathrm{t}}(F)).$$

We have here (where F is a clause-*set*) $\varphi * F = \bigcup_{C \in F} \varphi * \{C\}$. The effect on the basic measures of applying a partial assignment $\langle v \to \varepsilon \rangle$ to $F \in \mathcal{MCLS}$ with $v \in \mathrm{var}(F)$ is given by

$$\begin{aligned} n(\langle v \to \varepsilon \rangle * F) &\le n(F) - 1 \\ c(\langle v \to \varepsilon \rangle * F) &= c(F) - s_{(v,\varepsilon)}(F). \end{aligned}$$

A clause-set $F \in \mathcal{CLS}$ is **satisfiable** if there exists a partial assignment $\varphi \in \mathcal{PASS}$ with $\varphi * F = \top$, while otherwise F is **unsatisfiable**; the set of all satisfiable clause-sets is denoted by $\boldsymbol{\mathcal{SAT}}$, the set of all unsatisfiable clause-sets by $\boldsymbol{\mathcal{USAT}}$. A multi-clause-set $F \in \mathcal{MCLS}$ is called **minimally unsatisfiable** if F is unsatisfiable, but every $F' \lneq F$ is satisfiable; obviously if F is minimally unsatisfiable, then F actually is a clause-set. The set of all minimally unsatisfiable clause-sets is denoted by $\boldsymbol{\mathcal{MUSAT}}$.

It is useful to have some notations for the set of satisfying assignments ("models") as well as for the set of falsifying assignments. For a finite $V \subseteq \mathcal{VA}$ let $\boldsymbol{\mathcal{PASS}(V)}$ be the set of $\varphi \in \mathcal{PASS}$ with $\mathrm{var}(\varphi) = V$. Note that we have

$$|\mathcal{PASS}(V)| = \prod_{v \in V} |D_v|.$$

Now for a clause-set $F \in \mathcal{MCLS}$ and for a finite set V of variables with $\mathrm{var}(F) \subseteq V$ let $\mathbf{mod}_{\boldsymbol{V}}(\boldsymbol{F})$ be the set of $\varphi \in \mathcal{PASS}(V)$ with $\varphi * F = \top$, while $\overline{\mathbf{mod}}_{\boldsymbol{V}}(\boldsymbol{F})$ is the set of $\varphi \in \mathcal{PASS}(V)$ with $\bot \in \varphi * F$. Thus F is satisfiable iff $\mathrm{mod}_V(F) \neq \emptyset$; and for any clause C with $\mathrm{var}(C) \subseteq V$ we have

$$|\overline{\mathrm{mod}}_V(\{C\})| = |\mathcal{PASS}(V \setminus \mathrm{var}(C))| = \prod_{v \in V \setminus \mathrm{var}(C)} |D_v|,$$

that is, the falsifying assignments for a clause C are uniquely determined on variables from C and arbitrary elsewhere. At this point it might be useful to point out, that for (multi-)clause-sets typically *falsifying assignments* are significantly easier to handle than satisfying assignments ("those elusive idols"). Obviously $\mathrm{mod}_V(F) \cap \overline{\mathrm{mod}}_V(F) = \emptyset$ and $\mathrm{mod}_V(F) \cup \overline{\mathrm{mod}}_V(F) = \mathcal{PASS}(V)$. By definition we have

$$\overline{\mathrm{mod}}_V(F) = \bigcup_{C \in F} \overline{\mathrm{mod}}_V(\{C\}).$$

For clause-sets F_1, F_2 we write $\boldsymbol{F_1 \models F_2}$ ("F_1 implies F_2") if for all $\varphi \in \mathcal{PASS}$ with $\varphi * F_1 = \top$ we have $\varphi * F_2 = \top$ as well, and for clauses C we write $F \models C$ instead of $F \models \{C\}$. Trivially F is unsatisfiable iff $F \models \bot$. Note that $F_1 \models F_2$ holds iff for $V := \mathrm{var}(F_1) \cup \mathrm{var}(F_2)$ we have $\overline{\mathrm{mod}}_V(F_2) \subseteq \overline{\mathrm{mod}}_V(F_1)$. We call F_1, F_2 **equivalent** if $F_1 \models F_2$ and $F_2 \models F_1$.

The basic laws for the operation of partial assignments on multi-clause-sets are as follows, using $F, F_1, F_2 \in \mathcal{MCLS}$ and $\varphi, \psi \in \mathcal{PASS}$:

$$\emptyset * F = F$$
$$\varphi * \top = \top$$
$$(\varphi \circ \psi) * F = \varphi * (\psi * F)$$
$$\varphi * (F_1 + F_2) = \varphi * F_1 + \varphi * F_2.$$

These four laws hold also for the operation of partial assignments on clause-sets. If $F_1, F_2 \in \mathcal{CLS}$, then we have

$$\varphi * (F_1 \cup F_2) = \varphi * F_1 \cup \varphi * F_2$$

(but this does not hold for multi-clause-sets in general). Furthermore for a multi-clause-set F and a clause-set F' we have $\varphi * (F \setminus F') \geq (\varphi * F) \setminus (\varphi * F')$.

3.3 Renaming variables

Consider a multi-clause-set F and variables $v, w \in \mathcal{VA}$ (which might be equal) together with $h : D_v \to D_w$ such that in case of $v \neq w$ we have $w \notin \mathrm{var}(F)$. Then **replacing v by w using h in F** results in the multi-clause-set F' where every occurrence of a literal (v, ε) is replaced by the literal $(w, h(\varepsilon))$. The map h here is called the **value transfer**; if $D_v \subseteq D_w$ and h is unspecified, then the canonical injection is used.

Similarly, replacing v by w using h in a partial assignment φ, where in case of $v \neq w$ we have $w \notin \mathrm{var}(\varphi)$, results in a partial assignment φ' with $\mathrm{dom}(\varphi') = (\mathrm{dom}(\varphi) \setminus \{v\}) \cup \{w\}$ such that $\varphi'(u) = \varphi(u)$ for $u \in \mathrm{dom}(\varphi') \setminus \{w\}$, while $\varphi'(w) = h(\varphi(v))$. Here the value transfer needs to be specified only for the special value $\varphi(v)$. If $v = w$, then we just speak of **flipping v to ε in φ** for $\varepsilon = h(\varphi(v))$.

The replacement of v by w using h in F is **injective**, if for literals $(v, \varepsilon), (v, \varepsilon')$ occurring in F with $\varepsilon \neq \varepsilon'$ we have $h(\varepsilon) \neq h(\varepsilon')$. If $|D_w| \geq \#_v(F)$, then there is always some $h : D_v \to D_w$ such that replacing v by w in F using h is injective. For every injective h, replacing v by w in F using h is injective. Note that injective replacements alter the "meaning" (the set of models modulo isomorphism) exactly in the case where a non-pure variable (a variable such that all values occur in F; see Subsection 3.8) is rendered a pure variable by using a domain D_w with $|D_w| > \#_v(F)$. Special injective replacements are **renamings**, where h is a bijection from D_v to D_w. If we have a renaming of v by w using h in F, resulting in F', then we also have the renaming of w to v using h^{-1} in F', resulting in F. So the satisfying assignments for F' here are exactly the satisfying assignments for F where v is replaced by w using h. If several variables are renamed simultaneously, then we require that the same result is obtained by renaming single variables one after another (in some order).

For a multi-clause-set F a **domain-uniformisation** of F is F in case of $\mathrm{var}(F) = \emptyset$, while otherwise we consider the domain $D := \bigcup_{v \in \mathrm{var}(F)} D_v$, rename all variables v in F to v' with domain D, and add all unit clauses $\{v' \neq \varepsilon\}$ for $\varepsilon \in D \setminus D_v$.

3.4 Three operations of sets of variables on multi-clause-sets

Finally we consider various operations with sets of variables. The operation $\boldsymbol{V * F}$ is defined for finite $V \subseteq \mathcal{VA}$ and $F \in \mathcal{MCLS}$ via

$$(V * F)(C) := \sum_{\substack{C' \in \mathcal{CL} \\ V * C' = C}} F(C'),$$

where for a clause C we set $V * C := \{x \in C : \mathrm{var}(x) \notin V\} \in \mathcal{CL}$. That is, $V * F$ is obtained from F by crossing out all literal occurrences x with $\mathrm{var}(x) \in V$. Two basic properties are

$$\begin{aligned} \mathrm{var}(V * F) &= \mathrm{var}(F) \setminus V \\ c(V * F) &= c(F). \end{aligned}$$

The operation $\boldsymbol{V * F}$ for $F \in \mathcal{CLS}$ is defined by

$$V * F := \hat{\mathrm{t}}(V * \check{\mathrm{t}}(F)) \in \mathcal{CLS}.$$

We have here $V * F = \{V * C : C \in F\}$. The basic laws for $F, F_1, F_2 \in \mathcal{MCLS}$ and finite $V, V' \subseteq \mathcal{VA}$ are

$$\begin{gathered} \emptyset * F = F \\ V * \top = \top \\ (V \cup V') * F = V * (V' * F) \\ V * (F_1 + F_2) = V * F_1 + V * F_2. \end{gathered}$$

Again these four laws also hold for the operation of sets of variables on clause-sets. If $F_1, F_2 \in \mathcal{CLS}$, then we have

$$V * (F_1 \cup F_2) = V * F_1 \cup V * F_2$$

(again this does not hold for multi-clause-sets in general).

We conclude with different forms of selecting parts of a multi-clause-set. By $\boldsymbol{F_V}$ we denote the induced sub-multi-clause-set of F with $C \in F_V \Leftrightarrow \mathrm{var}(C) \cap V \neq \emptyset$; in other words, $F_V = F \setminus \{C \in F : \mathrm{var}(C) \cap V = \emptyset\}$. Basic properties are:

1. $F_{\emptyset} = \top$ and $F_{\mathrm{var}(F)} = F \setminus \{\bot\}$.
2. If $V_1 \subseteq V_2$, then F_{V_1} is an induced sub-multi-clause-set of F_{V_2}.
3. $F_{V_1 \cup V_2} = F_{V_1} \cup F_{V_2}$.
4. For $v \in \mathcal{VA}$ we have $c(F_{\{v\}}) = \#_v(F)$.

Finally

$$\boldsymbol{F[V]} := (\mathrm{var}(F) \setminus V) * F_V = ((\mathrm{var}(F) \setminus V) * F) \setminus \{\bot\} \in \mathcal{MCLS},$$

which, in analogy to the same process for hypergraphs (using usually also the same notation, see for example [13]), can be considered as the "restriction of F to V". Basis properties are

1. $F[\emptyset] = \top$ and $F[\mathrm{var}(F)] = F \setminus \{\bot\}$.
2. $c(F[V]) = c(F_V)$, $\mathrm{var}(F[V]) \subseteq \mathrm{var}(F_V)$.
3. $\mathrm{var}(F[V]) = V$ for $V \subseteq \mathrm{var}(F)$.

To summarise: We obtain $V * F$ from F by keeping all clauses but removing those literals x from them with $\mathrm{var}(x) \in V$, while we obtain F_V from F by removing those clauses C from F with $\mathrm{var}(C) \cap V = \emptyset$ (while keeping all clauses intact); finally $F[V]$ is obtained from F by first constructing F_V, and then crossing out all literal occurrences for literals x where there exists a clause $C \in F$ with $\mathrm{var}(C) \cap V = \emptyset$ and $\mathrm{var}(x) \in \mathrm{var}(C)$.

$F[V]$ is the formula derived from F when we want to consider total assignments relative to the variable set V, and is basic for the theory of autarkies reviewed in the subsequent subsection, while $V*F$ and F_V are fundamental constructions. As an example for these operations consider boolean variables a, b, c (the domains of variables do not matter here), and let $C_1 := \{(a,0),(b,1),(c,0)\}$, $C_2 := \{(a,0),(b,0),(c,1)\}$, $C_3 := \{(a,1),(b,0),(c,1)\}$ and $C_4 := \{(b,1),(c,1)\}$, and finally $F := \sum_{i=1}^{4}\{C_i\}$ (F corresponds to the CNF $(a \vee \overline{b} \vee c) \wedge (a \vee b \vee \overline{c}) \wedge (\overline{a} \vee b \vee \overline{c}) \wedge (\overline{b} \vee \overline{c})$). Now we have $F_{\{a\}} = \sum_{i=1}^{3}\{C_i\}$, $\{a\} * F = \{\{(b,1),(c,0)\}\} + 2 \cdot \{\{(b,0),(c,1)\}\} + \{\{(b,1),(c,1)\}\}$, while $F[\{a\}] = 2 \cdot \{\{(a,0)\}\} + \{\{(a,1)\}\}$.

3.5 Autarkies for generalised multi-clause-sets

Now we review the general properties of autarkies and autarky systems for generalised multi-clause-sets. See Section 3 in [30] for a general theory of autarkies and autarky systems, while in Section 4 of [30] autarky systems for generalised clause-sets have been discussed (easily generalised to autarky systems for generalised multi-clause-sets). General properties of autarkies for boolean clause-sets are thoroughly investigated in [29], Section 3, while autarky systems for boolean clause-sets have been introduced in [31] (see Sections 4 and 8 for the general theory).

A partial assignment $\varphi \in \mathcal{PASS}$ is an **autarky** for $F \in \mathcal{MCLS}$ if one (and thus all) of the following four equivalent conditions is fulfilled:

1. for all clauses $C \in F$ we have $\mathrm{var}(\varphi) \cap \mathrm{var}(C) \neq \emptyset \Rightarrow \varphi * \{C\} = \top$;
2. $\forall\, F' \leq F : \varphi * F' \leq F'$;
3. φ is a satisfying assignment for $F_{\mathrm{var}(\varphi)}$;

4. φ is a satisfying assignment for $F[\mathrm{var}(\varphi)]$.

Obviously, φ is an autarky for F iff φ is an autarky for $F \setminus \{\bot\}$ iff φ is an autarky for the underlying clause-set. The set of all autarkies for F is denoted by $\mathbf{Auk(}\boldsymbol{F}\mathbf{)}$; it is $\mathrm{Auk}(F)$ a sub-monoid of $\mathcal{PASS}$, containing all satisfying assignments for F in case F is satisfiable, and $\mathrm{Auk}(F) = \mathrm{Auk}(\hat{\mathrm{t}}(F))$. If $F' \le F$, then $\mathrm{Auk}(F) \subseteq \mathrm{Auk}(F')$, and for finite $V \subseteq \mathcal{VA}$ we have $\{\varphi \in \mathrm{Auk}(V * F) : \mathrm{var}(\varphi) \cap V = \emptyset\} = \{\varphi \in \mathrm{Auk}(F) : \mathrm{var}(\varphi) \cap V = \emptyset\}$. Furthermore we have $\mathrm{Auk}(F_1 + F_2) = \mathrm{Auk}(F_1) \cap \mathrm{Auk}(F_2)$. If $\varphi \in \mathrm{Auk}(F)$ and $\psi \in \mathrm{Auk}(\varphi * F)$, then $\psi \circ \varphi \in \mathrm{Auk}(F)$. An autarky $\varphi \in \mathrm{Auk}(F)$ is called **non-trivial** if $\mathrm{var}(\varphi) \cap \mathrm{var}(F) \neq \emptyset$ holds. F is called **lean**, if F has no non-trivial autarky; the set of all lean multi-clause-sets is denoted by $\boldsymbol{\mathcal{LEAN}}$. A sum of lean multi-clause-sets again is lean. If F is lean, so is $V * F$ for $V \subseteq \mathcal{VA}$. Trivially every F having only variables with trivial domain (i.e., one-element domains) is lean.

An **autarky reduction** is a reduction $F \mapsto \varphi * F$ for some non-trivial autarky φ for F (note that $\varphi * F$ is satisfiability equivalent to F). Autarky reduction is terminating and confluent (generalising Lemma 4.1 in [31], a special case of Lemma 3.7 in [30]), and thus the result of iterated autarky reductions until no further reductions are possible is uniquely determined; we denote it by $\mathbf{N_{Auk}(}\boldsymbol{F}\mathbf{)} \le F$. It is $\mathbf{N_a} := \mathrm{N}_{\mathrm{Auk}}$ a "kernel operator", that is, $\mathrm{N_a}(F) \le F$, $\mathrm{N_a}(\mathrm{N_a}(F)) = \mathrm{N_a}(F)$, and $F_1 \le F_2 \Rightarrow \mathrm{N_a}(F_1) \le \mathrm{N_a}(F_2)$; furthermore $\mathrm{N_a}(F)$ is satisfiability equivalent to F, and $\mathrm{N_a}(F) = \top$ iff $F \in \mathcal{SAT}$. We have $\mathrm{N_a}(F) \in \mathcal{LEAN}$, and $\mathrm{N_a}(F)$ is called the **lean kernel** of F; F is lean iff $\mathrm{N_a}(F) = F$. There exists an autarky $\varphi \in \mathrm{Auk}(F)$ with $\mathrm{N_a}(F) = \varphi * F$ (while for all $\varphi \in \mathrm{Auk}(F)$ we have $\mathrm{N_a}(F) \le \varphi * F$). It is $\mathrm{N_a}(F)$ the largest lean sub-multi-clause-set of F.

An **autark sub-multi-clause-set** F' of F is an induced sub-multi-clause-set of F, such that there exists an autarky $\varphi \in \mathrm{Auk}(F)$ so that for $C \in F$ we have $C \in F'$ iff $\varphi * \{C\} = \top$ (note that in this case we have $F' = F_{\mathrm{var}(\varphi)}$). The set of autark sub-multi-clause-sets of F is closed under union, and contains the smallest element $\top$ and the largest element $F \setminus \mathrm{N_a}(F)$. It is $F' \le F$ an autark sub-multi-clause-set of F iff there is $V \subseteq \mathrm{var}(F)$ with $F_V = F'$ and $F[V] \in \mathcal{SAT}$.

The relation between the lean kernel of F and the largest autark sub-multi-clause-set of F can be summarised as follows: For $F \in \mathcal{MCLS}$ there exist induced sub-multi-clause-sets $F_1, F_2 \le F$ with $F_1 + F_2 = F$, such that F_1 is lean, while $\mathrm{var}(F_1) * F_2$ is satisfiable; in this decomposition F_1, F_2 are uniquely determined, namely $F_1 = \mathrm{N_a}(F)$ is the largest lean sub-multi-clause-set (the lean kernel), while F_2 is the largest autark sub-multi-clause-set.

For an example consider variables a, b, c, d with $D_a = D_b = \{0, 1, 2\}$ and $D_c = D_d = \{0, 1\}$, and consider the clause-set

$$\begin{aligned} F &:= F_1 \cup F_2 \\ F_1 &:= \{\{a \neq 0, b \neq 0\}, \{a \neq 0, b \neq 1\}, \{a \neq 0, b \neq 2\}, \{a \neq 1\}, \{a \neq 2\}\} \\ F_2 &:= \{\{a \neq 0, c \neq 0, d \neq 1\}, \{b \neq 0, c \neq 1, d \neq 0\}\}. \end{aligned}$$

To see whether there is an autarky for F invoking exactly one variable we check satisfiability of $F[\{v\}]$ for $v \in \{a, b, c, d\}$; we see that all these clause-sets are unsatisfiable (e.g., $F[\{c\}] = \{\{c \neq 0\}, \{c \neq 1\}\}$), and so the smallest non-trivial autarky for F (if there is any) must involve at least two variables. Now $F[\{c, d\}] = \{\{c \neq 0, d \neq 1\}, \{c \neq 1, d \neq 0\}\} \in \mathcal{SAT}$, and thus the two partial assignments $\langle c, d \to 0\rangle, \langle c, d \to 1\rangle$ are autarkies for F; applying one of them yields F_1, which is lean (F_1 actually is minimally unsatisfiable), and thus F_1 is the lean kernel of F, while F_2 is the largest autark sub-clause-set of F.

3.6 Autarky systems

After having reviewed the general facts for autarkies for generalised multi-clause-sets, we now consider "autarky systems". The motivation for doing so is, that instead of (computationally infeasible) general autarkies we want to consider restricted autarkies, and under mild assumptions on these restricted autarkies all the above facts carry over (in generalised form). The monoid $(\mathcal{PASS}, \circ, \emptyset)$ together with the partial order $(\mathcal{MCLS}, \leq, \top)$ with least element and together with the operation $*$ of $\mathcal{PASS}$ on $\mathcal{MCLS}$ fulfils all the axioms required in Section 3 of [30], and thus all the general results there on autarky systems hold here.

An **autarky system** for generalised multi-clause-sets is a map $\mathcal{A}$, which assigns to every $F \in \mathcal{MCLS}$ a sub-monoid $\mathcal{A}(F)$ of $\mathrm{Auk}(F)$, such that for $F_1 \leq F_2$ we have $\mathcal{A}(F_2) \subseteq \mathcal{A}(F_1)$. The elements of $\mathcal{A}(F)$ are called **$\mathcal{A}$-autarkies** for F. Further possible restrictions on $\mathcal{A}$ are expressed by the following notions:

1. $\mathcal{A}$ is **iterative**, if for $\varphi \in \mathcal{A}(F)$ and $\psi \in \mathcal{A}(\varphi * F)$ we always have $\psi \circ \varphi \in \mathcal{A}(F)$.
2. $\mathcal{A}$ is called **standardised**, if for a partial assignment $\varphi \in \mathcal{PASS}$ we have $\varphi \in \mathcal{A}(F)$ iff $\varphi \mid \mathrm{var}(F) \in \mathcal{A}(F)$ (where $\varphi \mid \mathrm{var}(F)$ is the restriction of the map φ to the domain $\mathrm{var}(\varphi) \cap \mathrm{var}(F)$). (Remark: Thus for a standardised autarky system $\mathcal{A}$ all partial assignments φ with $\mathrm{var}(\varphi) \cap \mathrm{var}(F) = \emptyset$ are (trivial) $\mathcal{A}$-autarkies for F. In [30] only the direction "$\varphi \in \mathcal{A}(F) \Rightarrow \varphi \mid \mathrm{var}(F) \in \mathcal{A}(F)$ is required, but now it seems more systematic to me to require also the other direction.)
3. $\mathcal{A}$ is **$\perp$-invariant**, if always $\mathcal{A}(F) = \mathcal{A}(F + \{\perp\})$ holds (in [30, 31] this was called "normal").
4. $\mathcal{A}$ is **stable under variable elimination**, if for finite $V \subseteq \mathcal{VA}$ we always have $\{\varphi \in \mathcal{A}(V * F) : \mathrm{var}(\varphi) \cap V = \emptyset\} = \{\varphi \in \mathcal{A}(F) : \mathrm{var}(\varphi) \cap V = \emptyset\}$.
5. $\mathcal{A}$ is **invariant under renaming**, if for every F' obtained from F by renaming v to w using h (recall Subsection 3.3) and for every autarky $\varphi \in \mathcal{A}(F)$ we have $\varphi' \in \mathcal{A}(F')$ for the partial assignment φ' obtained from φ by renaming v to w using h.
6. $\mathcal{A}$ is **stable for unused values**, if for $\varphi \in \mathcal{A}(F)$, $v \in \mathrm{dom}(\varphi)$ and for $\varepsilon \in D_v$ such that none of the two literals $(v, \varphi(v)), (v, \varepsilon)$ occurs in F, also $\varphi' \in \mathcal{A}(F)$ holds, where φ' is obtained from φ by flipping v to ε.

An autarky system $\mathcal{A}$ is called **normal**, if it fulfils these six criteria, that is, if it is iterative, standardised, $\perp$-invariant, stable under variable elimination, invariant under renaming and stable for unused values. Considering the boolean case (where stability for unused values is covered by the standardisation condition, while invariance under renaming was not considered), in [30, 31] "normal autarky systems" have been called "strong autarky systems", but meanwhile the above properties seem not so strong anymore to me, but quite "normal" (while "ab-normality" is a defect which can be repaired; see for example Lemma 8.4 in [31], which can be generalised to generalised clause-sets). Examples for normal autarky systems are the smallest standardised autarky system $F \in \mathcal{MCLS} \mapsto \{\varphi \in \mathcal{PASS} : \mathrm{var}(\varphi) \cap \mathrm{var}(F) = \emptyset\}$ and the largest autarky system $F \in \mathcal{MCLS} \mapsto \mathrm{Auk}(F)$. In this paper our main interest is in normal autarky systems, and thus we don't investigate further the relations between the above notions and the other properties of autarky systems, but we will state general results only either for all autarky systems or for all normal autarky systems.

Consider an autarky system $\mathcal{A}$. An **$\mathcal{A}$-reduction** is a reduction $F \mapsto \varphi * F$ for some non-trivial $\varphi \in \mathcal{A}(F)$. Since multi-clause-sets have finite variable sets, $\mathcal{A}$-reduction is terminating, and thus by Lemma 3.7 in [30] $\mathcal{A}$-reduction is confluent, and the result of applying $\mathcal{A}$-reductions as long as possible is uniquely determined, yielding a normal form $\mathbf{N}_{\mathcal{A}}(F) \le F$. As before, the operator $\mathrm{N}_{\mathcal{A}}$ is a kernel operator, that is, $\mathrm{N}_{\mathcal{A}}(F) \le F$, $\mathrm{N}_{\mathcal{A}}(\mathrm{N}_{\mathcal{A}}(F)) = \mathrm{N}_{\mathcal{A}}(F)$ and $F_1 \le F_2 \Rightarrow \mathrm{N}_{\mathcal{A}}(F_1) \le \mathrm{N}_{\mathcal{A}}(F_2)$. Multi-clause-sets F with $\mathrm{N}_{\mathcal{A}}(F) = \top$ are called **$\mathcal{A}$-satisfiable**, while in case of $\mathrm{N}_{\mathcal{A}}(F) = F$ we call F **$\mathcal{A}$-lean**; the set of all $\mathcal{A}$-satisfiable multi-clause-sets is denoted by $\boldsymbol{\mathcal{SAT}_{\mathcal{A}}}$, the set of all $\mathcal{A}$-lean multi-clause-sets by $\boldsymbol{\mathcal{LEAN}_{\mathcal{A}}}$. It is F $\mathcal{A}$-lean iff $\mathcal{A}(F)$ contains no non-trivial autarky. The learn kernel $\mathrm{N}_{\mathcal{A}}(F)$ is the largest $\mathcal{A}$-lean sub-multi-clause-set of F. A sum of $\mathcal{A}$-lean multi-clause-sets again is $\mathcal{A}$-lean. Finally F is called **minimally $\mathcal{A}$-unsatisfiable**, if F is not $\mathcal{A}$-satisfiable, but every $F' \le F$ with $F' \ne F$ is A-satisfiable, while F is called **barely $\mathcal{A}$-lean** if F is $\mathcal{A}$-lean, but every $F' \le F$ with $c(F') = c(F) - 1$ is not A-lean; for more on these two notions see [38], while in this article we will consider only some basic properties of minimal $\mathcal{A}$-unsatisfiability. If F is minimally $\mathcal{A}$-unsatisfiable, then F is $\mathcal{A}$-lean, and for $F \ne \{\bot\}$ it is F also barely $\mathcal{A}$-lean. If $\mathcal{A}$ is the full autarky system, then minimal $\mathcal{A}$-unsatisfiability is just normal minimal unsatisfiability.

For the remainder of this subsection now assume that the autarky system $\mathcal{A}$ is normal. Then F is $\mathcal{A}$-satisfiable iff there exists $\varphi \in \mathcal{A}(F)$ with $\varphi * F = \top$. More generally, there always exists $\varphi \in \mathcal{A}(F)$ with $\varphi * F = \mathrm{N}_{\mathcal{A}}(F)$. If F is $\mathcal{A}$-lean, then so is $V * F$ for finite $V \subseteq \mathcal{VA}$. The $\mathcal{A}$-autark sub-multi-clause-sets of F, i.e., those multi-clause-sets F' where there is $\varphi \in \mathcal{A}(F)$ with $F' = F_{\mathrm{var}(\varphi)}$, are exactly those F_V for some $V \subseteq \mathrm{var}(F)$ where $F[V]$ is $\mathcal{A}$-satisfiable. On the other hand, if F is $\mathcal{A}$-lean, then so is $F[V]$ (for all finite $V \subseteq \mathcal{VA}$). The set of $\mathcal{A}$-autark sub-multi-clause-sets of F is closed under union, and contains the smallest element $\top$ and the largest element $F \setminus \mathrm{N}_{\mathcal{A}}(F)$. As before, the relation between the $\mathcal{A}$-lean kernel of F and the largest $\mathcal{A}$-autark sub-multi-clause-set of F can be summarised as follows: For $F \in \mathcal{MCLS}$ there exist induced sub-multi-clause-sets $F_1, F_2 \le F$ with $F_1 + F_2 = F$, such that F_1 is $\mathcal{A}$-lean, while $\mathrm{var}(F_1) * F_2$ is $\mathcal{A}$-satisfiable; in this decomposition F_1, F_2 are uniquely determined, namely $F_1 = \mathrm{N}_{\mathcal{A}}(F)$ is the largest $\mathcal{A}$-lean sub-multi-clause-set (the $\mathcal{A}$-lean kernel), while F_2 is the largest $\mathcal{A}$-autark sub-multi-clause-set.

We finish our review on autarkies and autarky systems by generalising Lemma 8.6 in [31]. The proof can be literally transferred to our generalised context, and thus is not reproduced here.

Lemma 3.1 *Let $\mathcal{A}$ be a normal autarky system. Given decision of membership in $\mathcal{LEAN}_{\mathcal{A}}$ as an oracle, the normal form $F \mapsto \mathrm{N}_{\mathcal{A}}(F)$ for $F \in \mathcal{MCLS}$ can be computed in polynomial time as follows:*

1. *If $F \in \mathcal{LEAN}_{\mathcal{A}}$ then output F.*
2. *Let* $\mathrm{var}(F) = \{v_1, \dots, v_{n(F)}\}$.
3. *Since $\emptyset * F = F \notin \mathcal{LEAN}_{\mathcal{A}}$ and $\mathrm{var}(F) * F = c(F) \cdot \check{\mathrm{t}}(\{\bot\}) \in \mathcal{LEAN}_{\mathcal{A}}$ holds, there is an index $1 \le i \le n(F)$ with*
$$\{v_1, \dots, v_{i-1}\} * F \notin \mathcal{LEAN}_{\mathcal{A}} \ \text{ and } \ \{v_1, \dots, v_i\} * F \in \mathcal{LEAN}_{\mathcal{A}}.$$

Replace F by the induced sub-multi-clause-set of F given by the clauses of F not containing variable v_i, and go to Step 1.

While the output of this procedure is $\mathrm{N_a}(F)$, if V is the set of variables v_i selected in Step 3, then F_V is the largest autark subset of F.

The idea behind the algorithm of Lemma 3.1 is, that we want to find a variable v such that there exists an autarky φ for F with $v \in \mathrm{var}(\varphi)$; if there is no such variable, then F is lean while otherwise we can eliminate all clauses from F containing variable v. Now the variable v_i selected in Step 3 must be such a variable: Consider a non-trivial autarky φ_i for $F_i := \{v_1, \dots, v_{i-1}\} * F$ with $\mathrm{var}(\varphi_i) \subseteq \mathrm{var}(F_i)$. Since $\{v_i\} * F_i$ is lean, it must $v_i \in \mathrm{var}(\varphi_i)$ be the case, while φ_i is an autarky also for F. For more on such reductions (for boolean clause-sets and the full autarky system) see [41].

3.7 Resolution

For autarky systems the number of occurrences of a clause in a multi-clause-set might make a difference (as it is the case for matching autarkies introduced in the subsequent section), however for all known resolution systems we do not need this distinction, and thus only (generalised) clause-sets are considered for resolution (that is, if multi-clause-sets $F \in \mathcal{MCLS}$ are to be treated, then they are automatically "downcast" to the underlying clause-set $\hat{\mathrm{t}}(F)$).

The resolution rule for generalised clause-sets is well-known. The most thorough study for my knowledge in given in [36], where actually resolution is considered for general "fipa-systems" (systems with finite instantiation by partial assignments) by reducing resolution for such axiomatic systems to resolution for generalised clause-sets, which act as "no-goods", i.e., out of the general system we get the clauses C belonging to the resolution refutation as clauses C_φ associated with such partial assignments, which led to a contradiction. In this subsection the most basic notions are reviewed, and the interesting connection to autarkies is given.

Consider a variable $v \in \mathcal{VA}$. "Parent clauses" $C_1, \dots, C_{|D_v|}$ are called **resolvable with resolution variable** v, if $\mathrm{val}_v(\{C_1, \dots, C_{|D_v|}\}) = D_v$ and the **resolvent** $R := \bigcup_{i=1}^{|D_v|} \{v\} * C_i$ actually is a clause (contains no clashing literals), that is, whenever there are literals $x \in C_i$, $y \in C_j$ for some $i, j \in \{1, \dots, |D_v|\}$ with $x \neq y$ and $\mathrm{var}(x) = \mathrm{var}(y)$, then $\mathrm{var}(x) = v$ must be the case. Resolution is a complete and sound refutation system; see for example Corollary 5.9 in [36], where, translating branching trees into resolution trees, the existence of a resolution tree with at most $(\max_{v \in \mathrm{var}(F)} |D_v|)^{n(F)}$ many leaves for unsatisfiable generalised clause-sets F is shown. Also stated in [36] is the (well-known) "strong completeness" of resolution, that is, for a multi-clause-set $F \in \mathcal{MCLS}$ and a clause $C \in \mathcal{CL}$ we have $F \models \{C\}$ iff there exists a resolution tree with axioms from F deriving a clause $C' \subseteq C$.

In Theorem 3.16 in [29] it was shown for boolean clause-sets, that the lean kernel of a clause-set F consists exactly of all clauses $C \in F$ which can be used in some resolution refutation of F.[12] This theorem can be immediately generalised to generalised clause-sets, using exactly the proof from [29] (together with the proof

[12] Where the resolution refutation may not contain "dead ends", which can be most easily enforced by considering only resolution *trees*.

transformation tools provided in [36]). In [30], Theorem 4.1 this generalisation is stated, but without a proof, which we now outline as follows. Consider the set $U(F)$ of clauses $C \in F$ for which there exists a tree resolution refutation of F using C as an axiom. The direction, that a clause $C \in F \setminus \mathrm{N_a}(F)$ can not be used in tree resolution refutations of F (i.e., $U(F) \subseteq \mathrm{N_a}(F)$), is easily proved by induction (an autarky of F satisfying C satisfies also all clauses derived from C in the tree). For the reverse direction the main technical lemma is, that for each variable $v \in \mathrm{var}(U(F))$ and each $\varepsilon \in D_v$ the unit-clause $\{(v, \varepsilon)\}$ can be derived from $U(F)$ by resolution (this is a little proof-theoretic exercise; see Lemma 3.14 in [29] for the boolean case). Now it follows, that $F \setminus U(F)$ is an autark sub-clause-set of F, since if the clause-set $\mathrm{var}(U(F)) * (F \setminus U(F))$ would be unsatisfiable, then there would be a tree resolution refutation T of $\mathrm{var}(U(F)) * (F \setminus U(F))$, where the axioms of T could be derived from the clauses in $F \setminus U(F)$ and the clauses in $U(F)$ by the above technical lemma, and thus we could construct a tree resolution refutation involving some clause of $F \setminus U(F)$, contradicting the definition of $U(F)$ (compare with Lemma 3.15 in [29] for the boolean case). That $F \setminus U(F)$ is an autark sub-clause-set of F means $\mathrm{N_a}(F) \subseteq U(F)$, and altogether we have shown

Theorem 3.2 *For any generalised clause-set $F \in \mathcal{CLS}$ the lean kernel $\mathrm{N_a}(F)$ equals the set $U(F)$ of clauses of F usable in some (tree) resolution refutation of F. Especially it is F lean if and only if $F = U(F)$, that is, if every clause of F can be used in some (tree) resolution refutation of F.*

As shown in Section 6 of [30], Theorem 3.2 yields an algorithm for computing $\mathrm{N_a}(F)$ by using "intelligent backtracking solvers", which on unsatisfiable instances can return the set of variables used in some resolution refutation of the input. Crossing out these variables from the input, removing the empty clause obtained, and repeating this process, we finally obtain a satisfiable clause-set F^*, and now any satisfying assignment φ for F^* with $\mathrm{var}(\varphi) \subseteq \mathrm{var}(F^*)$ is an autarky for F with $\varphi * F = F \setminus \mathrm{N_a}(F)$. See [40] for more details on this computation of the lean kernel (in [40] only boolean clause-sets are considered, but based on the results of the present article, all (mathematical) results can be generalised in the natural way).

Corollary 3.3 *Consider a class $\mathcal{C} \subseteq \mathcal{CLS}$ of generalised clause-sets, such that for all unsatisfiable $F \in \mathcal{C}$ and all $V \subseteq \mathcal{VA}$ we have $F[V] \in \mathcal{C}$. Assume furthermore, that there is an algorithm running for inputs $F \in \mathcal{C}$ in polynomial time, which either computes a satisfying assignment for F or computes the set of variables used in some tree resolution refutation of F. Then for inputs $F \in \mathcal{C}$ the lean kernel $\mathrm{N_a}(F)$ is computable in polynomial time.*

A general source of classes $\mathcal{C}$ as required in Corollary 3.3 are the classes $\mathcal{G}_k(\mathcal{U}, \mathcal{S})$ for levels $k \in \mathbb{N}_0$ and suitable oracles $\mathcal{U}$ for unsatisfiability and $\mathcal{C}$ for satisfiability, as introduced in [36] (in this way for example we get poly-time computation of the lean kernel for generalised Horn clause-sets (using non-boolean variables as well as considering higher levels k of "Horn-structures")).

We conclude this subsection by defining the **Davis-Putnam operator** DP for generalised clause-sets. Consider a clause-set $F \in \mathcal{CLS}$ and a variable $v \in \mathrm{var}(F)$. Let F_v be the set of all resolvents of parent clauses in F with resolution variable v. Now we set $\boldsymbol{\mathrm{DP}_v(F)} := \{C \in F : v \notin \mathrm{var}(C)\} \cup F_v$. From the completeness results for (generalised) resolution in [36] it follows immediately, that $\mathrm{DP}_v(F)$ is satisfiability equivalent to F, and that F is unsatisfiable if and only if by repeated applications of the Davis-Putnam operator we finally obtain the clause-set $\{\bot\}$ (while for satisfiable F finally we will obtain the clause-set $\top$). We can also generalise Lemma 7.6 in [39] about the commutativity of the Davis-Putnam operator, that is, if G_1 is the result of applying first DP_{v_1}, then DP_{v_2}, ..., and finally applying DP_{v_m}, while G_2 is the result of applying first $\mathrm{DP}_{v_{\pi(1)}}$, then $\mathrm{DP}_{v_{\pi(2)}}$, ..., and finally applying $\mathrm{DP}_{v_{\pi(m)}}$, for some permutation $\pi \in S_m$, then after elimination of subsumed clauses in G_1 and G_2 (see the following subsection) G_1 becomes equal to G_2. It follows that for any set of variables V the operator DP_V, computed by running through the variables of V in some order, is uniquely determined up to subsumption reduction in the result. We always have $\mathrm{DP}_V(F) = \mathrm{DP}_V(F_V) \cup (F \setminus F_V)$. If for some $V \subseteq \mathrm{var}(F)$ we have $\mathrm{DP}_V(F_V) = \top$, then F and $F \setminus F_V$ are satisfiability equivalent, generalising the elimination of autark sub-clause-sets: If $\varphi \in \mathrm{Auk}(F)$, then $\mathrm{DP}_{\mathrm{var}(\varphi)}(F_{\mathrm{var}(\varphi)}) = \top$, while the reverse direction need not hold, as the example $F = \{\{v, a\}, \{\overline{v}, \overline{a}\}\} \cup F'$, $v \notin \mathrm{var}(F')$, with $V = \{v\}$ shows (for boolean variables). We see that the Davis-Putnam operator, whose application for generalised clause-sets is basically the same as existential quantification, yields more powerful reductions, but this at the cost of potential exponential space usage.

3.8 Reductions

In this subsection we review the most basic polynomial time reduction concepts. For a thorough discussion in the boolean case, see [39]. We consider only clause-sets $F \in \mathcal{CLS}$, but all results are easily generalised to multi-clause-sets.

The most basic reduction (by which we mean a satisfiability-equivalent transformation, simplifying the clause-set in some sense) is **subsumption elimination**, the elimination of subsumed clauses, i.e., the transition $F \to F \setminus \{C\}$ for $C \in F$ in case there exists $C' \in F$ with $C' \subset C$. Iterated elimination of subsumed clauses is confluent, and thus the result of applying subsumption elimination as long as possible is uniquely determined (namely it is the set of all minimal clauses of F); if F has no subsumed clauses, then we call F **subsumption-free**.

The next reduction can be called the **trivial-domain reduction**: If there exists $v \in \mathrm{var}(F)$ with $|D_v| = 1$ (we call such a variable **trivial**), then for $D_v = \{\varepsilon\}$ reduce $F \mapsto \langle v \to \varepsilon \rangle * F$. (that is, all literal occurrences with underlying variable v are removed).

Elimination of "pure literals" is now better called **elimination of pure variables**: If there is $v \in \mathrm{var}(F)$ with $|\mathrm{val}_v(F)| < |D_v|$, that is, one of the values of v is not used in F, then for some $\varepsilon \in D_v \setminus \mathrm{val}_v(F)$ reduce $F \mapsto \langle v \to \varepsilon \rangle * F$. This is the basic form of a **pure autarky** as mentioned in Subsection 4.4 of [30].

Unit-clause elimination for generalised clause-sets is less powerful than in the boolean case: If F contains a unit-clause $\{(v, \varepsilon)\} \in F$, then in case of $D_v = \{\varepsilon\}$ by trivial-domain reduction we conclude that F is unsatisfiable, but otherwise we can only conclude that value ε is to be excluded from the domain of v, and in general we cannot eliminate the variable v. It seems most convenient here to include trivial-domain reduction into unit-clause elimination (so that we properly generalise the boolean case); in our context, where we fixed the domain of each variable (and thus renaming is needed to achieve a change of domain), **unit-clause propagation** for (generalised) clause-sets F is then the following procedure:

1. Apply trivial-domain reduction to F to eliminate all trivial variables.

2. If $\bot \in F$, then reduce F to $\{\bot\}$.
3. For $\{(v, \varepsilon)\} \in F$ eliminate all clauses containing the literal (v, ε) from F, and replace variable v in the remaining occurrences by a new variable v' with $D_{v'} = D_v \setminus \{\varepsilon\} \neq \emptyset$.
4. Repeat Steps 1 - 3 until all trivial variables and all unit-clauses have been eliminated.

So after unit-clause propagation every variable has a domain with at least two elements and, except of the case $F = \{\bot\}$, every clause contains at least two literals. Unit-clause propagation for boolean clause-sets is confluent (the final result does not depend on the order and choice of single reduction steps), while unit-clause propagation for arbitrary clause-sets is confluent *modulo renaming*. Generalising the well-known linear time algorithm for unit-clause propagation in the boolean case, this normal form (the result of unit-clause propagation) can be computed in linear time.

Considering clauses $C \in \mathcal{CL}$ as constraints of scope $\mathrm{var}(C)$ (see [11], Subsection 2.1.1), and thus clause-sets $F \in \mathcal{CLS}$ as *constraint networks* (or *constraint satisfaction problems*), F is (hyper-)arc-consistent ([11], Definition 3.6) iff for all $C \in F \setminus \{\bot\}$ we have $|C| \geq 2$, while F is relational arc-consistent ([11], Definition 8.1) iff for all $v \in \mathrm{var}(F)$ we have $|D_v| \geq 2$. So unit-clause propagation achieves both hyper-arc-consistency and relational arc-consistency.

Finally we consider the most harmless cases for **DP-reductions**. In general, application of DP_v to F eliminates $\#_v(F) = \sum_{\varepsilon \in D_v} \#_{(v,\varepsilon)}(F)$ clauses and creates up to $\prod_{\varepsilon \in D_v} \#_{(v,\varepsilon)}(F)$ new clauses (with potential repetitions; less iff some of the parent clause combinations are not eligible for resolution due to additional clashes). Thus we have

$$c(\mathrm{DP}_v(F)) \leq c(F) - \sum_{\varepsilon \in D_v} \#_{(v,\varepsilon)}(F) + \prod_{\varepsilon \in D_v} \#_{(v,\varepsilon)}(F). \qquad (1)$$

Note that equality holds in (1) if v is pure for F (which is equivalent to the product being zero). If in (1) we have a strict inequality or v is a pure variable for F, then we call v a **degenerated DP-variable w.r.t.** F, while otherwise v is called a **non-degenerated DP-variable w.r.t.** F. Note that a missing new clause due to additional clashes is not the only cause of a strict inequality, but it is also possible that a resolvent is already contained in the rest of F, or that two resolvents coincide (and thus in both cases contraction occurs). Two trivial cases:

1. If variable $v \in \mathrm{var}(F)$ has a trivial domain (i.e., $|D_v| = 1$), then either we have a subsumption $C, C \cup \{(v, \varepsilon)\} \in F$ (for some clause C not containing v), in which case v is a degenerated DP-variable w.r.t. F, or otherwise v is a non-degenerated DP-variable with $c(\mathrm{DP}_v(F)) = c(F)$, and in both cases $\mathrm{DP}_v(F)$ is the result of applying trivial domain reduction to F.
2. If v is pure w.r.t. F, then F is a degenerated DP-variable with $c(\mathrm{DP}_v(F)) = c(F) - \#_v(F)$, and $\mathrm{DP}_v(F)$ is the result of applying the elimination of pure variable v to F.

Besides these cases, in this article we consider only one very restricted form of DP-resolution, characterised by the condition that at most one of the factors in the product from (1) might be greater than one:

We call a variable v a **singular DP-variable** w.r.t. F if there exists $\varepsilon \in D_v$ such that for all $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we have $\#_{(v,\varepsilon')}(F) = 1$, while $\#_{(v,\varepsilon)}(F) \geq 1$, i.e., v is not pure for F. In such a case of a singular DP-variable, application of DP_v eliminates $|D_v| - 1 + \#_{(v,\varepsilon)}(F)$ clauses and creates up to $\#_{(v,\varepsilon)}(F)$ new clauses, so that the number of clauses goes down at least by one if $|D_v| \neq 1$. For a singular DP-variable v we have:

- If v is a non-degenerated DP-variable w.r.t. F, then we have $c(\mathrm{DP}_v(F)) = c(F) - |D_v| + 1$.
- If v is a degenerated DP-variable w.r.t. F, then at least one of the clauses in F containing v can be eliminated satisfiability-equivalently, and we call such a clause elimination a **singular DP-degeneration reduction**.

Since a singular DP-degeneration reduction cannot be applied to a minimally unsatisfiable clause-set, a singular variable w.r.t. a minimally unsatisfiable clause-set must be non-degenerated. Actually more can be said here:

Lemma 3.4 *Consider a generalised clause-set $F \in \mathcal{CLS}$ and a singular DP-variable v w.r.t. F. Then the following two conditions are equivalent:*

1. *F is minimally unsatisfiable.*
2. *v is a non-degenerated DP-variable w.r.t. F and $\mathrm{DP}_v(F)$ is minimally unsatisfiable.*

Proof: We have already seen, that if F is minimally unsatisfiable, then v is non-degenerated. If $\mathrm{DP}_v(F)$ were not minimally unsatisfiable, then there would be a clause $C \in \mathrm{DP}_v(F)$ such that $\mathrm{DP}_v(F) \setminus \{C\}$ would still be unsatisfiable, and thus would have a resolution refutation — now it is easy to see that in this case we would also obtain a resolution refutation of F not using one of the clauses in F.

For the reverse direction assume that v is non-degenerated and that $\mathrm{DP}_v(F)$ is minimally unsatisfiable. By a similar argumentation as for the other direction, if there would be a resolution refutation of F not using one of the clauses from F, then one could construct a resolution refutation of $\mathrm{DP}_v(F)$ not using (at least) one of the clauses from $\mathrm{DP}_v(F)$. ∎

We call application of DP-reduction for non-degenerated singular variables **non-degenerated singular DP-reduction**. In the boolean case, this form of DP-reduction is used at many places in the literature (in [26], Appendix B, it is called "$(1, \infty)$-reduction"); see Lemma 4.30 for more on singular DP-reduction.

We conclude by another reduction arising from the DP-operator. The notion of **blocked clauses** for boolean clause-sets (see [27, 28]) can be generalised by calling a clause C **blocked w.r.t.** F if there exists a variable $v \in \mathrm{var}(C)$ with $\mathrm{DP}_v(F \cup \{C\}) = \mathrm{DP}_v(F \setminus \{C\})$. If $C \in F$ is blocked w.r.t. F, then F is satisfiability equivalent to $F \setminus \{C\}$, and such a reduction is called **elimination of blocked clauses**. If v is a pure variable for F, then all clauses of F containing variable v are blocked w.r.t. F. And if v is a degenerated singular DP-variable, then F has at least one blocked clause containing v, and so singular DP-degeneration reduction is also covered by elimination of blocked clauses.

3.9 Conflict structure

A study of the "combinatorics of conflicts" for boolean clause-sets has been initiated with [32, 33] and continued with [19, 34]. We generalise here only a very few simple notions used later in this article.

The **conflict graph** $\mathbf{cg}(\boldsymbol{F})$ of a clause-set $F \in \mathcal{MCLS}$ has as vertices the clauses of F, and edges joining two vertices C, D with a **clashing literal** between C and D, that is, there is a literal $x \in C$ for which there exists a literal $y \in D$ with $\mathrm{var}(x) = \mathrm{var}(y)$ and $x \neq y$. A clause-set F is called a **hitting clause-set** if the conflict graph of F is a complete graph, and the **hitting degree** $\mathbf{hd}(\boldsymbol{F}) \in \mathbb{N}$ of a hitting clause-set with at least two clauses is the maximum of the number of edges joining two different vertices in the conflict multigraph of F. More specifically we call F a $\boldsymbol{r}$**-regular hitting clause-set** for $r \in \mathbb{N}_0$ if for every two different clauses in F have exactly r conflicts (thus if F is r-regular hitting for $r \geq 1$, then F is hitting), while a **regular hitting clause-set** is an r-regular hitting clause-set for some $r \geq 0$, and we denote the set of regular hitting clause-sets by $\boldsymbol{\mathcal{RHIT}}$.[13)]

More generally a clause-set F is called **at most** k**-multihitting** for some $k \in \mathbb{N}_0$ if the conflict graph of F is complete k-partite, while F is called **multihitting** if it is at most k-multihitting for some k; let $\boldsymbol{\mathcal{MHIT}}$ denote the set of all multihitting clause-sets. While "at most k-multihitting" implies that the chromatic number of the conflict graph is at most k, if we speak of k**-multihitting** then the chromatic number of the conflict graph must be equal to k (so that F is hitting iff F is $c(F)$-multihitting). For a multihitting clause-set F the **multihitting number** $\mathbf{mh}(\boldsymbol{F}) \in \mathbb{N}_0$ is the unique k such that F is k-multihitting. For a given multihitting clause-set F there is a unique partition $\mathbb{F}$ of F (that is, $\mathbb{F}$ is a set of sub-clause-sets of F which are non-empty and pairwise disjoint, such that their union is F), so that for any clauses $C_1, C_2 \in F$ with $C_i \in F_i \in \mathbb{F}$ for $i \in \{1, 2\}$ the clauses C_1 and C_2 clash if and only if $F_1 \neq F_2$; we call $\mathbb{F}$ the **multipartition** of F (if F is bihitting, then $\mathbb{F}$ is also called the **bipartition** of F).

4 Matching autarkies

In this section we introduce the autarky system for generalised clause-sets given by "matching autarkies", and we show various polynomial time procedures. "Matching autarkies" for clause-sets with non-boolean variables have been introduced in [30], and some basic properties have been stated regarding the standard translation of clause-sets with non-boolean variables to clause-sets with boolean variables, but as we will discuss in Subsection 4.5, this earlier version of the notion is actually too restrictive (another example which shows, that the generalisations in this paper are not completely straight-forward but invoke subtleties one needs to get right). An overview on our results is as follows.

The purpose of **Subsection 4.1** is to generalise the notion of deficiency $\delta(F)$, which has been introduced for boolean clause-sets F in [17] as $\delta(F) = c(F) - n(F)$. As for boolean clause-sets we will obtain "matching satisfiable clause-sets" F characterised by the condition $\delta^*(F) = 0$ (where $\delta^*(F)$ is the maximal deficiency taken over all sub-clause-sets of F), which is equivalent to a certain matching situation.

[13)] Regular hitting clause-sets were called "uniform hitting clause-sets" in [33], but "regular" seems a better choice, since regularity refers to constant degree, while uniformity typically refers to constant hyperedge-size, and so should better be used for clause-sets of constant clause-size.

Whence matching satisfiability can be decided in polynomial time by finding a maximum matching, which yields also a satisfying assignment (called a "matching satisfying assignment") in the positive case.

In **Subsection 4.2** we investigate the relation between general satisfiability and matching satisfiability. We will see that if a clause-set F is satisfiable, then it has a matching satisfiable sub-clause-set F' with at most $\delta^*(F)$ less clauses than F, and moreover there is a matching satisfying assignment φ_0 for F' which can be extended to a satisfying assignment φ for F using at most $\delta^*(F)$ additional variables. The proof shows, that every satisfying assignment φ for F can be modified (in polynomial time) to become such an extension by means of flips of (single) variable assignments such that throughout the whole process we always have a satisfying assignment for F.14) As an application we obtain in Corollary 4.10 that the hierarchy of clause-sets given by the parameter δ^* allows polynomial-time SAT decision for each level; in Theorem 5.5 we will see that actually this hierarchy is fixed-parameter tractable by combining the structural results from Subsection 4.3 with the fixed-parameter tractability of the boolean case.

Having a (restricted) concept $\mathfrak{C}$ of satisfying assignments, we can "typically" obtain an autarky system (recall Subsection 3.5) by calling φ a "$\mathfrak{C}$-autarky" for a clause-set F if φ is a $\mathfrak{C}$-satisfying assignment for $F[\mathrm{var}(\varphi)]$ (recall Subsection 3.4), or, equivalently at least for general satisfiability, if φ is a $\mathfrak{C}$-satisfying assignment for $F_{\mathrm{var}(\varphi)}$. We have to leave such a general theory to future work, but in this article we will consider "matching autarkies" obtained in this way from matching satisfying assignments in **Subsection 4.3**. A central notion is the notion of a "tight sub-clause-set" F' of a clause-set F, characterised by the condition $\delta(F') = \delta^*(F)$ (that is, F' realises the maximal deficiency of F). Translating general results of matching theory into our setting, the set of tight sub-clause-sets of F form a set-lattice (i.e, union and intersection of tight sub-clause-sets are again tight), and so we have a smallest and a largest tight sub-clause-set. In Corollary 4.20 we see that the smallest tight sub-clause-set of F is identical to the lean kernel of F (obtained from F by repeated matching-autarky reduction). It follows that if F is matching lean, then all strict sub-clause-sets of F have a deficiency strictly less than the deficiency of F (actually, this condition characterises matching leanness, as shown in Lemma 4.17), and thus, since the empty sub-clause-set has deficiency 0, we obtain $\delta(F) \geq 1$ for non-empty matching lean clause-sets. We remark here, that applying the general procedure from Lemma 3.1 we obtain polynomial-time computability of the matching lean kernel (in Corollary 4.19), but that a direct computation using matching arguments is more efficient (not discussed in this paper, since here we do not go into algorithmic details).

In Subsection 4.4 the notions of "expansion" and "surplus" are studied, that is, if we consider arbitrary non-empty sets of variables and all clauses containing at least one of them, how many more clauses than variables do we have at least? Having at least a surplus of 1 is by Lemma 4.25 equivalent to being matching lean. One important application of the surplus is in establishing that by applying a partial assignment using at most one variable the maximal deficiency must get smaller; this matter (which leads to fixed-parameter tractability of satisfiability in the maximal deficiency by the method of "bounded search trees") is discussed in the remainder of the subsection.

14) This additional property yields also new information for the boolean case; it is implicitly contained in the proofs from [14], which are not only generalised here, but also simplified in such a way, that the construction becomes more lucid.

Finally, in Subsection 4.5, we reflect on the definition of "deficiency" and "matching autarky" as defined in this article, by comparing it with an earlier version of these notions (for generalised clause-sets).

Let us close the introduction to this section by two technical remarks:

1. Matching arguments are sensitive to repetition of clauses, and thus in this section, instead of just using clause-sets we have to use the more general notion of a *multi*-clause-set (recall Subsection 3.1).
2. In case of a pure variable $v \in \mathrm{var}(F)$ for some $F \in \mathcal{MCLS}$ (that is, not all values $\varepsilon \in D_v$ are used in F) we assume that D_v contains exactly one value not used in F (i.e., $|D_v| = |\mathrm{val}_v(F)| + 1$); in this way we are not troubled anymore by the unknown domain size D_v, but we can measure the size of F just by $\ell(F)$, while this modification has no influence on any of the notions and procedures in this article (all autarky systems studied here are stable for unused values (recall Subsection 3.6)).

4.1 Matching satisfiable generalised clause-sets

We wish to generalise the notion of "matching satisfiable clause-sets", introduced in [31] for boolean clause-sets. Consider a multi-clause-set F together with a decomposition $F = F_1 + \cdots + F_m$ for $m \in \mathbb{N}_0$ and $F_i \in \mathcal{MCLS}$, fulfilling the following conditions:

(i) for $i \in \{1, \dots, m\}$ there are variables $v_i \in \mathrm{var}(F_i)$ such that for all clauses $C \in F_i$ we have $v_i \in \mathrm{var}(C)$;

(ii) the variables $v_1, \dots, v_m$ are pairwise different;

(iii) for all $i \in \{1, \dots, m\}$ we have $|D_{v_i}| > |\mathrm{val}_{v_i}(F_i)|$, that is, v_i is a pure variable for F_i.

Given such a decomposition, we see that F is satisfiable, since for each i there exists $\varepsilon_i \in D_{v_i} \setminus \mathrm{val}_{v_i}(F_i)$, and the assignment $\langle v_i \to \varepsilon_i : i \in \{1, \dots, m\}\rangle$ is a satisfying assignment for F (note that none of the variables v_i needs to be a pure variable in F). If we consider on the other hand an arbitrary partial assignment φ satisfying F with $\mathrm{var}(\varphi) = \{v_1, \dots, v_m\}$, and set F_i for $i \in \{1, \dots, m\}$ as the induced sub-multi-clause-set of F given by the clauses $C \in F$ with $v_i \in \mathrm{var}(C)$ and $(v_i, \varphi(v_i)) \notin C$, then we obviously fulfil the above conditions, and we see that conditions (i) - (iii) need to be restricted so that we can obtain a class of satisfiable clause-sets which is decidable in polynomial time. We observe that $c(F_i) \geq |\mathrm{val}_{v_i}(F_i)|$ is true for arbitrary multi-clause-sets F_i, and thus condition

(iii)' for all $i \in \{1, \dots, m\}$ we have $|D_{v_i}| > c(F_i)$

strengthens condition (iii). We call multi-clause-sets $F \in \mathcal{MCLS}$ having a decomposition $F = F_1 + \cdots + F_m$ fulfilling conditions (i), (ii) and (iii)' **matching satisfiable**, and the set of all matching satisfiable (generalised) multi-clause-sets is denoted by $\boldsymbol{\mathcal{MSAT}}$.

To understand the connection to matching problems, we introduce the bipartite graph $B(F)$ for generalised multi-clause-sets $F \in \mathcal{MCLS}$:

- Let
$$\begin{aligned}\underline{F} &:= \{\,(C,i) : C \in F,\ i \in \{1,\dots,F(C)\}\,\}\\ \underline{V} &:= \{\,(v,j) : v \in \operatorname{var}(F),\ j \in \{1,\dots,|D_v|-1\}\,\}.\end{aligned}$$

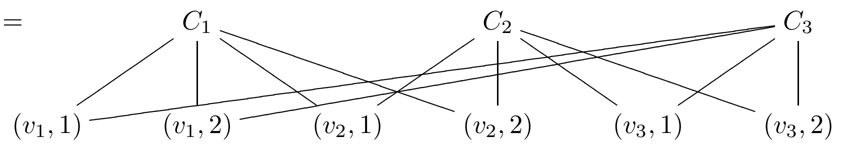
  The elements of $\underline{F}$ are called the *clause-nodes*, while the elements of $\underline{V}$ are called the *variable-nodes*.

- The vertex set of $B(F)$ is defined as $V(B(F)) := \underline{F} \mathbin{\dot\cup} \underline{V}$ with canonical bipartition $(\underline{F}, \underline{V})$.

- The edge set $E(B(F))$ is the set of all (undirected) edges $\{(C,i),(v,j)\}$ such that $v \in \operatorname{var}(C)$.

In other words, the graph $B(F)$ has as vertices $F(C)$-many copies of clauses $C \in F$ together with $(|D_v|-1)$-many copies of variables $v \in \operatorname{var}(F)$, while edges connect copies of variables v with copies of clauses C such that $v \in \operatorname{var}(C)$. We remark that variables $v \in \operatorname{var}(F)$ with trivial domain (i.e., $|D_v| = 1$) do not occur in $B(F)$, and that for boolean clause-sets F the graph $B(F)$ is the ordinary (bipartite) clause-variable graph. Consider for example the clause-set $F = \{C_1, C_2, C_3\}$ with $C_1 = \{(v_1,a),(v_2,a)\}$, $C_2 = \{(v_2,b),(v_3,b)\}$, $C_3 = \{(v_3,c),(v_1,c)\}$, where $D_{v_i} = \{a,b,c\}$. Now $B(F)$ is (suppressing the indices for the clause-copies, since here we just have a clause-set):

$B(F) =$



For a set V of variables we obtain $B(V * F)$ from $B(F)$ by deleting the variable-nodes (v,j) of $B(F)$ with $v \in V$, while $B(F[V])$ is the induced subgraph of $B(F)$ given by the variable-nodes (v,j) of $B(F)$ with $v \in V$ together with their neighbours (those clause-nodes (C,i) with $\operatorname{var}(C) \cap V \neq \emptyset$).

Lemma 4.1 *A multi-clause-set F is matching satisfiable iff there exists a matching in $B(F)$ covering all vertices of $\underline{F}$.*

Proof: If F is matching satisfiable, then (using the notations in the definition of matching satisfiability above) the clause-nodes corresponding to the clause-occurrences in F_i can all be covered by the the variable-nodes belonging to v_i (since $c(F_i)$ does not exceed the number of copies of v_i), and altogether we obtain a matching covering all clause-nodes. If (for the other direction) we have a matching in $B(F)$ covering all vertices of $\underline{F}$, then for each variable v involved in the matching consider a sub-multi-clause-set F_v of F corresponding to the clause-nodes connected via the matching to the variable-nodes associated with v. These F_v together constitute the desired decomposition of F. ▮

Using the **weighted number of variables $\mathbf{wn}(\boldsymbol{F})$** $:= \sum_{v \in \operatorname{var}(F)} (|D_v| - 1) \in \mathbb{N}_0$, the number of vertices of $B(F)$ is $|V(B(F))| = c(F) + \operatorname{wn}(F)$, while the number of edges is $|E(B(F))| = \sum_{v \in \operatorname{var}(F)} \#_v(F) \cdot (|D_v| - 1)$. We have $\operatorname{wn}(F) = (\sum_{v \in \operatorname{var}(F)} |D_v|) - n(F)$. If F is boolean, then $\operatorname{wn}(F) = n(F)$.

Let the **deficiency** of a (generalised) multi-clause-set F be defined as

$$\boldsymbol{\delta(F)} := c(F) - \operatorname{wn}(F) \in \mathbb{Z},$$

while the **maximal deficiency** is defined as

$$\boldsymbol{\delta^*(F)} := \max_{F' \leq F} \delta(F') \in \mathbb{N}_0$$

(we have $\delta^*(F) \geq 0$ due to $\delta(\top) = 0$); by definition we have $\delta^*(F) \leq c(F)$. We remark, that for a domain uniformisation F' of F we have $\delta(F') = \delta(F)$ as well as $\delta^*(F') = \delta^*(F)$ (in principle we could consider only multi-clause-sets with uniform domains here, but the advantages in doing so seem to be negligible).

Considering $F' \leq F$ as a subset of $\underline{F}$, the deficiency $\delta(F')$ of $F' \leq F$ is just the deficiency of this subset in $B(F)$ (as we have defined it for arbitrary graphs). By matching theory the maximal number of nodes of $\underline{F}$ coverable by some matching thus is $c(F) - \delta^*(F)$. Summarising we have (generalising Lemma 7.2 in [31]):

Lemma 4.2 *Consider a generalised multi-clause-set $F \in \mathcal{MCLS}$.*

1. *The maximal size of a matching satisfiable sub-multi-clause-set $F' \leq F$ is $c(F') = c(F) - \delta^*(F)$.*
2. *F is matching satisfiable if and only if $\delta^*(F) = 0$.*

As an application we can generalise the well-known fact, apparently first mentioned in the literature in [50], that if a boolean clause-set F has minimal clause-length k and maximal variable occurrence k for some $k \geq 1$, then F must be satisfiable (see [22] for recent further developments):

Corollary 4.3 *Consider a generalised clause-set $F \in \mathcal{CLS}$ containing a non-empty clause. Then*

$$\frac{\max_{v \in \mathrm{var}(F)} \#_v(F)}{\min_{C \in F} |C|} \leq \min_{v \in \mathrm{var}(F)} |D_v| - 1 \Longrightarrow F \in \mathcal{MSAT}.$$

Proof: Assume the condition holds, and consider $F' \subseteq F$. We have to show $\delta(F') \leq 0$. Let $d := \min_{v \in \mathrm{var}(F)} |D_v|$. Then $\delta(F') \leq c(F') - (d-1)n(F')$, and a sufficient condition for $\delta(F') \leq 0$ is $\frac{c(F')}{n(F')} \leq d - 1$. Let $a := \max_{v \in \mathrm{var}(F)} \#_v(F)$ and $b := \min_{C \in F} |C|$. We know $c(F') \cdot b \leq \ell(F') \leq n(F') \cdot a$, and thus $\frac{c(F')}{n(F')} \leq \frac{a}{b}$. ∎

Since matchings of maximal size can be computed in polynomial time (see Chapter 16 in [48]), we get the following poly-time results:

Lemma 4.4 *For every generalised clause-set $F \in \mathcal{MCLS}$, in polynomial time in $\ell(F)$ we can compute $F' \leq F$ with $F' \in \mathcal{MSAT}$ such that $c(F')$ is maximal. Since $F' = F$ iff F is matching satisfiable, it follows that whether F is matching satisfiable is decidable in polynomial time. And due to $c(F') = c(F) - \delta^*(F)$ furthermore the maximal deficiency $\delta^*(F)$ is also computable in polynomial time.*

4.2 Satisfying assignments versus matching satisfying assignments

After having established the notion of matching satisfiable (generalised) clause-sets in the previous paragraph, the first task now, tackled in Subsection 4.2.1, is to introduce and study the corresponding notion of "matching-satisfying assignments". For arbitrary clause-sets, matching-satisfying assignments are rare, so we consider in Subsection 4.2.2 the question whether at least there are some "good" satisfying assignments which are matching-satisfying assignments for sub-clause-sets as large as possible? In Corollary 4.8 we will see that this actually is always the case, and as a direct application we obtain in Corollary 4.10 polynomial time decidability of satisfiability for (generalised) (multi-)clause-sets, given that the maximal deficiency is bounded by some constant. Now in a certain sense this result is superceded by Theorem 5.5, where we show that satisfiability for (generalised) clause-sets is fixed-parameter tractable in the maximal deficiency, so what is the point here? I believe that the underlying method for proving Corollary 4.10 (which is different from the method underlying Theorem 5.5) is worth further investigations, when considering the stronger statement in Theorem 4.7 which allows to "repair" arbitrary (partial) assignments φ_0 by a sequence of "harmless" variable-flips, obtaining a (partial) assignment φ which satisfies everything which was already satisfied by φ_0 (and potentially more), while being matching-satisfying for a sub-clause-set of maximal size. Possible algorithmic applications (for example in the context of local search algorithms) are discussed in Subsection 7.1. The underlying graph-theoretical method, generalising the augmenting-path method to construct maximum matchings in bipartite graphs by allowing "parameterised graphs", may also be of independent interest (see Lemma 4.11).

4.2.1 The notion of matching-satisfying assignments

Consider a (generalised) multi-clause-set $F \in \mathcal{MCLS}$ and a partial assignment $\varphi \in \mathcal{PASS}$. The partial graph $\boldsymbol{B_\varphi(F)}$ of $B(F)$ is obtained from $B(F)$ by keeping (exactly) all edges $\{(C, i), (v, j)\}$ where φ satisfies the literal in C with underlying variable v (while keeping all vertices); in other words, all edges $\{(C, i), (v, j)\}$ are eliminated such that for the literal $(v, \varepsilon) \in C$ we either have $v \notin \mathrm{var}(\varphi)$ or $\varphi((v, \varepsilon)) = 0$ (i.e., $\varphi(v) = \varepsilon$). So the non-isolated clause-nodes in $B_\varphi(F)$ are (exactly) the clauses satisfied by φ, while the isolated variable-nodes in $B_\varphi(F)$ are (exactly) the variables in F not used by φ to satisfy any clause. Now φ is called a **matching-satisfying assignment** for F if $B_\varphi(F)$ contains a matching covering all clause-nodes (thus matching satisfying assignments are satisfying assignments). By Lemma 4.1 we get that F is matching satisfiable iff there exists a matching satisfying assignment for F, while by computing a maximum matching in $B(F)$ we can also efficiently compute a matching-satisfying assignment for matching satisfiable multi-clause-sets (compare Lemma 4.4). The following two lemmas give simple basic properties regarding these notions.

Lemma 4.5 *Consider a generalised multi-clause-set $F \in \mathcal{MCLS}$ and a partial assignment $\varphi \in \mathcal{PASS}$.*

1. *If φ is satisfying for F, then there exists $\varphi' \subseteq \varphi$ with $n(\varphi') \le c(F)$ such that also φ' is satisfying for F. (So for satisfying an arbitrary clause-set we never need to use more variables than there are clauses.)*
2. *If φ is matching satisfying for F, then there exists $\varphi' \subseteq \varphi$ with $n(\varphi') = c(F)$ such that also φ' is matching-satisfying for F. (So for a matching satisfiable multi-clause-set there is a matching-satisfying assignment using exactly as many variables as there are clause-occurrences.)*
3. *If φ is satisfying for F, and there is no $\varphi' \subseteq \varphi$ with $n(\varphi') < c(F)$ such that φ' is satisfying for F, then φ is matching-satisfying for F.*

4. *Consider a minimal satisfying assignment φ for F w.r.t. the canonical partial ordering of partial assignments (that is, there is no $\varphi' \subset \varphi$ which still satisfies F). Then φ is matching-satisfying for F if and only if $n(\varphi) = c(F)$.*

Proof: The partial assignment φ' for Part 1 is obtained by removing edges from $B_\varphi(F)$ until every clause-node has degree 1, and using then only the variables from φ which are still covered. φ' for Part 2 is obtained in a similar way, only this time we remove all edges not contained in some (selected) maximum matching of $B_\varphi(F)$. Part 3 is shown by Hall's criterion as follows: Assume that there is $F' \le F$ such that the number $|\Gamma_{B_\varphi(F)}(F')|$ of neighbours of F' in $B_\varphi(F)$ is strictly smaller than $c(F')$, and let $\varphi' := \varphi \mid \Gamma_{B_\varphi(F)}(F')$ be the restriction of φ to these neighbours; by definition φ' is a satisfying assignment for F' with $n(\varphi') < c(F')$. Let $F'' := F - F'$. With Part 1 there is a satisfying assignment $\varphi'' \subseteq \varphi$ for F'' with $n(\varphi'') \le c(F'') = c(F) - c(F')$. Now let $\varphi^* := \varphi' \cup \varphi''$; by definition φ^* is a satisfying assignment for F with $n(\varphi^*) \le n(\varphi') + n(\varphi'') < c(F') + c(F) - c(F') = c(F)$ contradicting the assumption. Finally Part 4 follows by Parts 2 and 3. ∎

Lemma 4.6 *Consider a generalised multi-clause-set $F \in \mathcal{MCLS}$ and partial assignments $\varphi, \psi \in \mathcal{PASS}$. If $\varphi \circ \psi$ is matching-satisfying for F, then φ is matching-satisfying for $\psi * F$.*

Proof: Let M be a matching in $B_{\varphi\circ\psi}(F)$ covering all clause-nodes. The bipartite graph $B_\varphi(\psi * F)$ is obtained from $B_{\varphi\circ\psi}(F)$ by removing all clause-nodes satisfied by ψ, removing all variable-nodes assigned by ψ, and finally removing all variable-nodes where the variable does not occur in $\psi * F$ anymore. Now those edges from M which are still in $B_{\varphi\circ\psi}(F)$ yield a matching (obviously, since only edges have been removed) covering all remaining clause-nodes (they were covered before, and only useless edges have been removed). ∎

4.2.2 Matchings within satisfying assignments

As we already remarked, the matching number $\nu(B(G))$ of the clause-variable graph of G, the maximum size of a matching in $B(G)$, is $\nu(G) = c(F) - \delta^*(F)$. Obviously for partial assignments φ we have $\nu(B_\varphi(F)) \le \nu(B(F))$; call φ **matching-maximum** if $\nu(B_\varphi(F)) = \nu(B(F))$ holds. By Lemma 4.2 we know that there exists a matching-maximum partial assignment for every clause-set. The main result of Subsection 4.2 is the following theorem, which states that every partial assignment can be efficiently repaired by "conservative changes", so that we obtain a matching-maximum partial assignment. Here by a **conservative change** of a partial assignment φ w.r.t. a clause-set F we mean either adding some assignment $v \mapsto \varepsilon \in D_v$ for some $v \in \mathrm{var}(F) \setminus \mathrm{var}(\varphi)$, or performing a **conservative flip**, that is, changing the value $v \in \mathrm{var}(\varphi) \mapsto \varphi(v)$ to some $\varepsilon \in D_v \setminus \{\varphi(v)\}$, obtaining φ', such that all clauses of F satisfied by φ are also satisfied by φ' (note that this property holds automatically for the first type of change, the extension of φ). So if we have a sequence of conservative changes, then the corresponding sequence of sub-sets of satisfied clauses is monotonically increasing; especially if we start with a satisfying assignment, then all partial assignments in the sequence will also be satisfying assignments.

Theorem 4.7 *For a generalised multi-clause-set $F \in \mathcal{MCLS}$ and a partial assignment φ_0, in polynomial time a sequence of conservative changes w.r.t. F, starting* *with φ_0, can be computed such that the finally obtained partial assignment φ is matching-maximum for F.*

Before proving this theorem, we derive three corollaries.

Corollary 4.8 *For a satisfiable generalised multi-clause-set $F \in \mathcal{MCLS}$ there exists a satisfying assignment which is matching-maximum.*

We obtain the following generalisation of Theorem 7.16 in [31]:

Corollary 4.9 *For every satisfiable generalised multi-clause-set $F \in \mathcal{MCLS}$ there exists a partial assignment $\varphi \in \mathcal{PASS}$ with $n(\varphi) \leq \delta^*(F)$ such that $\varphi * F$ is matching satisfiable.*

Proof: By Corollary 4.8 there exists a satisfying assignment φ_0 for F and $F' \leq F$ with $c(F') = c(F) - \delta^*(F)$, such that φ_0 is matching-satisfying for F'. Let $F'' := F - F'$. We have $c(F'') = \delta^*(F)$ and φ_0 is satisfying for F'', so by Lemma 4.5, Part 1 there exists $\varphi \subseteq \varphi_0$ with $n(\varphi) \leq \delta^*(F)$ such that φ is satisfying for F''. Now $\varphi_0 = \varphi_0 \circ \varphi$ is matching-satisfying for F', and thus by Lemma 4.6 φ_0 is matching-satisfying for $\varphi * F' = \varphi * F' + \varphi * F'' = \varphi * (F' + F'') = \varphi * F$. ∎

Corollary 4.10 *The satisfiability problem for generalised multi-clause-sets F with $\delta^*(F) \leq k$ for constant $k \in \mathbb{N}_0$ is decidable in polynomial time (and if F is satisfiable, then a satisfying assignment can be computed). The algorithm runs through all partial assignments φ with $\mathrm{var}(\varphi) \subseteq \mathrm{var}(F)$ and $n(\varphi) \leq \delta^*(F)$ and checks whether $\varphi * F$ is matching satisfiable: If yes then for a matching satisfying assignment ψ for $\varphi * F$ we obtain a satisfying assignment $\varphi \circ \psi$ for F, while if no matching satisfiable sub-instance was found in this way, then F is unsatisfiable.*

In [49] it was shown, that in the boolean case the satisfiability problem for bounded maximal deficiency actually is fixed-parameter tractable. By reducing the general case to the boolean case, we will show fixed-parameter tractability in Theorem 5.5 also for generalised clause-sets.

The remainder of this subsection is devoted to the proof of Theorem 4.7 (generalising, simplifying and also strengthening the results on "admissible matchings" in [14]). In order to bring out the general structure of the proof we will present a more general result (which directly implies Theorem 4.7), exploring "parameterised maximum matching problems". We consider the situation where a fixed graph G is given together with an arbitrary set $\mathcal{P} \neq \emptyset$ of "parameters" (which in our case are partial assignments, while $G = B(F)$) and a mapping $\varphi \in \mathcal{P} \mapsto G_\varphi$, where G_φ is a partial graph of G (that is, $V(G_\varphi) = V(G)$ and $E(G_\varphi) \subseteq E(G)$). Let us call this parameterisation *matching-optimal*, if there exists some $\varphi \in \mathcal{P}$ such that $\nu(G_\varphi) = \nu(G)$. A matching-optimal parameterisation does not establish a method to find some ("good") $\varphi \in \mathcal{P}$ where $\nu(G_\varphi) = \nu(G)$ is attained. We consider the problem that we want to transform some arbitrary starting parameter φ_0 into such a good φ, and so we assume further that some relation $R \subseteq \mathcal{P} \times \mathcal{P}$ is given such that a relation $\varphi R \varphi'$ indicates an admissible move (in our application $\varphi R \varphi'$ holds if φ' results from φ by an conservative change). Denoting by R^* the reflexive-transitive hull of R (that is, allowing an arbitrary number of admissible moves), we call the parameterisation *strongly matching-optimal*, if for every $\varphi_0 \in \mathcal{P}$ there exists $\varphi \in \mathcal{P}$ with $\varphi_0 R^* \varphi$ and $\nu(G_\varphi) = \nu(G)$.

In the sequel we only consider bipartite G with a bipartition (A, B). We call the parameterisation $\mathcal{P}$ *conditional extensible* if for every $\varphi \in \mathcal{P}$, every matching M in G_φ and every edge $e \in E(G) \setminus M$ such that $M' := M \cup \{e\}$ is a matching in G and no matching M^* in G_φ with $M^* \supset M$ covers the endpoint of e in B, there exists $\varphi' \in \mathcal{P}$ with $\varphi R^* \varphi'$ such that M' is a matching in $G_{\varphi'}$.

Lemma 4.11 *If a parameterised matching problem $(G, \mathcal{P})$ is conditional extensible, then it is strongly matching-optimal. If furthermore in polynomial time in the size of G a sequence of admissible moves from φ to φ' in any conditional extension can be found, then in polynomial time (in the size of G) for any $\varphi_0 \in \mathcal{P}$ a sequence of admissible moves to $\varphi \in \mathcal{P}$ with $\nu(B_\varphi) = \nu(G)$ can be found.*

Before proving Lemma 4.11, we show that in our application, considering $B(F)$ with the parameterisation by $\varphi \in \mathcal{PASS} \mapsto B_\varphi(F)$ together with R as the relation of conservative change, the property of conditional extensibility holds true. So we consider the situation where we have a maximal matching M in $B_\varphi(F)$, where an edge $\{C, v\}$ in $B(F)$ exists with uncovered endpoints, and we show that by just one conservative change we can extend M by this additional edge.

Lemma 4.12 *Consider a generalised multi-clause-set $F \in \mathcal{MCLS}$, a partial assignment $\varphi \in \mathcal{PASS}$, a matching M in $B_\varphi(F)$ and an edge $\{(C_0, i), (v_0, j)\} \in E(B(F))$ such that neither $C := (C_0, i)$ nor $v := (v_0, j)$ is covered by M. We assume furthermore that no matching $M^* \supset M$ in $B_\varphi(F)$ covers v. Let $M' := M \cup \{\{C, v\}\}$. By definition M' is a matching in $B(F)$, and furthermore there exists a conservative change for φ w.r.t. F, resulting in φ', such that M' is a matching in $B_{\varphi'}(F)$.*

Proof: Let $(v_0, \varepsilon_0) \in C_0$. If $v_0 \notin \mathrm{var}(\varphi)$, then $\varphi' := \varphi \circ \langle v_0 \to \varepsilon \rangle$ for any $\varepsilon \in D_{v_0} \setminus \{\varepsilon_0\}$ yields the required conservative change; so assume $v_0 \in \mathrm{var}(\varphi)$. Now we have $\varphi(v_0) = \varepsilon_0$, since otherwise M' would be a matching in $B_\varphi(F)$ covering v. Let E be the set of values $\varepsilon \in D_{v_0}$ occurring in M, that is, there is some edge $\{(C'_0, i'), (v_0, j')\} \in M$ with $(v_0, \varepsilon) \in C'_0$. Since v is not covered by M and M is a matching, M can cover at most $(|D_{v_0}| - 1) - 1$ many variable-nodes with underlying variable v_0, and so we have $|E| \leq |D_{v_0}| - 2$. Thus there is $\varepsilon' \in D_{v_0} \setminus \{\varepsilon_0\}$. Set $\varphi' := \varphi \circ \langle v_0 \to \varepsilon' \rangle$. By definition M' is a matching in $B_{\varphi'}(F)$. Now consider a clause $D \in F$ falsified by φ', and assume that D is not falsified by φ. Thus $(v_0, \varepsilon') \in D$, and the literal (v_0, ε') is the only literal in D satisfied by φ. So no clause-node covered by M has clause D associated with it. It follows that $M^* := M \cup \{\{(D, 1), v\}\}$ would be a matching in $B_\varphi(F)$ extending M and covering v, contradicting the assumption of the assertion. ∎

Thus by Lemma 4.11 now Theorem 4.7 is proven. In the remainder of this subsection we prove Lemma 4.11. The reader might recall the preliminaries on matchings (Subsection 2.2), where the notion of an M-augmenting path P for a matching M in a graph G is discussed; a larger matching M^+ is obtained by adding the edges from P to M, which are not in M, while removing the other edges of P from M. In order to perform the "relinking", necessary for the transition from M to M^+, we show an auxiliary lemma.

Lemma 4.13 *As in Lemma 4.11 consider a conditional extensible parameterised matching problem $(G, \mathcal{P})$, where (A, B) is a bipartition of G. Consider furthermore $\varphi \in \mathcal{P}$, a maximal matching M in G_φ, an edge $e = \{a, b\} \in M$ with $a \in A, b \in B$, and an edge $e' = \{a, b'\} \in E(G)$ where b' is not covered by M. Then by at most* *one conditional-extension-step we obtain a parameter $\varphi' \in \mathcal{P}$ with $\varphi R^* \varphi'$ such that $M' := (M \setminus \{e\}) \cup \{e'\}$ is a matching in $G_{\varphi'}$.*

Proof: If M' is a matching in G_φ, then we are done. Otherwise we have $e' \notin E(G_\varphi)$; let $M_0 := M \setminus \{e\}$. We want to apply conditional extension to M_0 and $M' = M_0 \cup \{e'\}$ (if we succeed then we are done), so we have to show that there is no matching M_0^* in G_φ covering b' with $M_0 \subset M_0^*$. Assume the contrary, and consider the edge $\{x, b'\} \in M_0^*$: Since G is a graph (no parallel edges) we have $x \neq a$, and thus $M \cup \{\{x, b'\}\}$ is a matching in G_φ (using bipartiteness of G) contradicting maximality of M. ∎

Now we are in a position to prove Lemma 4.11; it suffices to show that for any given $\varphi \in \mathcal{P}$ and a matching M in G_φ with $|M| < \nu(G)$, by a polynomial number of extension steps we can find φ' with $\varphi R^* \varphi'$ and a matching M' in $G_{\varphi'}$ with $|M'| > |M|$ (by repeating this process we finally obtain a maximum matching for G). If M is not maximal in G_φ, then we can add one edge while keeping φ and we are done, so assume that M is maximal in G_φ. Since M is not a maximum matching in G, there exists an M-augmenting path P in G. P is of the form $(v_0, \dots, v_m)$ for (pairwise) different vertices v_i and m odd, such that v_0, v_m are not covered by M and such that for $0 \le i < m$ we have $\{v_i, v_{i+1}\} \notin M$ for even i, while for odd i we have $\{v_i, v_{i+1}\} \in M$. W.l.o.g. $v_i \in B$ for all even i. The first task is for odd $i < m$ to replace the edge $\{v_i, v_{i+1}\}$ by $\{v_i, v_{i-1}\}$, using Lemma 4.13; we proceed consecutively for $i = 1, 3, \dots$, where if at some point Lemma 4.13 is not applicable, then we constructed a matching of the same size as M which is not maximal w.r.t. its parameter, and so we obtain M' by enlarging this matching. Otherwise, if the process goes through, then at the end we obtain a matching M_0' in $G_{\varphi_0'}$ where $|M_0'| = |M|$ and v_{m-1}, v_m are not covered by M_0'. Again, if M_0' is not maximal, then we get the required larger M', while otherwise we can apply conditional extension, obtaining $M' := M_0' \cup \{\{v_{m-1}, v_m\}\}$. QED

4.3 Matching autarkies for generalised clause-sets

A partial assignment φ is called a **matching autarky** for $F \in \mathcal{MCLS}$ if φ is matching-satisfying for $F_{\mathrm{var}(\varphi)}$, which is equivalent to φ being matching-satisfying for $F[\mathrm{var}(\varphi)]$. The set of all matching autarkies for F is denoted by $\mathbf{MAuk}(\boldsymbol{F})$. Generalising Lemma 7.1 and the remarks in Section 8 of [31] we get

Lemma 4.14 *It is $F \in \mathcal{MCLS} \mapsto \mathrm{MAuk}(F) \subseteq \mathrm{Auk}(F)$ a normal autarky system.*

We denote by $\mathbf{N_{ma}} := \mathrm{N}_{\mathrm{MAuk}}$ the normal form for multi-clause-sets obtained by eliminating all matching autarkies. According to our general results and definitions on autarky systems, the set of MAuk-satisfiable multi-clause-sets is just $\mathcal{MSAT}$, the set of matching satisfiable multi-clause-sets. The set of MAuk-lean clause-sets is denoted by $\boldsymbol{\mathcal{MLEAN}}$, its elements are called **matching lean** multi-clause-sets. We now seek to characterise $\mathcal{MLEAN}$, and to compute $\mathrm{N}_{\mathrm{ma}}(F)$ in polynomial time.

A sub-multi-clause-set $F' \le F$ of a multi-clause-set $F \in \mathcal{MCLS}$ is called **tight** if $\delta(F') = \delta^*(F)$ holds. If F' is tight for F, then F' is an induced sub-multi-clause-set of F. By supermodularity of the deficiency (for graphs) we immediately get

Lemma 4.15 *Union and intersection of tight sub-multi-clause-sets of multi-clause-sets are again tight. So the tight sub-clause-sets of a clause-set form a set-lattice with smallest and largest element.*

Generalising Lemma 7.3 in [31], we obtain the fundamental relationship between tight sub-multi-clause-sets and matching autarkies, namely that application of matching autarkies does not reduce the deficiency, and application of suitable matching autarkies also allows to realise the maximal deficiency.

Lemma 4.16 *Consider a generalised multi-clause-set $F \in \mathcal{MCLS}$.*

1. *For every autarky φ for F we have $\delta(\varphi * F) = \delta(F) - \delta(F[\mathrm{var}(\varphi)])$.*
2. *For every matching autarky φ for F we have $\delta(\varphi * F) \geq \delta(F)$, and thus $\delta^*(\varphi * F) = \delta^*(F)$.*
3. *Consider an induced sub-multi-clause-set F' of F.*
   (a) *$\delta^*(\mathrm{var}(F') * (F - F')) \leq \delta^*(F) - \delta(F')$.*
   (b) *If F' is tight, then there is a matching autarky φ for F with $\varphi * F = F'$.*

Proof: For Part 1 note that by definition we have

$$c(F) = c(\varphi * F) + c(F[\mathrm{var}(\varphi)]),\ n(F) = n(\varphi * F) + n(F[\mathrm{var}(\varphi)])$$

due to $F = \varphi * F + F_{\mathrm{var}(\varphi)}$. Part 2 follows from Part 1. For Part 3a consider $G \leq \mathrm{var}(F') * (F - F')$. There exists $G_0 \leq F - F'$ with $\mathrm{var}(F') * G_0 = G$. Now

$$\delta^*(F) \geq \delta(F' + G_0) = c(F' + G_0) - \mathrm{wn}(F' + G_0) = \\ c(F') + c(G) - \mathrm{wn}(F') - \mathrm{wn}(G) = \delta(F') + \delta(G),$$

and thus $\delta(G) \leq \delta^*(F) - \delta(F')$. Now Part 3b follows immediately from Part 3a due to $\delta^*(\mathrm{var}(F') * (F - F')) \leq \delta^*(F) - \delta(F') = 0$, i.e., $F - F'$ is a matching autark sub-multi-clause-set of F. ∎

Generalising Theorem 7.5 in [31], we now can characterise matching lean multi-clause-sets:

Lemma 4.17 *Consider a generalised multi-clause-set $F \in \mathcal{MCLS}$. The following conditions are equivalent:*

1. *F is matching lean;*
2. *$\forall\, C \in F : \delta^*(F - \{C\}) < \delta^*(F)$;*
3. *$\forall\, F' \lneq F : \delta(F') < \delta(F)$;*
4. *F is a tight sub-multi-clause-set of F, and there are no other tight sub-multi-clause-sets of F.*

Proof: From Part 1 follows Part 4 by Lemma 4.16, Part 3b. Obviously, Part 4 implies Part 3, and Part 3 implies Part 2. Finally, Part 1 follows from Part 2 by Lemma 4.16, Part 2. ∎

By Lemma 4.17, Part 2 we get

Corollary 4.18 *It is decidable in polynomial time, whether a generalised multi-clause-set $F \in \mathcal{MCLS}$ is matching lean.*

Thus by Lemma 3.1:

Corollary 4.19 *The matching lean kernel* $\mathrm{N}_{\mathrm{ma}}(F)$ *for generalised multi-clause-sets* $F \in \mathcal{MCLS}$ *is computable in polynomial time.*

By Lemma 4.17, Part 4 together with Lemma 4.16, Part 3b we get

Corollary 4.20 *For every generalised multi-clause-set* $F \in \mathcal{MCLS}$ *the lean kernel* $\mathrm{N}_{\mathrm{ma}}(F)$ *is the intersection of all tight sub-multi-clause-sets of* F*. Thus* $\mathrm{N}_{\mathrm{ma}}(F)$ *is the smallest tight sub-multi-clause-set of* F*, and therefore* $\delta^*(F) = \delta(\mathrm{N}_{\mathrm{ma}}(F))$*.*

Using $\delta(\top) = 0$, from Lemma 4.17, Part 3 we get the following generalisation of "Tarsi's Lemma" (see [1]):

Corollary 4.21 *If the generalised multi-clause-set* $F \neq \top$ *is matching lean, then* $\delta^*(F) = \delta(F) \geq 1$*.*

Obviously $\mathcal{MUSAT} \subset \mathcal{LEAN}$, and thus:

Corollary 4.22 *If a generalised clause-set* $F \in \mathcal{CLS}$ *is minimally unsatisfiable, then we have* $\delta^*(F) = \delta(F) \geq 1$*.*

In [9], Theorem 4.5, arbitrary constraints over boolean variables are considered, and a lower bound on the number of clauses in terms of the number of variables for minimally unsatisfiable constraint satisfaction problems is derived, which necessarily is much weaker than Corollary 4.22. Considering minimally unsatisfiable clause-sets of minimal deficiency, we observe that removing any clause from a matching lean multi-clause-set F with $\delta(F) = 1$ yields a matching satisfiable multi-clause-set, and thus

Corollary 4.23 $\mathcal{MUSAT}_{\delta=1} = \mathcal{MLEAN}_{\delta=1} \cap \mathcal{USAT}$*.*

The class $\mathcal{MUSAT}_{\delta=1}$ of minimally unsatisfiable clause-sets of minimal deficiency is characterised in Theorem 6.16, and so we have a good understanding of the unsatisfiable elements of $\mathcal{MLEAN}_{\delta=1}$. The satisfiable elements of $\mathcal{MLEAN}_{\delta=1}$ on the other hand seem to have a more complicated nature, where the interest in this class may be justified by the following property.

Corollary 4.24 *The class* $\mathcal{MLEAN}_{\delta=1}$ *of matching lean generalised clause-sets of deficiency* 1 *is exactly the class of all minimally matching unsatisfiable clause-sets (clause-sets which are not matching satisfiable, while every strict subset is matching satisfiable).*

We conclude this subsection with an interesting example for a satisfiable boolean $F \in \mathcal{MLEAN}_{\delta=1}$ with $n(F) = 8$, exhibited in Section 5 of [33], whose clause-variable matrix is as follows (the rows correspond to the clauses, the columns to the variables, where an entry "$\pm$" denotes a positive/negative occurrence, while 0 denotes non-occurrence):

$$M(F) = \begin{pmatrix} + & 0 & + & 0 & 0 & 0 & 0 & + \\ + & 0 & - & + & 0 & 0 & 0 & 0 \\ - & - & 0 & 0 & 0 & + & + & 0 \\ - & - & 0 & 0 & - & - & 0 & 0 \\ 0 & + & - & - & - & 0 & 0 & 0 \\ 0 & 0 & 0 & - & + & 0 & - & - \\ - & 0 & 0 & 0 & + & - & 0 & + \\ - & 0 & 0 & + & 0 & + & - & 0 \\ 0 & + & + & 0 & 0 & 0 & + & - \end{pmatrix}.$$

Obviously $\delta(F) = 1$. Since every two different rows clash in exactly one element, F is a 1-regular hitting clause-set. Every column contains at least two $-$ and two $+$, and thus every variable occurs at least two times negatively as well as positively (the purpose of this example was to refute a conjecture of Endre Boros, that every boolean 1-regular hitting clause-set of deficiency 1 must contain a variable occurring in one sign only once). To demonstrate that F is matching lean, consider the following subgraph of $B(F)$:

$$\begin{tikzcd}
 & & \mathrm{c}^6 \arrow[r, dash] & \mathbf{v_4} \arrow[r, dash] & \mathrm{c}^2 \arrow[r, dash] & \mathbf{v_1} \arrow[dr, dash] & \\
\mathrm{c}^3 \arrow[r, dash] & \mathbf{v_2} \arrow[r, dash] & \mathrm{c}^4 \arrow[r, dash] & \mathbf{v_5} \arrow[r, dash] & \mathrm{c}^5 \arrow[r, dash] & \mathbf{v_3} \arrow[r, dash] & \mathrm{c}^1 \\
\mathrm{c}^7 \arrow[r, dash] & \mathbf{v_6} \arrow[r, dash] & \mathrm{c}^8 \arrow[r, dash] & \mathbf{v_7} \arrow[r, dash] & \mathrm{c}^9 \arrow[r, dash] & \mathbf{v_8} \arrow[ur, dash] &
\end{tikzcd}$$

Here variable-nodes corresponding to row j are denoted by v_j, and clause-nodes corresponding to row i by c^i. This subgraph has the special properties, that it is a spanning tree, where the variable-nodes all have degree 2. From these properties by Corollary 4.26 in the next subsection it follows, that F is matching lean.[15)]

4.4 Expansion and the surplus

In this subsection we generalise the results from [49] related to the notions of "expansion" and "surplus", yielding also a simplified proof that satisfiability for boolean multi-clause-sets F is fixed-parameter tractable w.r.t. the parameter $\delta^*(F)$. The main tool here is a poly-time reduction $S : \mathcal{MCLS} \to \mathcal{MCLS}$ (see Lemma 4.33) with the properties, that the maximal deficiency is not increased and that we have for all variables $v \in \mathrm{var}(S(F))$ and all $\varepsilon \in D_v$ the upper bound

$$\delta^*(\langle v \to \varepsilon \rangle * S(F)) \leq \delta^*(S(F)) - 1$$

(see Corollary 4.29). Thus by a trivial DLL branching algorithm, using additionally only the reduction $F \mapsto S(F)$ at each node, we obtain SAT decision for F in time

[15)]This can also be seen directly as follows. For a graph G let $\delta(G) := |E(G)| - |V(G)|$. Using $\kappa(G)$ for the number of connected components of G, we have that G is a forest iff $\delta(G) = -\kappa(G)$ (while for every graph G we have $\delta(G) \geq -\kappa(G)$). Now consider any non-empty set V' of variable-nodes in the above forest $\mathcal{F}$ (that actually we have a tree is irrelevant) together with the induced sub-graph $\mathcal{F}'$ given by $V' \cup \Gamma_{\mathcal{F}}(V')$. Since also $\mathcal{F}'$ is a forest we have $\delta(\mathcal{F}') \leq -\kappa(\mathcal{F}') \leq -1$, where $|E(\mathcal{F}')| = 2|V'|$ and $|V(\mathcal{F}')| = |V'| + |\Gamma_{\mathcal{F}'}(V')|$, and thus $|\Gamma_{\mathcal{F}'}(V')| \geq |V'| + 1$, where $|\Gamma_{\mathcal{F}'}(V')| \leq \delta(F[V']) + |V'|$, so that $\delta(F[V']) \geq 1$. Since this holds for every non-empty V', it follows that F cannot have a non-trivial matching autarky.

$(\max_{v \in \mathrm{var}(F)} |D_v|)^{\delta^*(F)} \cdot \mathrm{poly}(\ell(F))$ (in Theorem 5.5 we actually obtain the upper bound $2^{\delta^*(F)} \cdot \mathrm{poly}(\ell(F))$ also for non-boolean clause-sets, exploiting the translation to boolean clause-sets).

For multi-clause-sets F we have defined the bipartite graph $B(F)$ together with its canonical bipartition $(\underline{F}, \underline{V})$. The general definition of deficiency (for arbitrary graphs) then yields the deficiency $\delta_{B(F)}(F') = |F'| - |\Gamma_{B(F)}(F')|$ for sets $F' \subseteq \underline{F}$ of clause-nodes, as well as the deficiency $\delta_{B(F)}(V') = |V'| - |\Gamma_{B(F)}(V')|$ for sets $V' \subseteq \underline{V}$ of variable-nodes. So we have a "clause-based deficiency" as well as a "variable-based deficiency". Identifying F' with a sub-multi-clause-set of F, we have $|F'| = c(F')$ and $\Gamma_{B(F)}(F') = \mathrm{wn}(F')$, and thus $\delta_{B(F)}(F')$ (the graph-theoretical deficiency of sets of clause-nodes) is the same as the deficiency of multi-clause-sets as we have defined it in Subsection 4.1. At first sight, the situation for V' seems not to be naturally interpretable on the level of multi-clause-sets, since V' may contain for some variable v only some of the $|D_v| - 1$ copies of v. To consider this problem, let V'' the the set of variable-nodes obtained from V' by adding for $(v, i) \in V'$ all (v, j) for $j \in \{1, \dots, |D_v| - 1\}$. Now we have

$$\delta_{B(F)}(V'') = -\delta(F[V_0']),$$

where V_0 is the set of variables in V' (or V''). Since $\delta_{B(F)}(V') \leq \delta_{B(F)}(V'')$ due to $\Gamma_{B(F)}(V') = \Gamma_{B(F)}(V'')$, we see that actually, since we are only interested in *maximising* the deficiency, also the variable-based deficiency has a sensible interpretation at the (conceptual) level of multi-clause-sets. Now two changes are applied to the variable-based deficiency, resulting in the notion of "expansion" related to deficiency of a set of variables (analogous to the deficiency of a (sub-)multi-clause-set), and in the notion of "surplus" related to the maximal deficiency over all sets of variables (analogous to the maximal deficiency of a multi-clause-set). The first change is just to switch signs, so that we can use $\delta(F[V_0'])$ instead of $-\delta(F[V_0'])$. More substantially, we exclude the empty set of variables for the surplus: The maximal deficiency $\delta^*(F)$ of a multi-clause-set is only used to determine the size of a maximal matching in $B(F)$, and so negative deficiencies are not of interest (they indicate that a bigger matching number is possible — if only there would be more clauses), whence the empty clause-set is taken into account in $\delta^*(F)$ for convenience, to force the maximal deficiency to be at least 0. But now for the notion of surplus actually we are only interested in the negative values, that is, in the "surplus" which cannot be realised, and thus the empty set of variables has to be excluded. After these motivations, let us now start with the formal definitions.

For a multi-clause-set F and a set V of variables let the **expansion** be defined as $\delta(F[V])$. As explained above, using the deficiency $\delta_{B(F)}$ in the (bipartite) graph $B(F)$, the expansion equals $-\delta_{B(F)}(V')$, where V' is the set of variable nodes of $B(F)$ associated with some variable in V. The **surplus** of F is defined as

$$\boldsymbol{\sigma(F)} := \begin{cases} \min_{\emptyset \neq V \subseteq \mathrm{var}(F)} \delta(F[V]) & \text{if } \mathrm{var}(F) \neq \emptyset \\ 0 & \text{if } \mathrm{var}(F) = \emptyset \end{cases}.$$

The surplus of F equals the surplus of $B(F)$ as defined in Subsection 1.3 of [44] (but with the sides of the bipartition switched(!)). By definition we have $\sigma(F) \leq \delta(F[\mathrm{var}(F)]) = \delta(F) - F(\bot) \leq \delta(F)$. Generalising Lemma 7.7 in [31] we have:

Lemma 4.25 *A generalised multi-clause-set $F \in \mathcal{MCLS} \setminus \{\top\}$ is matching lean if and only if $\sigma(F) \geq 1$. More specifically, for any generalised multi-clause-set F we have:*

For $\emptyset \neq V \subseteq \mathrm{var}(F)$ in case of $\delta(F[V]) \leq 0$ it has $F[V]$ a non-trivial matching autarky; such a non-trivial matching autarky yields a non-trivial matching autarky for F, and every non-trivial matching autarky of F can be obtained in this way.

Proof: If F is matching lean, then by Lemma 4.14 for $V \subseteq \mathrm{var}(F)$ also $F[V]$ is matching lean, and thus by Corollary 4.21 we get $\sigma(F) \geq 1$. If on the other hand F is not matching lean, then there exists $\emptyset \neq V \subseteq \mathrm{var}(F)$ such that $F[V]$ is matching satisfiable (where V is the variable set of any non-trivial matching autarky), i.e., $\delta^*(F[V]) = 0$, and thus $\sigma(F) \leq 0$. ∎

By Theorem 1.3.8 in [44] we obtain the following characterisation of matching lean clause-sets:

Corollary 4.26 *A generalised multi-clause-set F is matching lean if and only if there exists a subgraph $\mathcal{F}$ of $B(F)$ with the following properties:*

(i) $\mathcal{F}$ is a forest;

(ii) $\mathcal{F}$ covers all variable-nodes;

(iii) every variable-node has degree 2 in $\mathcal{F}$.

An example for the application of Corollary 4.26 has been given at the end of Subsection 4.3. The problem of computing $\sigma(F)$ can be solved by Theorem 1.3.6 in [44] as follows (compare Lemma 15 in [49] for the case of boolean clause-sets):

Lemma 4.27 *Consider a generalised multi-clause-set F and a maximum matching M for $B(F)$.*

1. *If M does not cover all variable-nodes, then $\sigma(F) = |M| - \mathrm{wn}(F) < 0$ (that is, $\sigma(F)$ is the number of uncovered variable-nodes, multiplied by -1). Otherwise we have $\sigma(F) \geq 0$.*
2. *Assume $\sigma(F) \geq 0$, and consider $s \in \mathbb{N}_0$, $s \leq \delta(F)$. For $v \in \mathrm{var}(F)$ with $|D_v| \geq 2$ (trivial variables are ignored) let $F_{s,v}$ be the multi-clause-set obtained from F by adding s new elements to the domain of v (that is, a new variable v' with $D_v \subseteq D_{v'}$ and $|D_{v'}| = |D_v| + s$ is chosen, and $F_{s,v}$ is obtained by replacing v by v' in F). Let $M_{s,v}$ be a maximum matching in $B(F_{s,v})$. Then we have:*
   - *(a) If $M_{s,v}$ does not cover all variable-nodes in $B(F_{s,v})$, then $\sigma(F) < s$, and moreover, from $M_{s,v}$ in linear time in $\ell(F)$ a set $\emptyset \neq V \subseteq \mathrm{var}(F)$ with $\delta(F[V]) < s$ can be computed .*
   - *(b) If for all v the maximum matching $M_{s,v}$ covers all variable-nodes in $B(F_{s,v})$, then $\sigma(F) \geq s$.*

Lemma 4.25 together with Lemma 4.27 yields an alternative to the computation of $\mathrm{N}_{\mathrm{ma}}(F)$ as given in Corollary 4.19. The next lemma tackles the problem of giving a sufficient criterion for $\delta^*(\langle v \to \varepsilon \rangle * F) < \delta^*(F)$; Part 3 generalises Lemma 7.10 in [31] (the proof there is technically not fully correct).

Lemma 4.28 *Consider a generalised multi-clause-set $F \in \mathcal{MCLS}$.*

1. *For $F' \leq F$ we have*
$$\delta(F') = \delta(F) - \delta(F[\mathrm{var}(F) \setminus \mathrm{var}(F')]) - \eta(F, F') \\ \leq \delta(F) - \delta(F[\mathrm{var}(F) \setminus \mathrm{var}(F')]),$$
where $\eta(F, F') := c(F) - c(F_{\mathrm{var}(F)\setminus\mathrm{var}(F')}) - c(F') \geq 0$ is the number of clause occurrences in F of clauses C with $\mathrm{var}(C) \subseteq \mathrm{var}(F')$ but not occurring in F'.
2. *For $F' \leq F$ we have $\delta(F') \leq \delta(F) - \min(c(F) - c(F'), \sigma(F))$.*
3. *Consider $v \in \mathrm{var}(F)$. Then for $\varepsilon \in D_v$ we have*
$$\delta^*(\langle v \to \varepsilon\rangle * F) \leq \delta(F) - \min(s_{(v,\varepsilon)}(F), \sigma(F)) + |D_v| - 1.$$
Thus, using $m_v(F) := \min_{\varepsilon \in D_v} s_{(v,\varepsilon)}(F)$, we obtain
$$\delta^*(\langle v \to \varepsilon\rangle * F) \leq \delta(F) - \min(m_v(F), \sigma(F)) + |D_v| - 1.$$

Proof: For Parts 1 and 2 let $V := \mathrm{var}(F) \setminus \mathrm{var}(F')$.

The equation in Part 1 follows immediately with $\delta(F) = c(F) - \mathrm{wn}(F)$ and $\delta(F[V]) = c(F[V]) - \mathrm{wn}(F[V])$, where due to $\mathrm{var}(F[V]) = V$ we have $\mathrm{wn}(F[V]) = \mathrm{wn}(F) - \mathrm{wn}(F')$. And that $\eta(F, F') \geq 0$ holds follows with the explanation given.

For Part 2 we consider two cases. If $n(F') = n(F)$, then $\delta(F') = \delta(F) - (c(F) - c(F'))$. So assume $n(F') < n(F)$ (and thus $V \neq \emptyset$). By Part 1 we have $\delta(F') \leq \delta(F) - \delta(F[V])$, and thus by $\delta(F[V]) \geq \sigma(F)$ we get $\delta(F') \leq \delta(F) - \sigma(F)$.

For Part 3 consider an induced $F' \leq \langle v \to \varepsilon\rangle * F$, and let $F'' \leq F$ be the unique sub-multi-clause-set of F with $c(F'') = c(F')$ and $\langle v \to \varepsilon\rangle * F'' = F'$. We have $\mathrm{wn}(F'') \leq \mathrm{wn}(F') + |D_v| - 1$, and thus $\delta(F'') \geq \delta(F') - |D_v| + 1$.

By part 2 we get $\delta(F'') \leq \delta(F) - \min(c(F) - c(F''), \sigma(F))$, where $c(F'') \leq c(F) - s_{(v,\varepsilon)}(F)$, and the assertion follows. ▮

For a matching lean boolean clause-set F, Part 3 of Lemma 4.28 yields the upper bound $\delta^*(\langle v \to \varepsilon\rangle * F) \leq \delta(F)$ for non-pure variables v (using Lemma 4.25); we are interested in cases where the maximal deficiency actually decreases:

Corollary 4.29 *If for a generalised clause-set F and a variable $v \in \mathrm{var}(F)$ we have $m_v(F) \geq |D_v|$ (using the definition of $m_v(F)$ as in Lemma 4.28, Part 3) and $\sigma(F) \geq |D_v|$, then we have $\delta^*(\langle v \to \varepsilon\rangle * F) \leq \delta(F) - 1$.*

It is unclear, whether Corollary 4.29 is best possible — the condition $\sigma(F) \geq |D_v|$ is hard to establish for larger domain sizes. The key seems to be to improve the estimation used in the proof of Lemma 4.28, Part 3. By singular DP-reduction we can eliminate cases with $m_v < |D_v|$ as follows.

Lemma 4.30 *Consider a generalised multi-clause-set $F \in \mathcal{MCLS}$ and $v \in \mathrm{var}(F)$.*

1. *Assume v is not pure in F. Then v is a singular DP-variable for F if and only if $m_v(F) < |D_v|$ (where $m_v(F)$ is as defined in Part 3 of Lemma 4.28).*
2. *Assume v is a singular DP-variable.*
   (a) *$\delta(\mathrm{DP}_v(F)) \leq \delta(F)$.*
   (b) *If v is non-degenerated, then we have*

i. $\delta(\mathrm{DP}_v(F)) = \delta(F)$.

ii. *F is matching lean if and only if $\mathrm{DP}_v(F)$ is matching lean.*

iii. *If F is matching lean, then we have $\delta^*(\mathrm{DP}_v(F)) = \delta^*(F)$.*

Proof: The only non-obvious assertion is 2(b)ii (using the remarks made before Lemma 3.4, and Lemma 4.17). Let $D_v = \{\varepsilon_1, \dots, \varepsilon_k\}$ ($k = |D_v|$), and assume w.l.o.g. that $\#_{(v,\varepsilon_i)}(F) = 1$ for $i < k$; consider $C_1, \dots, C_{k-1} \in F$ with $(v, \varepsilon_i) \in C_i$ for $i < k$, and let $D_1, \dots, D_m \in F$ be the clauses containing (v, ε_k) ($m = \#_{(v,\varepsilon_k)}(F)$). Thus $F_{\{v\}} = \{C_1, \dots, C_{k-1}\} + \sum_{i=1}^m \{D_i\}$. Now with $F' := F - F_{\{v\}}$ we have

$$\mathrm{DP}_v(F) = F' + R,$$

where $R := \sum_{i=1}^m \{\, R_i \,\}$ and $R_i := C_1 \cup \dots \cup C_{k-1} \cup D_i$ for $i \in \{1, \dots, m\}$. First assume that F is matching lean, but that we have a non-trivial matching autarky φ for $\mathrm{DP}_v(F)$ with $\mathrm{var}(\varphi) \subseteq \mathrm{var}(\mathrm{DP}_v(F)) = \mathrm{var}(F) \setminus \{v\}$. Let $V := \mathrm{var}(\varphi)$. If $V \cap \mathrm{var}(R) = \emptyset$, then φ would also be a matching autarky for F, since $\mathrm{var}(R) = \mathrm{var}(F_{\{v\}}) \setminus \{v\}$. So assume $V \cap \mathrm{var}(R) \neq \emptyset$. If there exists $i \in \{1, \dots, m\}$ with $\mathrm{var}(D_i) \cap V = \emptyset$, then for all $j < k$ we have $\mathrm{var}(C_j) \cap V = \emptyset$, and it follows, that φ would also be a matching autarky for F. So assume that for all $i \in \{1, \dots, m\}$ we have $\mathrm{var}(D_i) \cap V \neq \emptyset$. Now $\langle v \to \varepsilon_k \rangle \circ \varphi$ is a matching autarky for F, contradicting matching leanness of F.

For the reverse direction assume that $\mathrm{DP}_v(F)$ is matching lean, but that we have a non-trivial matching autarky φ for F. Now it is not hard to see that φ is also a matching autarky for $\mathrm{DP}_v(F)$. ∎

Corollary 4.31 *There is a polynomial time computable map $r : \mathcal{MCLS} \to \mathcal{MCLS}$, such that for a generalised multi-clause-set F we have:*

(i) *$n(r(F)) \le n(F)$, $c(r(F)) \le c(F)$ and $\delta^*(r(F)) \le \delta^*(F)$.*

(ii) *$r(F)$ is satisfiability-equivalent to F.*

(iii) *$r(F)$ is matching lean.*

(iv) *$r(F)$ is lean w.r.t. pure autarkies (i.e., $r(F)$ does not contain pure variables).*

(v) *$r(F)$ does not contain singular DP-variables.*

Computation of $r(F)$ is as follows:

1. *Apply singular DP-degeneration reduction and reduction by pure autarkies and matching autarkies as long as possible.*
2. *If there exists a singular DP-variable, then it must be non-degenerated, thus applying DP-reduction does not increase the maximal defect by Part 2(b)iii of Lemma 4.30, so apply this reduction and go to Step 1. Otherwise output $r(F)$ and stop.*

The next lemma contains the main idea for establishing a surplus of two.

Lemma 4.32 *Consider a multi-clause-set $F \in \mathcal{MLEAN}$, such that for all variables $v \in \mathrm{var}(F)$ we have $\#_v(F) \ge |D_v| + 1$, and assume that $V \subseteq \mathrm{var}(F)$ is given with $\delta(F[V]) = 1$. Then $F[V]$ is satisfiable, and a satisfying assignment φ with $\mathrm{var}(\varphi) \subseteq \mathrm{var}(F[V])$ can be found in polynomial time. (With φ thus we have found a non-trivial autarky for F.)*

Proof: Assume that $F[V]$ is unsatisfiable. By Corollary 4.23 thus we have $F[V] \in \mathcal{MUSAT}_{\delta=1}$ which would contain a variable occurring in all signs exactly once (see the later Lemma 6.15, whose proof does not make use of the results of this subsection) contradicting the assumption. So $F[V]$ is satisfiable, and a satisfying assignment φ can be found by Corollary 4.10. ∎

Strengthening Corollary 4.31 we get now:

Lemma 4.33 *There is a polynomial time computable map $S : \mathcal{MCLS} \to \mathcal{MCLS}$, such that for a generalised multi-clause-set F we have:*

(i) $n(S(F)) \le n(F)$, $c(S(F)) \le c(F)$ *and* $\delta^*(S(F)) \le \delta^*(F)$.

(ii) $S(F)$ *is satisfiability-equivalent to* F.

(iii) $S(F)$ *is matching lean and lean w.r.t. pure autarkies, and does not contain singular DP-variables.*

(iv) If $\operatorname{var}(S(F)) \neq \emptyset$, *then* $\sigma(S(F)) \ge 2$.

Computation of $S(F)$ is as follows:

1. *First reduce $F := r(F)$ (see Corollary 4.31).*
2. *If* $\operatorname{var}(F) = \emptyset$ *or* $\sigma(F) \ge 2$ *then stop.*
3. *Otherwise find some* $\emptyset \neq V \subseteq \operatorname{var}(F)$ *with* $\delta(F[V]) = \sigma(F) = 1$ *by Lemma 4.27, Part 2; by Lemma 4.32 we can now find a non-trivial autarky φ for F: reduce $F := \varphi * F$, and go to Step 1.*

If we only allow boolean clause-sets, then, as explained at the beginning of this subsection, we obtain fixed-parameter tractability of satisfiability decision w.r.t. the parameter $\delta^*(F)$ by Lemma 4.33 and Corollary 4.29 (together with Lemma 4.30, Part 1). Finally we mention, that a good possibility for further improvements is to generalise and strengthen the approach from [26] based on matroid theory.

4.5 Comparison with an earlier version of "matching autarkies"

In [30] an earlier version of matching autarkies has been introduced, which we will call here "non-repetitive matching autarkies": A partial assignment φ is called *non-repetitive matching satisfying* for a multi-clause-set $F \in \mathcal{MCLS}$, if for every clause-occurrence C in F (taking multiple occurrences into account) a literal $x_C \in C$ can be chosen with $\varphi(x_C) = 1$ such that for different clause-occurrences C, C' we have $x_C \neq x_{C'}$. And φ is called a *non-repetitive matching autarky* for F if φ is non-repetitive matching satisfying for $F_{\operatorname{var}(\varphi)}$.

Recalling the three conditions (i) - (iii) from Subsection 4.1 and strengthening condition (i) to

(i)' for $i \in \{1, \dots, m\}$ there are variables $v_i \in \operatorname{var}(F_i)$ such that for all clause-occurrence C in F_i there are literals $x_C \in C$ with $\operatorname{var}(x_C) = v_i$, and such that for different clause-occurrences C, C' we have $x_C \neq x_{C'}$;

we get that F is non-repetitive matching satisfiable iff F has a decomposition fulfilling (i)', (ii) and (iii). By (i)' we get $c(F_i) = |\mathrm{val}_{v_i}(F_i)|$, and thus from (iii) follows (iii)'. Whence a non-repetitive matching satisfying assignment φ for F is matching satisfying for F, and a non-repetitive matching autarky for F is also a matching autarky for F.

For boolean clause-sets non-repetitive matching autarkies are the same as matching autarkies, but in general non-repetitive matching autarkies are more restrictive than matching autarkies; examples for (multi-)clause-sets F_1, F_2 which are matching satisfiable but are lean w.r.t. non-repetitive matching autarkies are discussed in Subsection 5.3. These examples actually show that non-repetitive matching autarkies are preserved by the standard translation of (generalised) clause-sets into boolean clause-sets, which in general is not the case for matching autarkies, and so perhaps non-repetitive matching autarkies are preferable over matching autarkies?

The main problem with the notion of non-repetitive matching autarkies is that it does not seem to support a natural notion of related deficiency (with the same nice properties as the combination of matching autarkies and (standard) deficiency), and, related to this problem, it does not seem obvious how to achieve polynomial time decision of the class of non-repetitive matching lean (multi-)clause-sets. The whole problem boils down to the point, that non-repetitive matching autarkies do not seem to be given solely by a matching condition, but require some other form of a more global condition. Thus, to conclude, the generalisation of (boolean) matching autarkies together with the generalisation of (boolean) deficiency introduced in this section seems to be the right choice, as demonstrated by the theory build up in this section, and as further validated by the results in the following sections on the standard translation and on minimally unsatisfiable clause-sets of deficiency one.

5 Translating generalised clause-sets into boolean clause-sets

In this section we investigate translations of generalised clause-sets into boolean clause-sets. Different from previous research (for an overview see [47]), here we are not interested in experimental results (and how good different translations perform in various experiments for different SAT solvers), but we are interest in *structure-preserving* translations. At least regarding our focus on (matching) autarkies and the deficiency, the only reasonable possibility here seems to be what in [47] has been coined the "multivalued encoding", which is the "standard translation", but without AMO ("at most one") clauses (since these many binary clauses would destroy the combinatorial structures we are considering):

- For every literal (v, ε) we consider a boolean variable $\tau((v, \varepsilon))$ expressing that v shall not get value ε.[16)]
- Clauses C are translated into (positive) boolean clauses $\tau(C)$ by replacing each literal $x \in C$ with the (positive) boolean literal $\tau(x)$.
- We add "ALO clauses" requiring that each variable gets at least one value (if it gets more than one value, then one of the values can be chosen).

[16)] In the literature typically the variables denote "v shall get value ε", which results only in flipping signs here, but as hopefully this articles helps to point out, for conjunctive normal form *falsity* is the norm (while for *disjunctive* normal forms verity is the norm), and thus our choice.

In [30] in Subsection 4.5 ("An autarky preserving reduction to boolean clause-sets") it has already been stated that the standard translation not only preserves satisfiability, unsatisfiability and minimally unsatisfiability, but also leanness. We have to expand these results especially regarding the notions of matching autarkies and deficiency, since in [30] only a restricted notion of "matching autarkies" has been used (recall Subsection 4.5) without an associated notion of deficiency

Another source relevant here is [2], where "monosigned CNF formulas" are translated, a generalisation of "generalised clause-sets" allowing also to express that a variable must get a certain value; in other words, where our literals (v, ε) express "$v \neq \varepsilon$", for monosigned formulas also "positive" literals "$v = \varepsilon$" are allowed. This generalisation can be motivated by the fact, that these formulas are exactly those which can be translated by the standard translation; however the price which have to be paid here is that now the AMO clauses are necessary in the standard translation! This adds further to the point we want to make, that generalised clause-sets in our definition (allowing only "negative literals") are the appropriate generalisation of boolean conjunctive normal forms, while further generalisations (like "monosigned formulas") enter new areas, where the combinatorics of clause-sets no longer can be applied. For a local search algorithm working directly with "monosigned CNF formulas" see [18] (using the notion of "nb-formulas" (for "non-boolean")).

It is worth to mention here, that in [45] it has been shown, that resolution which works only with generalised clause-sets, that is, where in the corresponding branching approach for a variable v only a branching of width $|D_v|$ assigning in each branch one of the possible values to v (see [36]) is considered, can be exponentially worse than resolution on the translation into boolean logic, where now branchings "v gets value ε" and "v does *not* get value ε" are possible. From this is follows that generalised DPLL-algorithms should not be restricted to branchings where in each branch a variable needs to be fixed to some value; however the focus of this article is not generalisation of SAT solvers, but generalisation of *combinatorial structure*, and thus we do not further pursue these (important) investigations.

5.1 The details of the translation

Formally, the translation proceeds as follows. We consider some bijection $\tau : \mathcal{LIT} \to \mathcal{VA}_{\{0,1\}}$ from the set of all (generalised) literals to the set of all boolean variables.[17] The intended meaning of the (positive) boolean literal $\tau((v, \varepsilon))$ for a literal $(v, \varepsilon) \in \mathcal{LIT}$ is the same as the interpretation of the original (generalised) literal, namely "v shall not get value ε". We obtain an injection $\tau : \mathcal{CL} \to \mathcal{CL}(\mathcal{VA}_{\{0,1\}})$ by setting $\tau(C) := \{\tau(x) : x \in C\}$ for $C \in \mathcal{CL}$. Actually $\tau : \mathcal{CL} \to \mathcal{CL}(\mathcal{VA}_{\{0,1\}})$ constitutes a bijection from $\mathcal{CL}$ to the set of all positive boolean clauses. The translation τ can be further extended to an injection $\tau : \mathcal{CLS} \to \mathcal{CLS}(\mathcal{VA}_{\{0,1\}})$ by $\tau(F) := \{\tau(C) : C \in F\}$ for $F \in \mathcal{CLS}$. Again, $\tau : \mathcal{CLS} \to \mathcal{CLS}(\mathcal{VA}_{\{0,1\}})$ constitutes a bijection from the set of (generalised) clause-sets to the set of boolean clause-sets containing only positive clauses. Finally, for $v \in \mathcal{VA}$ let

$$\mathbf{ALO}_{\boldsymbol{v}} := \{\overline{\tau((v, \varepsilon))} : \varepsilon \in D_v\} \in \mathcal{CL}(\mathcal{VA}_{\{0,1\}})$$

be the (negative, boolean) clause expressing that v gets assigned at least one of the values $\varepsilon \in D_v$ (that is, not all (positive) literals $\tau((v, \varepsilon))$ for $\varepsilon \in D_v$ can be true),

[17] Such a bijection exists due to our assumption on $\mathcal{VA}$, since the set of all literals has the same cardinality as the set of variables, as it is well known from elementary set theory.

and let the full translation $\Theta : \mathcal{CLS} \to \mathcal{CLS}(\mathcal{VA}_{\{0,1\}})$ (which again is an injection) by given as

$$\boldsymbol{\Theta(F)} := \tau(F) \cup \{\mathrm{ALO}_v : v \in \mathrm{var}(F)\}.$$

Note that the union in the definition of $\Theta(F)$ is disjoint, since $\tau(F)$ consists only of positive clauses, while $\{\mathrm{ALO}_v : v \in \mathrm{var}(F)\}$ consists only of non-empty negative clauses (and thus $\Theta(F)$ is a "PN-clause-set" as defined in [19]). As an example, consider $F = \big\{ \{v \neq 0, w \neq 1\}, \{v \neq 1, w \neq 0\}, \{v \neq 2, w \neq 2\} \big\}$ for variables v, w with $D_v = D_w = \{0, 1, 2\}$. Now, using $a_\varepsilon := \tau((v, \varepsilon))$ and $b_\varepsilon := \tau((w, \varepsilon))$ for $\varepsilon \in \{0, 1, 2\}$ (so altogether we get six boolean variables here), we have

$$\Theta(F) = \big\{ \{a_0, b_1\}, \{a_1, b_0\}, \{a_2, b_2\}, \{\overline{a_0}, \overline{a_1}, \overline{a_2}\}, \{\overline{b_0}, \overline{b_1}, \overline{b_2}\} \big\}.$$

In general we have, that the sub-clause-sets of $\Theta(F)$ not containing pure variables (recall Subsection 3.8) are exactly the $\Theta(F')$ for $F' \subseteq F$ not containing pure variables.

5.2 Preservation of general structure

Regarding set-theoretical operations we have, that Θ is an embedding of the semilattice $(\mathcal{CLS}, \cup)$ into $(\mathcal{CLS}(\mathcal{VA}_{\{0,1\}}), \cup)$, that is, for $F_1, F_2 \in \mathcal{CLS}$ we have

$$\Theta(F_1 \cup F_2) = \Theta(F_1) \cup \Theta(F_2).$$

Thus Θ is also an order embedding, i.e., $F_1 \subseteq F_2 \Leftrightarrow \Theta(F_1) \subseteq \Theta(F_2)$. By definition we have for $F \in \mathcal{CLS}$ the equalities

$$\begin{aligned} c(\Theta(F)) &= c(F) + n(F) \\ n(\Theta(F)) &= \sum_{v \in \mathrm{var}(F)} |D_v| \\ \delta(\Theta(F)) &= c(\Theta(F)) - n(\Theta(F)) = c(F) - \mathrm{wn}(F) = \delta(F), \end{aligned}$$

and thus the translation Θ preserves the deficiency of clause–sets as defined in Subsection 4.1. It follows immediately, that $\delta^*(\Theta(F)) \geq \delta^*(F)$ holds for all $F \in \mathcal{CLS}$, but inequality can occur here (see Subsection 5.3).

We consider now the relations between partial assignments $\varphi \in \mathcal{PASS}$ for $F \in \mathcal{CLS}$ and partial assignments $\psi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ for $\Theta(F) \in \mathcal{CLS}(\mathcal{VA}_{\{0,1\}})$. For $\varphi \in \mathcal{PASS}$ we define the partial assignment $\boldsymbol{\tau(\varphi)} \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ by letting $\mathrm{var}(\tau(\varphi)) := \{\tau((v, \varepsilon)) : v \in \mathrm{var}(\varphi), \varepsilon \in D_v\}$ be the set of all boolean variables corresponding via the translation to literals over the variables in $\mathrm{var}(\varphi)$, while $\tau(\varphi)((v, \varepsilon)) = 0$ iff $\varphi(v) = \varepsilon$. If we consider for example the partial assignment $\langle v \to 1, w \to 2 \rangle$ for variables v, w with $D_v = D_w = \{0, 1, 2\}$, then, using as above $a_\varepsilon := \tau((v, \varepsilon))$ and $b_\varepsilon := \tau((w, \varepsilon))$ for $\varepsilon \in \{0, 1, 2\}$, we get

$$\tau(\langle v \to 1, w \to 2 \rangle) = \langle a_0 \to 1, a_1 \to 0, a_2 \to 1, b_0 \to 1, b_2 \to 1, b_2 \to 0 \rangle.$$

The partial assignments in $\mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ of the form $\tau(\varphi)$ for some $\varphi \in \mathcal{PASS}$ are called **standard partial assignments (w.r.t. τ)**. So τ constitutes a bijection between $\mathcal{PASS}$ and the standard partial assignments (which are always boolean), and standard partial assignments $\varphi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ are characterised by the condition, that whenever some $\tau((v, \varepsilon)) \in \mathrm{var}(\varphi)$, then for all $\varepsilon' \in D_v$ we have $\tau((v, \varepsilon')) \in \mathrm{var}(\varphi)$, and there is exactly one $\varepsilon_0 \in D_v$ with $\varphi(\tau((v, \varepsilon_0))) = 0$; for the corresponding partial assignment $\tau^{-1}(\varphi) \in \mathcal{PASS}$ we then have $\tau^{-1}(\varphi)(v) = \varepsilon_0$.

In the following lemma we see that the properties of φ regarding touching or satisfying clauses are well reflected by $\tau(\varphi)$, and hence the translation is invariant regarding the autarky property and the property of satisfying a clause-set.

Lemma 5.1 *For $\varphi \in \mathcal{PASS}$, $C \in \mathcal{CL}$ and $F \in \mathcal{CLS}$ we have*

1. *φ touches resp. satisfies C if and only if $\tau(\varphi)$ touches resp. satisfies $\tau(C)$. Thus*
$$\begin{aligned}\tau(F_{\mathrm{var}(\varphi)}) &= \tau(F)_{\mathrm{var}(\tau(\varphi))}\\ \Theta(F[\mathrm{var}(\varphi)]) &= \Theta(F)[\mathrm{var}(\tau(\varphi))].\end{aligned}$$
2. *$\tau(\varphi)$ is an autarky for the set of clauses $\{\mathrm{ALO}_v : v \in \mathcal{VA}\}$.*
3. *φ is an autarky for F if and only if $\tau(\varphi)$ is an autarky for $\Theta(F)$.*
4. *If $\tau(\varphi)$ satisfies $\Theta(F)$, then φ satisfies F. If on the other hand φ satisfies F and $\mathrm{var}(\varphi) \supseteq \mathrm{var}(F)$ holds, then $\tau(\varphi)$ satisfies $\Theta(F)$.*

Proof: Parts 1, 2 follow directly from the definitions, while Part 3 follows from Parts 1, 2, and Part 4 follows from Parts 1, 3. ▮

For the reverse direction, from partial assignments in $\mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ to partial assignments in $\mathcal{PASS}$, call $\varphi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ **admissible** if φ is an autarky for the set of clauses $\{\mathrm{ALO}_v : v \in \mathcal{VA}\}$, that is, if $\tau((v,\varepsilon)) \in \mathrm{var}(\varphi)$, then there is $\varepsilon_0 \in D_v$ with $\varphi(\tau((v,\varepsilon_0))) = 0$. In words: a partial assignment φ for the boolean variables is admissible iff in case it has some variable $\tau((v,\varepsilon))$ in its domain, then there exists a value $\varepsilon_0 \in D_v$ such that $\tau((v,\varepsilon_0))$ is in the domain of φ as well with $\varphi(\tau((v,\varepsilon_0))) = 0$. Note that an autarky $\varphi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ for $\Theta(F)$ (this includes satisfying assignments) with $\mathrm{var}(\varphi) \subseteq \mathrm{var}(\Theta(F))$ is always admissible.

Call a standard partial assignment $\psi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ a **standard completion** of an admissible $\varphi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ if ψ touches (satisfies) exactly the same ALO-clauses as φ, and if from $\psi(\tau((v,\varepsilon))) = 0$ always follows $\varphi(\tau((v,\varepsilon))) = 0$; in other words a standard completion ψ of an admissible φ is obtained from φ by considering all variables v such that $\varepsilon \in D_v$ with $\tau((v,\varepsilon)) \in \mathrm{var}(\varphi)$ exists, choosing $\varepsilon_0(v) \in D_v$ with $\varphi(\tau((v,\varepsilon_0(v)))) = 0$, and setting $\psi(\tau((v,\varepsilon'))) := 1$ for $\varepsilon' \in D_v \setminus \{\varepsilon_0(v)\}$, while $\psi(\tau((v,\varepsilon_0(v)))) := 0$.

The purpose of standard completions ψ of admissible partial assignments φ is, that from an autarky φ for $\Theta(F)$ we obtain an autarky ψ for $\Theta(F)$, where now ψ is a standard partial assignment, and so by Lemma 5.1 we obtain from ψ an autarky for F. The following lemma (with obvious proofs) states the basic properties of standard completions.

Lemma 5.2 *For $C \in \mathcal{CL}$ and $F \in \mathcal{CLS}$, an admissible $\varphi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ and a standard completion $\psi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ of φ we have*

1. *If φ touches resp. satisfies $\tau(C)$ then ψ touches resp. satisfies $\tau(C)$.*
2. *If φ is an autarky for $\Theta(F)$ then ψ is an autarky for $\Theta(F)$.*

Lemma 5.3 *For a (generalised) clause-set $F \in \mathcal{CLS}$ we have:*

1. *$F \in \mathcal{SAT} \Leftrightarrow \Theta(F) \in \mathcal{SAT}$.*

2. *$F \in \mathcal{MUSAT} \Leftrightarrow \Theta(F) \in \mathcal{MUSAT}$.*

3. *$F \in \mathcal{LEAN} \Leftrightarrow \Theta(F) \in \mathcal{LEAN}$.*

Proof: If $F \in \mathcal{SAT}$ then $\Theta(F) \in \mathcal{SAT}$ with Lemma 5.1, Part 4, and if $\Theta(F) \in \mathcal{SAT}$, then $F \in \mathcal{SAT}$ with Lemma 5.2, Part 1 and Lemma 5.1, Part 4.

If $F \in \mathcal{MUSAT}$, but there were some minimally unsatisfiable $F^* \subset \Theta(F)$, then there would be $F' \subset F$ with $\Theta(F') = F^*$ (since F^* does not contain pure variables), and thus F' would be unsatisfiable by Part 1. If on the other hand $\Theta(F) \in \mathcal{MUSAT}$, but there were some unsatisfiable $F' \subset F$, then $\Theta(F')$ would be unsatisfiable as well by Part 1.

Finally, if $F \in \mathcal{LEAN}$ then $\Theta(F) \in \mathcal{LEAN}$ by Lemma 5.2, Part 2 and Lemma 5.1, Part 3, and if $\Theta(F) \in \mathcal{LEAN}$ then $F \in \mathcal{LEAN}$ by Lemma 5.1, Part 3 (the other direction). ∎

Parts 1 and 3 have been concluded in Corollary 20 in [30] from the stronger property $\mathrm{N_a}(\Theta(F)) = \Theta(\mathrm{N_a}(F))$ (recall that $\mathrm{N_a}$ is the lean kernel operator); in this article we do not go further with the study of the translation Θ, but we restrict ourselves to the minimum required to understand our applications.

5.3 Preservation of matching structure

Lemma 5.4 *For $\varphi \in \mathcal{PASS}$, $C \in \mathcal{CL}$ and $F \in \mathcal{CLS}$ we have*

1. *If $\tau(\varphi)$ matching satisfies $\Theta(F)$, then φ matching satisfies F.*

2. *If $\tau(\varphi)$ is a matching autarky for $\Theta(F)$, then φ is a matching autarky for F.*

Proof: If the partial assignment $\tau(\varphi)$ matching satisfies $\Theta(F)$, then (by definition) for each clause $D \in \Theta(F)$ one can choose a literal $x_D \in D$ with $\varphi(x_D) = 1$, such that for the variables $\mathrm{var}(x_D) = \tau((v_D, \varepsilon_D))$ the map $D \in \Theta(F) \mapsto \tau((v_D, \varepsilon_D))$ is injective (whence $D \in \Theta(F) \mapsto (v_D, \varepsilon_D)$ is injective). Now the map $C \in F \mapsto v_{\tau(C)}$ has for each image $v_{\tau(C)}$ at most $|D_v| - 1$ inverse images, since for each $\varepsilon \in D_v$ there is at most one $D \in \Theta(F)$ with $v_D = v_{\tau(C)}$ and $\varepsilon_D = \varepsilon$, and exactly one of these D is the clause ALO_{v_D}.

For Part 2 recall that φ is a matching autarky for F iff φ matching satisfies $F[\mathrm{var}(\varphi)]$, which by Part 1 follows from $\tau(\varphi)$ matching satisfying $\Theta(F[\mathrm{var}(\varphi)])$, where by Lemma 5.1, Part 1 we have $\Theta(F[\mathrm{var}(\varphi)]) = \Theta(F)[\mathrm{var}(\tau(\varphi))]$, and thus the latter assertion is equivalent to $\tau(\varphi)$ being a matching autarky for $\Theta(F)$. ∎

Lemma 19, Part (1)(d) of [30] rephrased in the terminology of Subsection 4.5 says, that if φ is a *non-repetitive* matching autarky for F then $\tau(\varphi)$ is a matching autarky for $\Theta(F)$; in follows then in Corollary 20 of [30], that if $\Theta(F)$ is matching lean, then F is lean w.r.t. non-repetitive matching autarkies. These properties do not hold for matching autarkies in general (in the presence of non-boolean variables), as the following examples show.

An example, where a matching autarky φ for a (generalised) clause-set $F \in \mathcal{CLS}$ does not yield a matching autarky $\tau(\varphi)$ for $\Theta(F)$, is given for *multi*-clause-sets by the multi-clause-set $F_1 := 2 \cdot \{(v, 0)\}$ for a variable v with $D_v = \{0, 1, 2\}$: F_1 is matching satisfiable (but note that F_1 is lean w.r.t. non-repetitive matching autarkies), while $\mathrm{N_{ma}}(\Theta(F_1)) = \tau(F_1)$ (via matching autarkies we can only eliminate the ALO-clause), and thus $\Theta(F_1)$ is not matching satisfiable. One sees that the problem with transferring matching autarkies from generalised (multi-)clause-sets to their boolean translation lies in the possibility that a matching in the clause-variable graph $B(F)$ might use the same literal several times, which is not possible for the translated literals. To obtain an example using clause-*sets*, consider additionally two boolean variables w, w' and let

$$F_2 = \big\{ \{v \neq 0, w \neq 0\}, \{v \neq 0, w' \neq 0\}, \{w \neq 1\}, \{w' \neq 1\} \big\}.$$

The partial assignment $\varphi := \langle v \to 1, w \to 0, w' \to 0\rangle$ is matching satisfying for F_2 (note that again F_2 is lean w.r.t. non-repetitive matching autarkies), but $\tau(\varphi)$ is not a matching autarky for $\Theta(F_2)$, and moreover the matching lean kernel of $\Theta(F_2)$ is $\Theta(F_2) \setminus \{\mathrm{ALO}_v\}$ (again only the ALO-clause for v can be eliminated via matching autarkies), and thus $\Theta(F_2)$ is not matching satisfiable. Furthermore we have in this example $\delta^*(F_2) = 0$ and $\delta^*(\Theta(F_2)) = \delta(\Theta(F_2) \setminus \{\mathrm{ALO}_v\}) = \delta(\Theta(F_2)) - (1-2) = \delta(F_2) + 1 = 1$.

Now consider the transfer of matching autarkies in the other direction, that is, we have given a matching autarky $\varphi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ for $\Theta(F)$, and we want to obtain some associated matching autarky for F. The problem here is, that φ might use some variable $\tau((v,\varepsilon))$, but not a variable $\tau((v,\varepsilon'))$ for some $\varepsilon' \in D_v \setminus \{\varepsilon\}$, and such situations cannot be translated back to F. The simplest example for this phenomenon is (again) given by a *multi*-clause-set $F_3 := \{(v,1)\} + 2 \cdot \{(v,2)\}$ for a variable v with $D_v = \{0,1,2\}$: It is F_3 matching lean, but $\mathrm{N_{ma}}(\Theta(F_3)) = \tau(2 \cdot \{(v,2)\})$ (via the matching autarky $\langle \tau((v,0)) \to 0, \tau((v,1)) \to 1\rangle$ for $\Theta(F_3)$). A clause-*set* F_4, where F_4 is matching lean but $\Theta(F_4)$ is not is given by

$$F_4 := \big\{ \{v \neq 1\}, \{v \neq 2\}, \{v \neq 2, w \neq 0\}, \{w \neq 1\} \big\}$$

for an additional boolean variable w, since here

$$\mathrm{N_{ma}}(\Theta(F_4)) = \tau(\{\{v \neq 2\}, \{v \neq 2, w \neq 0\}, \{w \neq 1\}\}) \cup \{\mathrm{ALO}_w\}$$

via the matching autarky $\langle \tau((v,0)) \to 0, \tau((v,1)) \to 1\rangle$ for $\Theta(F_4)$.

As we have seen now, matching autarkies for (generalised) clause-sets $F \in \mathcal{CLS}$ and matching autarkies for $\Theta(F) \in \mathcal{CLS}(\mathcal{VA}_{\{0,1\}})$ in general are incomparable. Nevertheless we can use them to show fixed-parameter tractability for generalised clause-sets w.r.t. the parameter $\delta^*(F)$ as follows.

Theorem 5.5 *SAT decision for (generalised) clause-sets $F \in \mathcal{CLS}$ can be done in time $O\big(2^{\delta^*(F)} \cdot (\sum_{v \in \mathrm{var}(F)} |D_v|)^3\big)$*

Proof: Consider $F \in \mathcal{CLS}$ and let F^* be the result of reducing $\Theta(F)$ w.r.t. matching autarkies and pure autarkies (thus F^* is the unique maximal sub-clause-set of F which is matching lean and does not contain pure variables). We can compute F^* in polynomial time, and F^* is satisfiability equivalent to F. Since F^* contains no pure literals, it corresponds to a sub-clause-set of F, and thus we have $\delta(F^*) \leq \delta^*(F)$, and since F^* is matching lean we have $\delta^*(F^*) = \delta(F^*)$. Theorem 4 in [49] says, that satisfiability of F^* can be tested in time $O(2^{\delta^*(F^*)} \cdot n(F^*)^3)$, where in this procedure actually already the cost of reducing $\Theta(F)$ to F^* is included if we use $n(\Theta(F))$ instead of $n(F^*)$ in the big-Oh expression (see Section 5 in [49], or use the argumentation of Subsection 4.4 of this article), and the theorem follows. ∎

6 Irredundant and minimally unsatisfiable generalised clause-sets

One of the main motivation for the notion of "lean clause-sets" is, that in this way we get a "smooth" and flexible generalisation of the "rigid" but fundamental notion of minimally unsatisfiable clause-sets. In this section we will now consider some of the basic facts on minimally unsatisfiable clause-sets in our generalised setting. We start in Subsection 6.1 with a discussion of the notion of "irredundant clause-sets", a notion applicable also to satisfiable clause-sets, concentrating on the basic question of preservation of irredundancy under application of partial assignments. More detailed studies of irredundant clause-sets in the boolean case can be found in the following references:

1. [7] (speaking of "clause minimal formula") focuses on questions related to the problem (from a complexity theoretical perspective) when for given clause-sets F, H there exists a clause-set G such that $F \cup G$ is equivalent to H.
2. [43] considers in various forms (also mostly from a complexity-theoretical perspective) the problem of finding an irredundant core in a given clause-set.

In Subsection 6.2 we consider the in some sense most extreme case of irredundant clause-sets, namely "hitting clause-sets" (every two different clauses clash, that is, have no common falsifying assignment; in other words, the conflict graph is complete), and the natural generalisation to "multihitting clause-sets" (the conflict graph is multipartite). In Corollary 6.6 we show that hitting clause-sets are exactly those clause-sets which are irredundant after application of any partial assignment, and thus unsatisfiable hitting clause-sets are exactly those clause-sets which are minimally unsatisfiable after application of any partial assignment (Corollary 6.7). For unsatisfiable multihitting clause-sets in Lemma 6.8 it is shown that they have exactly one minimally unsatisfiable sub-clause-set (which can be computed efficiently by subsumption-elimination), and in Lemma 6.10 we show that the satisfiability problem for bihitting clause-sets (where the conflict graph is bipartite) is solvable in quasi-polynomial time (this problem is essentially the same problem as the hypergraph transversal problem).

In Subsection 6.3 "saturated minimally unsatisfiable clause-sets" are discussed; here we see a concrete example, where generalised clause-sets behave essentially more complicated than boolean clause-sets. In Subsection 6.4 we characterise minimally unsatisfiable generalised clause-sets of deficiency one as well as the special cases of saturated and marginal clause-sets, while finally in Subsection 6.5 we collect some observations which might serve for further progress in the characterisation of minimally unsatisfiable clause-sets.

6.1 Irredundant clause-sets

A clause $C \in F$ is called **redundant** (or **unnecessary**) for clause-set $F \in \mathcal{CLS}$ if $F \setminus \{C\} \models C$ holds, while otherwise C is called **irredundant** (or **necessary**) for F. The following conditions are equivalent for a clause $C \in F$:

- C is redundant for F.
- $F \setminus \{C\}$ is equivalent to F.

- The set $\overline{\mathrm{mod}}_{\mathrm{var}(F)}(C)$ of falsifying assignments for C is covered by the family $(\overline{\mathrm{mod}}_{\mathrm{var}(F)}(C'))_{C' \in F \setminus \{C\}}$ of sets of falsifying assignments for the remaining clauses.

A (generalised) clause-set $F \in \mathcal{CLS}$ is called **irredundant** if all $C \in F$ are irredundant for F, otherwise F is called **redundant**. A clause-set F is minimally unsatisfiable if and only if F is unsatisfiable and irredundant. Regarding complexity classifications of decision problems related to (ir)redundancy we have the following:

1. In [46] it is shown that the decision problem whether a (boolean) clause-set is irredundant is NP-complete, while the decision problem whether a (boolean) clause-set is minimally unsatisfiable is D^P-complete. Trivially these results also hold for generalised clause-sets.
2. As we have seen in Theorem 5.5, SAT decision for (generalised) clause-sets is fixed-parameter tractable in the maximal deficiency, and thus also irredundancy decision is fixed-parameter tractable in the maximal deficiency. Since for minimally unsatisfiable (generalised) clause-sets maximal deficiency and deficiency coincide (Corollary 4.22), minimally unsatisfiability decision is also fixed-parameter tractable in the deficiency; however, as shown in Proposition 1 in [7], the decision whether a (boolean) clause-set is irredundant with deficiency k is NP-complete for every fixed $k \in \mathbb{N}$ (different from minimally unsatisfiable clause-sets, irredundant clause-sets of deficiency k can contain sub-clause-sets of arbitrary deficiency). Obviously the same holds for generalised clause-sets.

We are interested here in the question, given a partial assignment φ and a clause $C \in F$ with $\varphi * \{C\} \neq \top$ (i.e., C is not satisfied by φ), under what circumstances is the clause $\varphi * C = C \setminus C_\varphi$ redundant for $\varphi * F$? We will see, that this question is closely related to the question, how "much irredundant" C is for F, that is, how much of $\overline{\mathrm{mod}}_{\mathrm{var}(F)}(C)$ is covered by $(\overline{\mathrm{mod}}_{\mathrm{var}(F)}(C'))_{C' \in F \setminus \{C\}}$, which can be recast as the question, whether for some $C' \supseteq C$ we have $F \setminus \{C\} \models C'$.

Assume that $\varphi * C$ is redundant for $\varphi * F$, that is, $(\varphi * F) \setminus (\varphi * \{C\}) \models \varphi * C$ holds. Due to $(\varphi * F) \setminus (\varphi * \{C\}) \subseteq \varphi * (F \setminus \{C\})$ it follows $\varphi * (F \setminus \{C\}) \models \varphi * C$, which is equivalent to $F \setminus \{C\} \models C \cup C_\varphi$. Let us call C **φ-redundant** for F if $F \setminus \{C\} \models C \cup C_\varphi$ holds, and otherwise **φ-irredundant**. In other words, C is φ-redundant for F iff the part of $\overline{\mathrm{mod}}_{\mathrm{var}(F)}(C)$ which consists of assignments compatible with φ is covered by $(\overline{\mathrm{mod}}_{\mathrm{var}(F)}(C'))_{C' \in F \setminus \{C\}}$. Obviously, C is redundant for F iff C is $\emptyset$-redundant for F, and if C is φ-redundant for F, then C is also φ'-redundant for F for every partial assignment φ' with $\varphi \subseteq \varphi'$. For an example consider boolean variables a, b and the irredundant clause-set $F = \{\{a\}, \{b\}\}$: $\{b\}$ is $\langle a \rightarrow 0 \rangle$-redundant for F.

If C is φ-irredundant for F, then $\varphi * C$ is irredundant for $\varphi * F$, but the reverse direction is not true in general due to the fact, that there might be other clauses $C' \in F$ with $\varphi * C' = \varphi * C$. To repair this, let us call clause C **contraction-φ-redundant** for F if

$$F \setminus \{C' \in F : \varphi * \{C'\} = \varphi * \{C\}\} \models C \cup C_\varphi,$$

while otherwise we call C **contraction-φ-irredundant** for F. We summarise (and extend) the foregoing discussion in Lemma 6.1, whose proof should be obvious by now.

Lemma 6.1 *Consider a generalised clause-set $F \in \mathcal{CLS}$, a clause $C \in F$ and a partial assignment $\varphi \in \mathcal{PASS}$ such that $\varphi * \{C\} \neq \top$.*

1. *$\varphi * C$ is (ir)redundant for $\varphi * F$ if and only if C is contraction-φ-(ir)redundant for F.*
2. (a) *If C is φ-irredundant for F, then C is contraction-φ-irredundant for F.*
   (b) *If there is no clause $C' \in F \setminus \{C\}$ with $\varphi * \{C'\} = \varphi * \{C\}$ (that is, C is "contraction-free" in F w.r.t. φ), then also the reverse direction holds, that is, if C is contraction-φ-irredundant for F then C is φ-irredundant for F. It is C contraction-free in F w.r.t. φ in the following cases:*
      (i) *$n(\varphi) = 0$ (i.e., φ is the empty partial assignment);*
      (ii) *$n(\varphi) = 1$ and F is subsumption-free;*
      (iii) *C clashes with every $C' \in F \setminus \{C\}$.*

Corollary 6.2 *Consider a generalised clause-set $F \in \mathcal{CLS}$ which is subsumption-free, a clause $C \in F$ and a variable $v \in \mathcal{VA}$ together with $\varepsilon \in D_v$ such that for all $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we have $(v, \varepsilon') \notin C$. Then $\langle v \to \varepsilon \rangle * C = C \setminus \{(v, \varepsilon)\}$ is irredundant for $\langle v \to \varepsilon \rangle * F$ if and only if C is $\langle v \to \varepsilon \rangle$-irredundant for F, that is, iff $F \setminus \{C\} \not\models C \cup \{(v, \varepsilon)\}$.*

Obviously irredundant clause-sets are subsumption-free, and from Corollary 6.2 we get immediately:

Corollary 6.3 *Consider an irredundant generalised clause-set $F \in \mathcal{CLS}$, a clause $C \in F$ and a variable $v \in \mathcal{VA}$ together with $\varepsilon \in D_v$.*

1. *If there exists $\varepsilon' \in D_v \setminus \{\varepsilon\}$ with $(v, \varepsilon') \in C$, then clause C vanishes when applying $\langle v \to \varepsilon \rangle$ to F (and in that sense it becomes redundant in $\langle v \to \varepsilon \rangle$). So assume $\mathrm{val}_v(\{C\}) \subseteq \{\varepsilon\}$ in the sequel.*
2. *If $(v, \varepsilon) \in C$, then $\langle v \to \varepsilon \rangle * C = C \setminus \{(v, \varepsilon)\}$ is irredundant for $\langle v \to \varepsilon \rangle * F$.*
3. *If $(v, \varepsilon) \notin C$, then C is irredundant for $\langle v \to \varepsilon \rangle * F$ if and only if C is $\langle v \to \varepsilon \rangle$-irredundant for F, i.e., iff $F \setminus \{C\} \not\models C \cup \{(v, \varepsilon)\}$.*

Considering a clause $C \in F$, we called C redundant for F iff $F \setminus \{C\} \models C$; now for arbitrary clauses C we can call C "dependent" on F if $F \models C$ holds (that is, if the set of falsifying assignments of F covers the set of falsifying assignments of C), and otherwise "independent". If $C \in F$, then C is dependent on F, while C is redundant for F iff C is dependent on $F \setminus \{C\}$. The relation of C depending on F allows two dimensions for minimisation: Considering a minimal clause C which is dependent on F we arrive at the notion of a *prime implicant* of F, while considering a minimal clause-set F such that C depends on F we arrive at a "minimal premise set" for C. The following lemma states the relation between minimal premise sets and minimally unsatisfiable clause-sets.

Lemma 6.4 *Consider a generalised clause-set $F \in \mathcal{CLS}$ and a clause $C \in \mathcal{CL}$. Then the following assertions are equivalent:*

1. *F is a minimal premise set for C.*
2. *$\varphi_C * F$ is minimally unsatisfiable, no clause of F is satisfied by φ, and F is φ-contraction free, that is, there are no clauses $C, C' \in F$, $C \neq C'$, with $\varphi * \{C\} = \varphi * \{C'\}$.*

6.2 Hitting and multihitting clause-sets

The next lemma answers the question which clauses C remain irredundant for a clause-set F under *all* applications of partial assignments; this strongest form of irredundancy of C for F turns out to be equivalent to the condition, that the set of falsifying assignments for C is not covered at all by $(\overline{\mathrm{mod}}_{\mathrm{var}(F)}(C'))_{C' \in F \setminus \{C\}}$. A simple but important observation here is, that for two clauses C, C' and $\mathrm{var}(C) \cup \mathrm{var}(C') \subseteq V$ we have $\overline{\mathrm{mod}}_V(C) \cap \overline{\mathrm{mod}}_V(C') = \emptyset$ iff C and C' clash.

Lemma 6.5 *Consider a generalised clause-set $F \in \mathcal{CLS}$ and a clause $C \in \mathcal{CL}$. Then the following assertions are equivalent:*

(i) *C is φ-irredundant for all $\varphi \in \mathcal{PASS}$.*

(ii) *C is contraction-φ-irredundant for all $\varphi \in \mathcal{PASS}$.*

(iii) $\overline{\mathrm{mod}}_{\mathrm{var}(F)}(C) \cap \bigcup_{C' \in F \setminus \{C\}} \overline{\mathrm{mod}}_{\mathrm{var}(F)}(C') = \emptyset$.

(iv) *C clashes with every $C' \in F \setminus \{C\}$, i.e., clause C is connected in the conflict graph $\mathrm{cg}(F)$ to every other vertex.*

Proof: By the above remark we see that (iii) and (iv) are equivalent. By definition (iii) is equivalent to (i), while by Lemma 6.1, part 2 it is (i) equivalent to (ii). ∎

Corollary 6.6 *A generalised clause-set $F \in \mathcal{CLS}$ is a hitting clause-set if and only if for all $\varphi \in \mathcal{PASS}$ it is $\varphi * F$ irredundant.*

Generalising Theorem 32 in [33]:

Corollary 6.7 *A generalised clause-set $F \in \mathcal{CLS}$ is an unsatisfiable hitting clause-set if and only if $\varphi * F$ is minimally unsatisfiable for every $\varphi \in \mathcal{PASS}$.*

Hitting clause-sets are irredundant; the more general class of *multihitting clause-sets* (clause-sets with complete multipartite conflict graph) contains redundant clause-sets, but all redundancies can be removed efficiently (and canonically), as the following lemma shows. We use the notion of an **irredundant core** of a clause-set $F \in \mathcal{CLS}$ which is an irredundant $F' \subseteq F$ such that F' is equivalent to F (in [43] the notion "irredundant equivalent subset" is used). An irredundant core of an unsatisfiable clause-set is called a **minimally unsatisfiable core**.

Lemma 6.8 *Consider a generalised clause-set $F \in \mathcal{CLS}$ without trivial variables which is multihitting. Let $\mathbb{F}$ be the multipartition of F, and $V := \mathrm{var}(F)$.*

1. *For $F_1, F_2 \in \mathbb{F}$, $F_1 \neq F_2$ we have $\overline{\mathrm{mod}}_V(F_1) \cap \overline{\mathrm{mod}}_V(F_2) = \emptyset$.*
2. *If for $F' \subseteq F$ and $C \in F \setminus F'$ we have $F' \models C$, then there must be some $C' \in F'$ with $C' \subset C$.*
3. *F has exactly one irredundant core, which is obtained from F by subsumption-elimination. Thus if F is unsatisfiable, then F has exactly one minimally unsatisfiable core, which is obtained from F by subsumption-elimination.*
4. *A hitting clause-set F is unsatisfiable iff $\sum_{C \in F} |\overline{\mathrm{mod}}_V(\{C\})| = |\mathcal{PASS}(V)|$.*

Proof: Part 1 follows by definition. In Part 2 it is $\overline{\mathrm{mod}}_V(\{C\})$ covered by $\overline{\mathrm{mod}}_V(F')$, and thus by Part 1 in fact $\overline{\mathrm{mod}}_V(\{C\})$ is covered by $\overline{\mathrm{mod}}_V(F' \cap F_C)$, where $F_C \in \mathbb{F}$ with $C \in F_C$; i.e., $F_C \cap F' \models \{C\}$. By the strong completeness of resolution and the fact that within F_C no clashes exist, it follows that there must be $C' \in F' \cap F_C$ with $C' \subset C$. Part 3 follows immediately from Part 2. Finally Part 4 follows immediately from Part 1. ∎

Corollary 6.9 *A multihitting clause-set is irredundant if and only if F is subsumption-free. Thus an unsatisfiable multihitting clause-set is minimally unsatisfiable if and only if F is subsumption-free.*

By Corollary 6.9 we know that deciding whether a multihitting clause-set is minimally unsatisfiable is the same task (up to subsumption elimination) as deciding whether it is unsatisfiable. Obviously $\mathcal{MHIT} \cap \mathcal{USAT}$ is in co-NP (and thus also $\mathcal{MHIT} \cap \mathcal{MUSAT}$). We have more precise information only for special cases.

Using $|\overline{\mathrm{mod}}_{\mathrm{var}(F)}(C)| = \prod_{v \in \mathrm{var}(F) \setminus \mathrm{var}(C)} |D_v|$ for $C \in F$ it follows that satisfiability for hitting clause-sets is decidable in polynomial time (generalising the well-known special case for boolean clause-sets). For boolean *bihitting* clause-sets in [19] it was shown, that satisfiability decision can be done in quasi-polynomial time (where "quasi-polynomial" means a "polynomial" upper bound with the exponent of logarithmic order in the size of the input), since satisfiability decision for bihitting clause-sets is essentially the same as deciding whether for two given hypergraphs one is the transversal hypergraph of the other; this can immediately be generalised:

Lemma 6.10 *Satisfiability for generalised clause-sets which are bihitting is decidable in quasi-polynomial time.*

Proof: Variables with a domain size greater than two appearing in a bihitting clause-set must be pure variables, since if a generalised clause-set contains a variable of domain size k, then the conflict graph contains the complete graph K_k (which is not bipartite). ∎

It seems to be a very interesting question, to what degree (generalised) multihitting clause-sets have efficient satisfiability decision (see Subsection 7.5 for further discussion, and see [34] for more information in the boolean case).

We conclude this section by some general results on irredundant cores, generalising [40] (where only unsatisfiable boolean clause-sets have been considered). Recall that a (finite) hypergraph is a pair $(V, \mathbb{E})$ such that V is a (finite) set (the "vertex set") and $\mathbb{E}$ is a set of subsets of V (the set of "hyperedges"). For a hypergraph G by $\min(G)$ resp. $\max(G)$ we denote the hypergraph with the same vertex set and with all inclusion-minimal resp. maximal hyperedges. Consider a (generalised) clause-set F. Let $\mathrm{EQ}(F)$ be the hypergraph with vertex set F (the clauses of F), while the hyperedges are all subsets of F which are equivalent to F, and let $\mathrm{NEQ}(F)$ be the hypergraph with vertex set F and hyperedges the subsets of F which are not equivalent to F. If F is unsatisfiable, then $\mathrm{EQ}(F) = \mathcal{USAT}(F)$, the hypergraph consisting of all unsatisfiable sub-cause-sets of F, while $\mathrm{NEQ}(F) = \mathcal{SAT}(F)$, the hypergraph of all satisfiable sub-clause-sets of F. Now $\min(\mathrm{EQ}(F))$ is the hypergraph consisting of all irredundant cores of F; if F is unsatisfiable then $\min(\mathrm{EQ}(F)) = \mathrm{MU}(F)$, the hypergraph of all minimally unsatisfiable cores of F.

Generalising [43, 40], the elements of $\bigcap \mathrm{EQ}(F) = \bigcap \min(\mathrm{EQ}(F))$, the clauses which are in every irredundant core of F, are called *necessary clauses*, while, following [40], the elements of $\bigcup \min(\mathrm{EQ}(F))$, the clauses which are in some irredundant core, are called *potentially necessary clauses* (in [43] such clauses are called "useful"). We see that necessary clauses are exactly the irredundant clauses as defined before. Regarding decision complexity we have:

1. A clause-set F is satisfiable iff $\bot$ is necessary for $F \cup \{\bot\}$, and thus already for (boolean) unsatisfiable clause-sets decision whether a clause is necessary is NP-complete (this was noticed for (arbitrary) boolean clause-sets in Theorem 3 in [43], and trivially also the decision problem whether some clause is necessary for an generalised clause-sets is NP-complete as well).
2. By Theorem 4 in [43] we have, that decision whether a clause C is potentially necessary for a (boolean) clause-set F is Σ_2^{P}-complete (where Σ_2^{P} is the class of problems reducible to the decision problem whether a quantified boolean formula with quantifier-prefix $\exists^*\forall^*$ is true). Trivially this holds also for all generalised clause-sets, and due to $\bigcup \min(\mathrm{EQ}(F \cup \{\bot\})) = \{\bot\} \cup \bigcup \min(\mathrm{EQ}(F))$ we can restrict F here again to unsatisfiable clause-sets.
3. In Theorem 5 in [43] it is shown that decision whether a (boolean) clause-set has a unique irredundant core is $\Delta_2^{\mathrm{P}}[\log n]$-complete (where Δ_2^{P} is the class of problems decidable in polynomial time by arbitrary use of an NP-oracle, while for $\Delta_2^{\mathrm{P}}[\log n]$ only logarithmically many oracle calls are allowed). Obviously this carries over to generalised clause-sets, however whether again restriction to unsatisfiable clause-sets is possible (that is, deciding whether an unsatisfiable clause-set has a unique minimally unsatisfiable core) is not clear.

Finally we can also generalise the observation of Bailey and Stuckey, using the same proof as in [40] (Section 2): For a hypergraph G denote by $\mathrm{Tr}(G)$ the hypergraph with the same vertex set $V(G)$, while the hyperedges are the minimal transversals of G (minimal subsets of $V(G)$ intersecting every hyperedge), and denote by $\complement(G)$ the hypergraph with vertex set $V(G)$ and hyperedges $V(G) \setminus H$ for $H \in E(G)$.

Lemma 6.11 *For every (generalised) clause-set F we have*

$$\min(\mathrm{EQ}(F)) = \mathrm{Tr}(\complement(\max(\mathrm{NEQ}(F)))).$$

Proof: The assertion is equivalent to $\complement(\mathrm{Tr}(\min(\mathrm{EQ}(F)))) = \max(\mathrm{NEQ}(F))$, which just states that the maximal non-equivalent sub-clause-sets of F are exactly the maximal independent vertex sets of $\min(\mathrm{EQ}(F))$, i.e., those maximal sets of clauses not containing an irredundant core. ∎

6.3 Saturated minimally unsatisfiable clause-sets

A clause-set $F \in \mathcal{CLS}$ is called **saturated minimally unsatisfiable**, if F is unsatisfiable, but for any clause $C \in F$ replacing C in F by $C \cup \{x\}$ for any literal x with $\mathrm{var}(x) \notin \mathrm{var}(C)$ and $|D_{\mathrm{var}(x)}| \geq 2$ yields a satisfiable clause-set.[18] Saturated minimally unsatisfiable clause-sets are minimally unsatisfiable (consider x such that $\mathrm{var}(x) \notin \mathrm{var}(F)$), and actually a clause-set F is saturated minimally unsatisfiable

[18] Instead of "saturated" in [1] "strong" is used, and in [8] "maximal"; we follow [16].

iff it is minimally unsatisfiable and addition of a literal x with $\mathrm{var}(x) \in \mathrm{var}(F)$ to any clause C with $\mathrm{var}(x) \notin \mathrm{var}(C)$ yields a satisfiable clause-sets. The set of all saturated minimally unsatisfiable clause-sets is called $\boldsymbol{\mathcal{SMUSAT}}$. By Lemma 6.8, part 4 we see that unsatisfiable hitting clause-sets are in $\mathcal{SMUSAT}$.

Lemma 6.12 *Every minimally unsatisfiable clause-set $F \in \mathcal{MUSAT}$ can be **saturated**, that is there exists $F^* \in \mathcal{SMUSAT}$ with $\mathrm{var}(F^*) = \mathrm{var}(F)$ and a bijection $\pi : F \rightarrow F^*$ such that for all $C \in F$ we have $C \subseteq \pi(C)$.*

Proof: The observation needed here is, that if for a minimally unsatisfiable clause-set F we replace some clause $C \in F$ by a clause $C' \supset C$, obtaining $F' := (F \setminus \{C\}) \cup \{C'\}$, then F' is minimally unsatisfiable if F' is unsatisfiable (the only possibly redundant clause in F' is C', and if C' is redundant in F', then F' is satisfiable, since $F' \setminus \{C'\} = F \setminus \{C\} \in \mathcal{SAT}$). So we can add literals x with $\mathrm{var}(x) \in \mathrm{var}(F)$ to clauses such that we maintain (minimally) unsatisfiability, and finally we will end up with a saturated F^*. ∎

For boolean clause-sets the characterisation of $\mathcal{SMUSAT}$ from Lemma C.1 in [26] is fundamental: A minimally unsatisfiable boolean clause-set F is saturated if and only if for every variable $v \in \mathrm{var}(F)$ and each $\varepsilon \in D_v = \{0,1\}$ it is $\langle v \rightarrow \varepsilon \rangle * F$ minimally unsatisfiable. Together with saturation this characterisation provides a powerful method for proving properties of minimally unsatisfiable clause-sets via induction on the number of variables. For generalised clause-sets saturatedness is weaker, and the above condition is only sufficient for being minimally unsatisfiable, but is no longer necessary. The following lemma develops these fundamental facts, using the following notion: We say that *addition of literal x renders clause-set F satisfiable* iff for all clauses $C \in F$ with $\mathrm{var}(x) \notin \mathrm{var}(C)$ the clause-set $(F \setminus \{C\}) \cup \{C \cup \{x\}\}$ is satisfiable (thus a clause-set F is saturated minimally unsatisfiable iff F is unsatisfiable and addition of any literal renders F satisfiable).

Lemma 6.13 *Consider a generalised clause-set $F \in \mathcal{MUSAT}$ and a literal $(v, \varepsilon) \in \mathcal{LIT}$.*

1. *If $\langle v \rightarrow \varepsilon \rangle * F \in \mathcal{MUSAT}$, then for all $\varepsilon' \in D_v \setminus \{\varepsilon\}$ addition of literal (v, ε') renders F satisfiable.*
2. *If v is boolean, and for $\varepsilon' \in D_v \setminus \{\varepsilon\}$ addition of literal (v, ε') renders F satisfiable, then $\langle v \rightarrow \varepsilon \rangle * F \in \mathcal{MUSAT}$.*

Proof: For Part 1 assume that there is $C \in F$, $v \notin \mathrm{var}(C)$ and $\varepsilon' \in D_v \setminus \{\varepsilon\}$ such that $F' := (F \setminus \{C\}) \cup \{C \cup \{(v, \varepsilon')\}\}$ is unsatisfiable. Then $\langle v \rightarrow \varepsilon \rangle * F' \in \mathcal{USAT}$ with $\langle v \rightarrow \varepsilon \rangle * F' = (\langle v \rightarrow \varepsilon \rangle * F) \setminus \{C\}$, and thus C would be redundant in $\langle v \rightarrow \varepsilon \rangle * F$.

For Part 2 assume that $\langle v \rightarrow \varepsilon \rangle * F$ is not minimally unsatisfiable; by Corollary 6.3 thus there is a clause $C \in F$, $v \notin \mathrm{var}(C)$ such that $F \setminus \{C\} \models C \cup \{(v, \varepsilon)\}$. It follows that for $F' := (F \setminus \{C\}) \cup \{C \cup \{(v, \varepsilon')\}\}$ we have $F' \models C$ (using one resolution step), and thus F' would be unsatisfiable. ∎

Corollary 6.14 *If for the generalised clause-set $F \in \mathcal{CLS}$ for every partial assignment $\varphi \in \mathcal{PASS}$ with $n(\varphi) \leq 1$ we have $\varphi * F \in \mathcal{MUSAT}$, then $F \in \mathcal{SMUSAT}$. If F is boolean, then also the reverse direction holds, that is, $F \in \mathcal{SMUSAT}$ if and only if for every partial assignment $\varphi \in \mathcal{PASS}$ with $n(\varphi) \leq 1$ we have $\varphi * F \in \mathcal{MUSAT}$.*

An example to show that the implication "$F \in \mathcal{SMUSAT} \Rightarrow \langle v \to \varepsilon \rangle * F \in \mathcal{MUSAT}$" in Corollary 6.14 does not hold for generalised clause-sets is as follows: Consider variables a, b with $D_a = D_b = \{0, 1, 2\}$, and let F be the following clause-set with 4 binary clauses and 2 unary clauses:

$$F := \big\{ \{a \neq 0, b \neq 0\}, \{a \neq 1, b \neq 0\}, \{a \neq 0, b \neq 1\}, \{a \neq 1, b \neq 1\}, \\ \{a \neq 2\}, \{b \neq 2\} \big\}.$$

It is $F \in \mathcal{SMUSAT}$ (after unit-clause elimination we obtain a boolean clause-set with all possible (full) clauses), while $\langle a \to 2 \rangle * F = \{\bot, \{b \neq 2\}\} \notin \mathcal{MUSAT}$ (as well as $\langle b \to 2 \rangle * F = \{\bot, \{a \neq 2\}\} \notin \mathcal{MUSAT}$). It might be worth investigating the class of (generalised) clause-sets F such that for all partial assignments φ with $n(\varphi) \leq 1$ we have $\varphi * F \in \mathcal{MUSAT}$ (a strict subset of $\mathcal{SMUSAT}$).

An important application of the process of saturation for *boolean* clause-sets is given by Lemma C.2 in [26], proving that for every $F \in \mathcal{MUSAT}$, $F \neq \{\bot\}$ there is a variable $v \in \mathrm{var}(F)$ such that for $\varepsilon \in D_v = \{0, 1\}$ we have $\#_{(v,\varepsilon)}(F) \leq \delta(F)$, that is, F has a "min-max var-degree" of at most $\delta(F)$. The proof is based on the characterisation of saturated minimally unsatisfiable boolean clause-sets in Corollary 6.14 and uses $\delta(F) \geq 1$ for $F' \in \mathcal{MUSAT}$, where F' is obtained from F by applying suitable partial assignments φ with $n(\varphi) = 1$. It is not completely clear how to obtain a full generalisation for generalised clause-sets (the problem is that saturation is not that powerful anymore); in Lemma 6.15 we obtain the generalisation of the min-max var-degree bound to generalised clause-sets in the special case of deficiency one, while in Subsection 6.5 we consider the class of minimally unsatisfiable clause-sets stable under application of partial assignments with at most one variable, for which then the full min-max var-degree bound can be shown (in Lemma 6.19).

6.4 Characterisation of the basic case of deficiency one

Generalising the tree construction from [26] (exploiting a formula class introduced by Stephen Cook and communicated to me by Alasdair Urquhart), let a **deficiency-1 tree representation** (in the remainder of this section just called "tree representation") be a 4-tuple (T, r, v, ε), where

- (T, r) is a finite tree with root r (inner nodes (that is, nodes which are not leaves) can have an arbitrary number of children).
- v labels each inner node w of (T, r) with a variable $v(w)$.
- ε labels each edge e leading from a node w to a node w' (edges are directed from the root towards the leaves) with a value $\varepsilon(e) \in D_{v(w)}$ such that the labelling of the edges going out from w yields a bijection to $D_{v(w)}$.

If an order on the value set $D_{v(w)}$ is given, then also the outgoing edges are ordered by the same order; in the special case of boolean variables thus we can speak of "left" and "right" branches, corresponding to the positive and negative literal. An example R is given as follows.

$$\begin{tikzcd}
 & & & & a \arrow[dlll, "0"'] \arrow[d, "1"] \arrow[drrr, "2"] & & & & \\
 & b \arrow[dl, "0"'] \arrow[dr, "1"] & & & c \arrow[dl, "0"'] \arrow[d, "1"] \arrow[dr, "2"] & & & d \arrow[d, "0"] & \\
 {} & & e \arrow[d, "0"] & {} & {} & {} & & f \arrow[dl, "0"'] \arrow[dr, "1"] & \\
 & & {} & & & & {} & & {}
\end{tikzcd}$$

This tree representation R uses six variables $a, \dots, f$ with $D_a = D_c = \{0, 1, 2\}$, $D_b = D_f = \{0, 1\}$ (thus b, f are boolean variables), and $D_d = D_e = \{0\}$.

Given a tree representation (T, r, v, ε), to every node w of (T, r) we associate a clause C_w by considering the path $w_0, e_1, w_1, \dots, e_m, w_m$ from the root to w in T (thus $w_0 = r$, $w_m = w$, and the e_i are the connecting edges from w_{i-1} to w_i, while m is the length of the path), and setting $C_w := \{(v(w_i), \varepsilon(e_{i+1})) : i \in \{0, \dots, m-1\}\}$. The clause-set $\boldsymbol{F(T, r, v, \varepsilon)}$ is defined as the set of all clauses C_w for leaves w of (T, r). For the above example R we get

$$\begin{aligned} F(R) = \big\{ & \{a \neq 0, b \neq 0\},\ \{a \neq 0, b \neq 1, e \neq 0\}, \\ & \qquad \{a \neq 1, c \neq 0\},\ \{a \neq 1, c \neq 1\},\ \{a \neq 1 c \neq 2\}, \\ & \qquad\qquad \{a \neq 2, d \neq 0, f \neq 0\},\ \{a \neq 2, d \neq 0, f \neq 1\} \big\}. \end{aligned}$$

We list some basic properties of the clause-sets $F(T, r, v, \varepsilon)$:

1. The rooted tree (T, r) yields a resolution tree for $F(T, r, v, \varepsilon)$ by labelling the nodes w with clauses C_w and considering the variables $v(w)$ for inner nodes w as resolution variables; since $C_r = \perp$ we see that $F(T, r, v, \varepsilon)$ is unsatisfiable.

2. $F(T, r, v, \varepsilon)$ is a 1-regular hitting clause-set (for two different clauses C_{w_1}, C_{w_2} the unique clashing variable is $v(w_0)$ for the root w_0 of the smallest sub-tree of (T, r) containing w_1 and w_2). It follows that $F(T, r, v, \varepsilon)$ is saturated minimally unsatisfiable.

3. $\delta(F(T, r, v, \varepsilon)) = 1$, since $c(F(T, r, v, \varepsilon))$ is the number of leaves of (T, r), while $\mathrm{wn}(F(T, r, v, \varepsilon))$ is the number of edges of T minus the number of inner nodes of (T, r), and thus $\delta(F(T, r, v, \varepsilon))$ is the difference of the number of vertices and the number of edges of T, which is 1 for every tree.

4. If $n(F(T, r, v, \varepsilon)) > 0$ (that is, if (T, r) is not trivial), then we have:

   (a) There is exactly one variable occurring in every clause of $F(T, r, v, \varepsilon)$ (namely $v(r)$).

   (b) Every clause $C \in F(T, r, v, \varepsilon)$ contains a literal $x \in C$ with $\#_x(F) = 1$ (namely with $\mathrm{var}(x) = v(w_0)$, where $C = C_w$ and w_0 is the parent node of w).

   (c) There exists a variable $v \in \mathrm{var}(F(T, r, v, \varepsilon))$ such that for all $\varepsilon \in D_v$ we have $\#_{(v,\varepsilon)}(F(T, r, v, \varepsilon)) = 1$ (choose $v = v(w)$ for an inner node w of (T, r) such that all children of w are leaves).

We can read off many more properties of $F(T, r, v, \varepsilon)$ directly from the tree representation, for example the minimal resp. maximal clause-length is the minimal resp. maximal depth of a leaf, but we need here only the above listed properties. Using $\mathcal{RHIT}$ for the set of regular hitting clause-sets and hd for the hitting degree, as introduced before, we have $F(T, r, v, \varepsilon) \in \mathcal{RHIT}^{\mathrm{sat}=0}_{\mathrm{hd}=1,\delta=1}$.

We say that a clause-set $F' \in \mathcal{CLS}$ is obtained from $F(T, r, v, \varepsilon)$ by **literal elimination** if F' is obtained from $F(T, r, v, \varepsilon)$ by eliminating some literal occurrences (at least one) without ever creating a pure variable. Replacing "$F(T, r, v, \varepsilon)$" by F', Properties 1, 3, 4b, 4c are still valid, while Properties 2, 4a are lost: F' is definitely not a hitting clause-set anymore, and there does not need to exist a variable occurring in every clause. It is furthermore F' definitely not saturated anymore (by the definition of F'), however F' is still minimally unsatisfiable (since removal of any clause either creates a pure variable or removes the only clause).

In Lemma C.5 from [26] it is shown, that the boolean elements of $\mathcal{SMUSAT}_{\delta=1}$ are exactly the clause-sets $F(T, r, v, \varepsilon)$ using only boolean variables, while the elements of $\mathcal{MUSAT}_{\delta=1} \setminus \mathcal{SMUSAT}_{\delta=1}$ are exactly the clause-sets obtained from such $F(T, r, v, \varepsilon)$ by literal elimination. To generalise this characterisation, the following lemma is central (compare Property 4c from above).

Lemma 6.15 *For every (generalised) clause-set $F \in \mathcal{MUSAT}_{\delta=1}$ with $n(F) > 0$ there exists a variable $v \in \mathrm{var}(F)$ such that for all $\varepsilon \in D_v$ we have $\#_{(v,\varepsilon)}(F) = 1$.*

Proof: Consider $F \in \mathcal{MUSAT}_{\delta=1}$. We investigate the structure of $\Theta(F)$ (recall Section 5). As we remarked in Subsection 5.2, we have $\delta(\Theta(F)) = 1$, and thus by Lemma 5.3 we have $\Theta(F) \in \mathcal{MUSAT}_{\delta=1}$. Since $\Theta(F)$ is a boolean clause-set, we can conclude that $\Theta(F)$ is obtained by literal elimination from some tree representation (T, r, v, ε) as defined above (using only boolean variables).

$\Theta(F)$ always has the following special properties:

(i) $\Theta(F)$ is a PN-clause-set, that is, every clause is either positive or negative.

(ii) For every negative clause $N \in \Theta(F)$ we have $\forall\, x \in N : \#_x(F) = 1$ (recall that the negative clauses are the ALO-clauses introduced by the translation Θ).

Call a boolean $F \in \mathcal{MUSAT}_{\delta=1}$ *special* if these two conditions are fulfilled. (These "special" boolean clause-sets constitute exactly the image $\Theta(\mathcal{MUSAT}_{\delta=1})$ of the translation, but we do not need this simple fact here.) Consider a tree representation (T, r, v, ε) of a special F; obviously also all clause-sets given by the subtrees of (T, r) are special again. Now we proof by induction on the height of the tree representation of special formulas F with $n(F) > 0$, that there always exists a negative clause $N \in F$ such that $\forall\, x \in N : \#_{\overline{x}}(F) = 1$, using the standard complement notation for boolean literals here; this proves the lemma by definition of the translation Θ.

If the height of (T, r) is 1, then F is $\{\{v(r)\}, \{\overline{v(r)}\}\}$, and the assertion is true. So assume the height of (T, r) is greater than 1, and consider the left subtree T_0 and the right subtree T_1 of T with associated special $F_0, F_1 \in \mathcal{MUSAT}_{\delta=1}$. If T_0 is not the trivial tree (has more than one node), then by the induction hypothesis there exists a negative clause (non-empty) $N_0 \in F_0$ with $\forall\, x \in N_0 : \#_{\overline{x}}(F_0) = 1$. Now we must have $N_0 \in F$, since otherwise $N_0 \cup \{v\} \in F$, where this clause is neither positive nor negative; using $N := N_0$ proves the assertion (since none of the variables in N_0 occurs in T_1 in this case). So the remaining case is that T_0

is the trivial tree. Again by the induction hypothesis there is a negative clause (non-empty) $N_1 \in F_1$ with $\forall\, x \in N_1 : \#_{\overline{x}}(F_1) = 1$. Either we have $N := N_1 \in F$ or $N := N_1 \cup \{\overline{v}\} \in F$, proving the assertion (in the second case due to the triviality of T_0). ∎

Now we are able to generalise Lemma C.5 in [26] (Part (i) of Theorem 6.16 has been shown for boolean clause-sets in [10]):

Theorem 6.16 *The class $\mathcal{MUSAT}_{\delta=1}$ of minimally unsatisfiable (generalised) clause-sets of deficiency 1 has the following two characterisations:*

(i) For $F \in \mathcal{CLS}$ we have $F \in \mathcal{MUSAT}_{\delta=1}$ if and only if F can be reduced to the clause-set $\{\bot\}$ by applying non-degenerated singular DP-reduction (as long as possible, in any order).

(ii) $\mathcal{MUSAT}_{\delta=1}$ is the class of all clause-sets $F(T, r, v, \varepsilon)$ together with all clause-sets F' derived by literal elimination from such clause-sets.

Proof: Part (i) follows from Lemma 6.15 together with Lemma 3.4 and Lemma 4.30, Part 2(b)i. For Part (ii) it remains to show that every $F \in \mathcal{MUSAT}_{\delta=1}$ can be obtained from some $F(T, r, v, \varepsilon)$ by a (possibly empty) sequence of literal eliminations. We show this by induction on $n(F)$. If $n(F) = 0$, then $F = \{\bot\}$, and we can take the trivial rooted tree. So assume $n(F) > 0$. By Lemma 6.15 there exists a variable $v \in \mathrm{var}(F)$ such that for all $\varepsilon \in D_v$ we have $\#_{(v,\varepsilon)}(F) = 1$; let $C_\varepsilon \in F$ be the unique clause with $(v, \varepsilon) \in C_\varepsilon$. Thus v is a singular DP-variable w.r.t. F. Let $G := \mathrm{DP}_v(F)$; we have $G = (F \setminus \{C_\varepsilon\}_{\varepsilon \in D_v}) \cup \{R\}$, where $R = \bigcup_{\varepsilon \in D_v}(C \setminus \{(v, \varepsilon)\})$. As already argued for Part (i) we have $G \in \mathcal{MUSAT}_{\delta=1}$, and thus we can apply the induction hypothesis to G; the assertion follows now immediately by extending the tree representation of G at the leaf labelled by R by adding new leaves C_ε for $\varepsilon \in D_v$. ∎

Theorem 6.16 yields also two further poly-time decision procedures for the class $\mathcal{MUSAT}_{\delta=1}$ (while two general poly-time decision procedures for the classes $\mathcal{MUSAT}_{\delta=k}$ for $k \in \mathbb{N}$ are given by Corollary 4.10 and Theorem 5.5). To conclude, we characterise the saturated and the marginal elements of $\mathcal{MUSAT}_{\delta=1}$.

Corollary 6.17 *The class $\mathcal{SMUSAT}_{\delta=1}$ of saturated minimally unsatisfiable (generalised) clause-sets of deficiency 1 is exactly the class of all clause-sets $F(T, r, v, \varepsilon)$. It follows that the following conditions are equivalent for a clause-set $F \in \mathcal{CLS}$:*

1. *$F = F(T, r, v, \varepsilon)$ for some deficiency-1 tree representation (T, r, v, ε).*

2. *F is an unsatisfiable 1-regular hitting clause-set of deficiency 1 (i.e., $F \in \mathcal{RHIT}^{\mathrm{sat}=0}_{\mathrm{hd}=1,\delta=1}$).*

3. *F is an unsatisfiable regular hitting clause-set of deficiency 1 (i.e., $F \in \mathcal{RHIT}^{\mathrm{sat}=0}_{\delta=1}$).*

4. *F is an unsatisfiable hitting clause-set of deficiency 1 (i.e., $F \in \mathcal{HIT}^{\mathrm{sat}=0}_{\delta=1}$).*

5. *F is a saturated minimally unsatisfiable clause-set of deficiency 1 (i.e., $F \in \mathcal{SMUSAT}_{\delta=1}$).*

If a minimally unsatisfiable clause-set is hitting, then it is saturated; Corollary 6.17 proves the reverse for deficiency 1 (which does not hold for higher deficiencies).

While saturated minimally unsatisfiable clause-sets do not allow addition of any literal occurrence to any clause without destroying the property of being minimally *unsatisfiable*, on the other end of the spectrum we have **marginal minimally unsatisfiable clause-sets**, which are minimally unsatisfiable clause-sets such that removing any literal occurrence destroys the property of being *minimally* unsatisfiable.

Corollary 6.18 *The class of marginal minimally unsatisfiable (generalised) clause-sets of deficiency* 1 *is exactly the class of all* $F \in \mathcal{MUSAT}_{\delta=1}$ *for which no further literal eliminations are possible, which is equivalent to the property, that for every variable* $v \in \mathrm{var}(F)$ *and every* $\varepsilon \in D_v$ *we have* $\#_{(v,\varepsilon)}(F) = 1$.

Proof: If for a minimally unsatisfiable clause-set F every literal in it occurs exactly once, then obviously it is marginal; Corollary 6.18 proves the reverse for deficiency 1 (which does not hold for higher deficiencies). ∎

We remark that in [35] it is shown that in the boolean case the class of conflict graphs of $F \in \mathcal{MUSAT}_{\delta=1}$ is exactly the class of all connected graphs, while the conflict graphs of saturated (boolean) $F \in \mathcal{MUSAT}_{\delta=1}$ are exactly all complete graphs, and the conflict graphs of marginal (boolean) $F \in \mathcal{MUSAT}_{\delta=1}$ are exactly all trees; furthermore a boolean element of $\mathcal{MUSAT}_{\delta=1}$ is saturated resp. marginal iff the conflict graph is complete respectively a tree.

Finally we remark that in general the decision problem whether a (generalised) clause-set is saturated resp. marginal is D^P-complete as shown in [8] (there for boolean clause-sets, which obviously immediately generalises).

6.5 Stability parameter and minimal variable degree

Let us conclude the chapter by some considerations which summarise certain observations made for which definitely yet the final words are not spoken.

For a multi-clause-set F let **sir(F)**, the **(substitution) stability parameter regarding irredundancy**, be the supremum in $\mathbb{Z}_{\geq -1} \cup \{+\infty\}$ of $n \in \mathbb{N}_0$ such that for all $\varphi \in \mathcal{PASS}$ with $n(\varphi) \leq n$ the multi-clause-set $\varphi * F$ is irredundant. We have the following basic properties.

1. $\mathrm{sir}(F) = -1$ iff F is redundant, $\mathrm{sir}(F) \geq 0$ iff F is irredundant.
2. $\mathrm{sir}(F) = +\infty$ iff $\mathrm{sir}(F) \geq n(F)$ iff F is a hitting clause-set (see Corollary 6.6).
3. Assume that F is unsatisfiable. Then $\mathrm{sir}(F) \geq 1 \Rightarrow F \in \mathcal{SMUSAT}$ by Corollary 6.14, and for boolean F we have equivalence.

For $n(F) > 0$ let the **min-max var-degree** resp. the **minimal var-degree** be defined by

$$\mathbf{mmvd}(F) := \min_{v \in \mathrm{var}(F)} \max_{\varepsilon \in D_v} \#_{(v,\varepsilon)}(F) \in \mathbb{N}$$

$$\mathbf{mvd}(F) := \min_{v \in \mathrm{var}(F)} \#_v(F) \in \mathbb{N}$$

Lemma 6.19 *Consider a generalised multi-clause-set F and a variable $v \in \mathrm{var}(F)$ which is not pure for F (i.e., $\forall\, \varepsilon \in D_v : \#_{(v,\varepsilon)}(F) \geq 1$), such that $\#_v(F) = \mathrm{mvd}(F)$. Then we have for $\varepsilon \in D_v$ the following.*

1. *$\delta(\langle v \to \varepsilon\rangle * F) = \delta(F) - s_{(v,\varepsilon)}(F) + |D_v| - 1$. (Compare Lemma 4.28, Part 3.)*
2. *Assume $\mathrm{sir}(F) \geq 1$ and that F is unsatisfiable.*
   (a) *$s_{(v,\varepsilon)}(F) \leq \delta(F) + |D_v| - 2$.*
   (b) *If v is non-trivial (i.e., $|D_v| \geq 2$), then $\#_{(v,\varepsilon)}(F) \leq \delta(F)$.*

Proof: Part 1 follows by the observation, that $\mathrm{var}(\langle v \to \varepsilon\rangle * F) = \mathrm{var}(F) \setminus \{v\}$ (if another variable w would vanish, then every occurrence of w would be in a clause C with some $(v, \varepsilon') \in C$ for $\varepsilon' \in D_v \setminus \{\varepsilon\}$, and so $\#_w(F) \leq s_{(v,\varepsilon)}(F) = \#_v(F) - \#_{(v,\varepsilon)}(F) < \#_v(F) = \mathrm{mvd}(F) \leq \#_w(F)$). For Part 2 we have $\delta(\langle v \to \varepsilon\rangle * F) \geq 1$, and thus Part 2a follows. For Part 2b consider $\varepsilon' \in D_v \setminus \{\varepsilon\}$. By Part 2a we have $s_{(v,\varepsilon')}(F) \leq \delta(F) + |D_v| - 2$, where $s_{(v,\varepsilon')}(F) = \#_{(v,\varepsilon)}(F) + \sum_{\varepsilon'' \in D_v \setminus \{\varepsilon,\varepsilon'\}} \#_{\varepsilon''}(F) \geq \#_{(v,\varepsilon)}(F) + |D_v| - 2$. ∎

Corollary 6.20 *For an unsatisfiable generalised clause-set F with $\mathrm{sir}(F) \geq 1$ we have*

$$\mathrm{mmvd}(F) \leq \delta(F)$$

(in case of $n(F) > 0$).

Proof: Eliminating all trivial variables from F we obtain the clause-set F' with $\delta(F') = \delta(F)$ and $c(F') = c(F)$; now the assertion follows by Part 2b of Lemma 6.19. ∎

Since every minimally unsatisfiable (generalised) clause-set can be saturated (see Lemma 6.12), and every boolean saturated minimally unsatisfiable clause-set F fulfils $\mathrm{sir}(F) \geq 1$, we get for arbitrary *boolean* $F \in \mathcal{MUSAT}$ the upper bound $\mathrm{mmvd}(F) \leq \delta(F)$ (as shown in [26]). For generalised minimally unsatisfiable clause-sets we showed this upper bound in Lemma 6.15 for the simplest case $\delta(F) = 1$, while the general case is open.

7 Conclusion and open problems

The first purpose of this article was to set the stage for the study of generalised clause-sets as sets of "no-goods", where literals are given by one "forbidden value": We defined and summarised the basic properties of syntax, semantics, resolution calculus and autarky systems. Then we considered the generalisation of the notion of deficiency for these generalised clause-sets, and we studied the basic autarky system related to this notion, namely matching autarkies. We showed fixed parameter tractability of generalised clause-sets in the maximal deficiency. For autarky systems both the application of autarkies as reductions and the properties of autarky-freeness, i.e., lean clause-sets are of interest. Lean clause-sets are a generalisation of minimally unsatisfiable clause-sets, for which we considered the basic problem, when the property of being minimally unsatisfiable is preserved under application of partial assignments, and we characterised also minimally unsatisfiable clause-sets of minimal deficiency. Besides using the generalised tools transferred from the boolean case, also the structure preserving properties of the boolean translation are important, and we investigated basic cases.

7.1 Matching autarkies

In Corollary 4.10 polynomial-time satisfiability decision in the maximal deficiency is shown, using the existence of matching-maximum satisfying assignments; the question now is whether one can show also *fixed-parameter tractability* in this way, using some form of local search which considers only matching-maximum assignments in some form. Implementations of the various poly-time algorithms (especially the algorithm exploited in the proof of Theorem 5.5) need to be carried out in order to study whether interesting applications exist (and also to judge whether "native algorithms" for generalised clause-sets are preferable here, or whether the boolean translation is superior).

Regarding practical applications, one should implement reduction by matching autarkies, and also the extension in Lemma 4.32, and study its uses. Regarding Lemma 4.32 one should study the underlying autarky system.

In general it seems that applications of "expensive" algorithms is more fruitful for harder problems like QBF (it seems most appropriate to restrict autarkies for quantified boolean formulas to existential variables).

7.2 Satisfiable minimally matching-unsatisfiable clause-sets

The class of minimally matching-unsatisfiable clause-sets is exactly $\mathcal{MLEAN}_{\delta=1}$, the set of matching lean clause-sets of deficiency one. The unsatisfiable elements we know quite well by $\mathcal{USAT} \cap \mathcal{MLEAN}_{\delta=1} = \mathcal{MUSAT}_{\delta=1}$, however the satisfiable elements, that is, the set $\mathcal{SAT} \cap \mathcal{MLEAN}_{\delta=1}$ seems to exhibit a much richer structure; see for example the special clause-set at the end of Subsection 4.3 (which in fact is a 1-regular hitting-set — even for this special sub-class we know not much).

7.3 Minimally unsatisfiable clause-sets of low deficiency

Having transferred the characterisation of minimally unsatisfiable clause-sets of deficiency one from the boolean case in Subsection 6.4, the next question concerns the generalisation of the structure of boolean $\mathcal{MUSAT}_{\delta=2}$ as studied in [5]. This generalisation seems to be not straightforward, but we believe that minimally unsatisfiable generalised clause-sets of deficiency two are still quite close to the boolean case (while from deficiency three on generalised clause-sets behave more wildly).

A key tool for the study of boolean minimally unsatisfiable clause-sets in the observation in [26] that for every boolean minimally unsatisfiable clause-set F with $n(F) > 0$ there exists a variable $v \in \mathrm{var}(F)$ such that for both $\varepsilon \in \{0, 1\}$ we have $\#_{(v,\varepsilon)}(F) \leq \delta(F)$; see Lemma 6.19 for a discussion of this subject.

7.4 Regular hitting clause-sets

Regarding the base case of deficiency 1 for minimally unsatisfiable clause-sets, a natural question is, whether every regular unsatisfiable hitting clause-set F has necessarily deficiency 1 (we know that for every minimally unsatisfiable clause-set the deficiency is at least 1, so the upper bound $\delta(F) \leq 1$ is in question here) ? We conjecture that this is the case:

Conjecture 7.1 $\mathcal{RHIT}^{\mathrm{sat}=0} = \mathcal{RHIT}^{\mathrm{sat}=0}_{\delta=1}$.

Obviously $\mathcal{RHIT}^{\mathrm{sat}=0}_{\mathrm{hd}=r}$ is empty for $r \geq 2$, since no resolution is possible here. From Conjecture 7.1 it would follow by Corollary 6.17, that the class of unsatisfiable regular hitting (generalised) clause-sets is equal to the class of saturated minimally unsatisfiable clause-sets of deficiency 1, generalising Corollary 34 in [33].

In [33] the property $\delta(F) \leq 1$ was actually shown for arbitrary (not necessarily unsatisfiable) boolean regular hitting clause-sets F. Whether this holds for generalised clause-sets seems to be a non-trivial problem, since the notion of hermitian rank exploited in [33] is specifically tailored to the use of matrices (which are inherently two-dimensional) and real numbers (with positive and negative values) and hence boolean clause-sets. Though we do not know how to prove it, we nevertheless believe that the generalisation holds true:

Conjecture 7.2 $\mathcal{RHIT} = \mathcal{RHIT}_{\delta \leq 1}$.

Note that Conjecture 7.2 implies Conjecture 7.1 (using Corollary 4.22). In the terminology of graph partitions, Conjecture 7.2 generalises "Witsenhausen's Theorem", the special case of the Graham-Pollak Theorem asserting that every biclique partition of a complete graph K_m needs at least $m - 1$ bicliques:

Now we allow to partition the edge set of $r \cdot K_m$ (exactly r edges joining two different nodes) into complete *multipartite* graphs, where every complete k-partite component ($k \geq 2$) contributes the "cost" $k - 1$, and Conjecture 7.2 says, that the total cost must be at least $m - 1$ (allowing only $k = 2$ is the Theorem of Witsenhausen, while allowing only $k = m$ is trivial).

7.5 Multihitting clause-sets

As touched upon in Lemma 6.10, the SAT problem for bihitting clause-sets is essentially the same as the hypergraph transversal problem, and whether the latter problem can be decided in polynomial time is a long outstanding open question. Being optimistic about the potential of (generalised) clause-sets to provide a unifying framework for (hard) graph and hypergraph problems, we propose:

Conjecture 7.3 *Satisfiability decision for multihitting (generalised) clause-sets can be done in polynomial time.*

References

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

[2] Carlos Ansótegui and Felip Manyà. Mapping problems with finite-domain variables to problems with boolean variables. In Hoos and Mitchell [23], pages 1–15. ISBN 3-540-27829-X.

[3] Andrew B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results.* PhD thesis, University of Oregon, March 1995. The ps-file can be down loaded using the URL `http://www.cirl.uoregon.edu/baker/thesis.ps.gz`.

[4] Hans Kleine Büning. An upper bound for minimal resolution refutations. In Georg Gottlob, Etienne Grandjean, and Katrin Seyr, editors, *Computer Science Logic 12th International Workshop, CSL'98*, volume 1584 of *Lecture Notes in Computer Science*, pages 171–178. Springer, 1999.

[5] Hans Kleine Büning. On subclasses of minimal unsatisfiable formulas. *Discrete Applied Mathematics*, 107:83–98, 2000.

[6] Hans Kleine Büning and Xishun Zhao. On the structure of some classes of minimal unsatisfiable formulas. *Discrete Applied Mathematics*, 130:185–207, 2003.

[7] Hans Kleine Büning and Xishun Zhao. Extension and equivalence problems for clause minimal formulae. *Annals of Mathematics and Artificial Intelligence*, 43:295–306, 2005.

[8] Hans Kleine Büning and Xishun Zhao. The complexity of some subclasses of minimal unsatisfiable formulas. *Journal on Satisfiability, Boolean Modeling and Computation*, to appear, 2007.

[9] Nadia Creignou and Hervé Daudé. Generalized satisfiability problems: minimal elements and phase transitions. *Theoretical Computer Science*, 302:417–430, 2003.

[10] Gennady Davydov, Inna Davydova, and Hans Kleine Büning. An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF. *Annals of Mathematics and Artificial Intelligence*, 23:229–245, 1998.

[11] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, San Francisco, 2003. ISBN 1-55860-890-7; QA76.612.D43 2003.

[12] Michael R. Dransfield, Lengning Liu, Victor W. Marek, and Miroslaw Truszczyński. Satisfiability and computing van der Waerden numbers. *The Electronic Journal of Combinatorics*, 11(#R41), 2004.

[13] Pierre Duchet. Hypergraphs. In Ronald L. Graham, Martin Grötschel, and László Lovász, editors, *Handbook of Combinatorics, Volume 1*, chapter 7, pages 381–432. North-Holland, Amsterdam, 1995. ISBN 0-444-82346-8; QA164.H33 1995.

[14] Herbert Fleischner, Oliver Kullmann, and Stefan Szeider. Polynomial–time recognition of minimal unsatisfiable formulas with fixed clause–variable difference. *Theoretical Computer Science*, 289(1):503–516, November 2002.

[15] Herbert Fleischner and Stefan Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. Technical Report TR00-049, Electronic Colloquium on Computational Complexity (ECCC), July 2000.

[16] T. Fliti and G. Reynaud. Sizes of minimally unsatisfiable conjunctive normal forms. Faculté des Sciences de Luminy, Dpt. Mathematique-Informatique, 13288 Marseille, France, November 1994.

[17] John Franco and Allen Van Gelder. A perspective on certain polynomial-time solvable classes of satisfiability. *Discrete Applied Mathematics*, 125:177–214, 2003.

[18] Alan M. Frisch and Timothy J. Peugniez. Solving non-boolean satisfiability problems with stochastic local search. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 282–288, 2001.

[19] Nicola Galesi and Oliver Kullmann. Polynomial time SAT decision, hypergraph transversals and the hermitian rank. In Hoos and Mitchell [23], pages 89–104. ISBN 3-540-27829-X.

[20] Enrico Giunchiglia and Armando Tacchella, editors. *Theory and Applications of Satisfiability Testing 2003*, volume 2919 of *Lecture Notes in Computer Science*, Berlin, 2004. Springer. ISBN 3-540-20851-8.

[21] P.R. Herwig, M.J.H. Heule, P.M. van Lambalgen, and H. van Maaren. A new method to construct lower bounds for van der Waerden numbers. *The Electronic Journal of Combinatorics*, 12(#R00), 2005.

[22] Shlomo Hoory and Stefan Szeider. A note on unsatisfiable k-CNF formulas with few occurrences per variable. *SIAM Journal on Discrete Mathematics*, 20(2):523–528, 2006.

[23] Holger H. Hoos and David G. Mitchell, editors. *Theory and Applications of Satisfiability Testing 2004*, volume 3542 of *Lecture Notes in Computer Science*, Berlin, 2005. Springer. ISBN 3-540-27829-X.

[24] Michal Kouril. $W(2, 6) = 1132$, January 2007. See `http://www.cs.uc.edu/~kourilm/`.

[25] Michal Kouril and John Franco. Resolution tunnels for improved SAT solver performance. In Fahiem Bacchus and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing 2005*, volume 3569 of *Lecture Notes in Computer Science*, pages 143–157, Berlin, 2005. Springer. ISBN 3-540-26276-8.

[26] Oliver Kullmann. An application of matroid theory to the SAT problem. In *Fifteenth Annual IEEE Conference on Computational Complexity (2000)*, pages 116–124.

[27] Oliver Kullmann. New methods for 3-SAT decision and worst-case analysis. *Theoretical Computer Science*, 223(1-2):1–72, July 1999.

[28] Oliver Kullmann. Restricted versions of extended resolution. *The Bulletin of Symbolic Logic*, 5(1):119, 1999. Abstracts of contributed papers of the Logic Colloquium' 98, Prague, Czech Republic, August 9 - 15, 1998.

[29] Oliver Kullmann. Investigations on autark assignments. *Discrete Applied Mathematics*, 107:99–137, 2000.

[30] Oliver Kullmann. On the use of autarkies for satisfiability decision. In Henry Kautz and Bart Selman, editors, *LICS 2001 Workshop on Theory and Applications of Satisfiability Testing (SAT 2001)*, volume 9 of *Electronic Notes in Discrete Mathematics (ENDM)*. Elsevier Science, June 2001.

[31] Oliver Kullmann. Lean clause-sets: Generalizations of minimally unsatisfiable clause-sets. *Discrete Applied Mathematics*, 130:209–249, 2003.

[32] Oliver Kullmann. On the conflict matrix of clause-sets. Technical Report CSR 7-2003, University of Wales Swansea, Computer Science Report Series, 2003.

[33] Oliver Kullmann. The combinatorics of conflicts between clauses. In Giunchiglia and Tacchella [20], pages 426–440. ISBN 3-540-20851-8.

[34] Oliver Kullmann. Conflict matrices and multi-hitting clause-sets. Extended Abstract for the Guangzhou Symposium on Satisfiability and its Applications, September 2004.

[35] Oliver Kullmann. The conflict matrix of (multi-)clause-sets — a link between combinatorics and (generalised) satisfiability problems. In preparation; continuation of [32], 2004.

[36] Oliver Kullmann. Upper and lower bounds on the complexity of generalised resolution and generalised constraint satisfaction problems. *Annals of Mathematics and Artificial Intelligence*, 40(3-4):303–352, March 2004.

[37] Oliver Kullmann. Constraint satisfaction problems in clausal form: Autarkies, minimal unsatisfiability, and applications to hypergraph inequalities. In Nadia Creignou, Phokion Kolaitis, and Heribert Vollmer, editors, *Complexity of Constraints*, number 06401 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006. `http://drops.dagstuhl.de/opus/volltexte/2006/803`.

[38] Oliver Kullmann. Polynomial time SAT decision for complementation-invariant clause-sets, and sign-non-singular matrices. In Joao Marques-Silva and Karem A. Sakallah, editors, *Theory and Applications of Satisfiability Testing - SAT 2007*, volume 4501 of *Lecture Notes in Computer Science*, pages 314–327. Springer, 2007. ISBN 978-3-540-72787-3.

[39] Oliver Kullmann and Horst Luckhardt. Algorithms for SAT/TAUT decision based on various measures. Preprint, 71 pages; the ps-file can be obtained from `http://cs-svr1.swan.ac.uk/~csoliver`, December 1998.

[40] Oliver Kullmann, Inês Lynce, and João Marques-Silva. Categorisation of clauses in conjunctive normal forms: Minimally unsatisfiable sub-clause-sets and the lean kernel. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006*, volume 4121 of *Lecture Notes in Computer Science*, pages 22–35. Springer, 2006. ISBN 3-540-37206-7.

[41] Oliver Kullmann, Victor W. Marek, and Mirosław Truszczyński. Computing autarkies and properties of the autarky monoid. In preparation, January 2007.

[42] Bruce Landman, Aaron Robertson, and Clay Culver. Some new exact van der Waerden numbers. arXiv:math.CO/0507019 v1, July 2005.

[43] Paolo Liberatore. Redundancy in logic I: CNF propositional formulae. *Artificial Intelligence*, 163:203–232, 2005.

[44] László Lovász and Michael D. Plummer. *Matching Theory*, volume 121 of *Mathematics Studies*. North-Holland, 1986.

[45] David G. Mitchell and Joey Hwang. 2-way vs. d-way branching for CSP. In Peter van Beek, editor, *Principles and Practice of Constraint Programming — CP 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 343–357. Springer, 2005. ISBN 3-540-29238-1.

[46] Christos H. Papadimitriou and David Wolfe. The complexity of facets resolved. *Journal of Computer and System Sciences*, 37:2–13, 1988.

[47] Steven Prestwich. Local search on SAT-encoded colouring problems. In Giunchiglia and Tacchella [20], pages 105–119. ISBN 3-540-20851-8.

[48] Alexander Schrijver. *Combinatorial Optimization*, volume A. Springer, Berlin, 2003. ISBN 3-540-44389-4; Series Algorithms and Combinatorics, no. 24.

[49] Stefan Szeider. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. *Journal of Computer and System Sciences*, 69(4):656–674, 2004.

[50] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8:85–89, 1984.

[51] Vitaly I. Voloshin. *Coloring Mixed Hypergraphs: Theory, Algorithms and Applications*. Fields Institute Monographs. American Mathematical Society, 2002. ISBN 0-8218-2812-6.

[52] Xishun Zhao and Ding Decheng. Two tractable subclasses of minimal unsatisfiable formulas. *Science in China (Series A)*, 42(7):720–731, July 1999.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/