PRIFYSGOL CYMRU ABERTAWE
UNIVERSITY OF WALES SWANSEA

UNIVERSITY OF WALES SWANSEA

REPORT SERIES

Constraint satisfaction problems in clausal form: Autarkies, minimal unsatisfiability, and applications to hypergraph inequalities

by

Oliver Kullmann

Report # CSR 13-2006

Dagstuhl Seminar Proceedings 06401
Complexity of Constraints
http://drops.dagstuhl.de/opus/volltexte/2006/803

Constraint satisfaction problems in clausal form: Autarkies, minimal unsatisfiability, and applications to hypergraph inequalities

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/~csoliver

September 23, 2006

Abstract

We investigate in depth a natural generalisation of boolean formulas in conjunctive normal forms (or "clause-sets") allowing *non-boolean variables*; this generalisation is closely related to "sets of no-goods" in the AI literature, and we will argue that it is the right generalisation of boolean clause-sets maintaining essential *combinatorial* properties. First we build 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. We obtain *fixed parameter tractability* (FPT) of satisfiability decision for generalised clause-sets, using as parameter a suitably generalised notion of *maximal deficiency*, where in the boolean case the deficiency is the difference between the number of clauses and the number of variables. 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* (autarkies are generalisations of satisfying assignments), and we concentrate especially on *matching autarkies* (based on matching theory) and special cases of *linear autarkies* (based on linear programming and linear algebra). Minimally unsatisfiable (generalised) clause-sets are "lean", i.e., they do not admit non-trivial autarkies. Besides minimally unsatisfiable clause-sets we consider also "irredundant" clause-sets, which allow for satisfiable clause-sets, with a special emphasise on "hitting clause-sets" (which are irredundant in a very strong sense) and the generalisation to "multihitting" clause-sets.

Autarky theory provides methods for deriving lower bounds on the deficiency $\delta(F)$ of generalised clause-sets F, where $\delta(F)$ relates the number of

*Supported by EPSRC Grant GR/S58393/01

clauses and the number of variables of F, while the structural properties of minimally unsatisfiable (generalised) clause-sets enable classifications of minimally unsatisfiable clause-sets of low deficiency. Using the canonical translation of *hypergraph colouring problems* into (generalised) clause-sets, we are able to interprete classical basic results from hypergraph theory within this more general framework. We provide a general method for proving the existence of a matching in the bipartite graph of a hypergraph G covering all vertex nodes (if such a matching exists, then G must have at least as meany hyperedges as vertices). This methods allows to derive the (generalised) Fisher inequality, that a non-trivial pairwise balanced design must have as least as many blocks as points, as well as a the bound by Seymour, that a minimally non-2-colourable hypergraph must have at least as many hyperedges as vertices. Regarding Seymour's inequality, without any additional work the result generalises to non-k-colourable hypergraphs for $k \geq 2$: Consider a hypergraph G and the canonical translation of the k-colouring problem for G into a (generalised) "colouring" clause-set $F_{[k]}(G)$. "Minimally non k-colourability" of G is equivalent to $F_{[k]}(G)$ being minimally unsatisfiable, a complicated and "fragile" notion. But the lower bound on the deficiency on $F_{[k]}(G)$ does actually not hinge on the minimal unsatisfiability, but only on the absence of a special class of linear autarkies (namely "balanced linear autarkies"), and so we can use the "smooth" generalisation of minimally unsatisfiable clause-sets by (balanced linearly) lean clause-sets. Finally we touch upon the characterisation of minimally unsatisfiable colouring clause-sets with minimal deficiency, recasting the characterisation of "self-blocking square condensers" by Seymour.

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 [20, 45, 2] for various techniques), however often they cause performance problems, and 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 report[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 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*.

One driving force for the study of generalised clause-sets comes from the applications of learning falsifying partial assignments in (generalised) SAT algorithms — the notion of a generalised clause as a "no-good" embodies this point of view (a clause encodes the partial assignment leading to a failure, and a partial assignment falsifies this clause iff it extends the failing assignment). In [37] a theoretical study of such uses of generalised clauses was initiated, however in this article our main

[1)] substantially extending the early report [38]

application area is *hypergraph theory*. The emphasise is on building up a framework for showing that certain classes of hypergraphs have at least as many hyperedges as vertices, and for classifying those hypergraphs within this class for which equality holds. This is done by reformulating the inequality as a lower bound on the deficiency of certain generalised clause-sets, while the classification task becomes the task of classifying certain minimally unsatisfiable clause-sets of minimal deficiency.

As a (very) first evidence for the potential power of autarky theory as a unifying framework in combinatorics we will see, that early results of Seymour on critically colourable hypergraphs can be extended naturally by interpreting hypergraph colouring as a satisfiability problem for generalised clause-sets — here the formulation as a (generalised) satisfiability problem does not mask structure (the transformation is canonical and structure-preserving), but actually helps to *reveal* structure.

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 [19] 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 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 $F \in \mathcal{MUSAT}$ the property $\forall\, F' \subset F : \delta(F') < \delta(F)$ has been shown; using $\delta^*(F) := \max_{F' \subseteq F} \delta(F')$ 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 $\delta(\top) = 0$). Furthermore 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 [11] a (poly-time) characterisation of $\mathcal{MUSAT}(1)$ has been obtained, followed by a characterisation of $\mathcal{MUSAT}(2)$ in [6], while in [51] 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 [5] 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 [5] it has been conjectured that in fact all classes $\mathcal{MUSAT}(k)$ are in P.

This conjecture has been proven true in [28] (using tools from matroid theory), where more generally 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. Actually

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 [11], 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 $B(F)$ (the independent subsets of $T(F)$ are the matching satisfiable sub-clause-sets of F). Independently also in [17] 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 [17], the present author joint the team in [16]. Actually refining the techniques from [28], in [48] 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 4.4 an earlier version is discussed, which doesn't seem to have the right properties), in Corollary 4.9 the "satisfiability-based" approach from [16] yields polynomial time satisfiability decision for generalised clause-sets with bounded maximal deficiency. Generalising fixed-parameter tractability turns out not to be straightforward (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 [31], with emphasise on *matching autarkies* as introduced in [33].

A key point for structural investigations in (generalised) clause-sets is to understand the effects of applying partial assignments (see for example [8, 7], 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").

Finally, we leave generalisations behind, and with the third main subject of this paper, the applications of autarky theory to hypergraph theory, we enter new ground; I believe that such combinations of (generalised) satisfiability theory and combinatorial theory have a future, waiting for us to be explored.

1.2 Translating hypergraph colouring

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 $E \in E(G)$ is "monochromatic" (that is, there must be vertices $v, w \in E$ with $f(v) \neq f(w)$). Translating this colouring problem into a generalised satisfiability problem $F_C(G)$ is straightforward[2)]: For each hyperedge $E \in E(G)$ and each colour $\varepsilon \in C$ form the clause $\{v \neq \varepsilon : v \in E\}$,

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

and $F_C(G)$ is the set of all these clauses; 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 [14, 25], and it seems that SAT solvers are performing quite well on them, and that possibly SAT solvers could 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 $\{1,\ldots,n\}$, while the hyperedges of $\mathrm{WH}(m,n)$ are the subsets $E \subseteq \{1,\ldots,n\}$ of size m which form an arithmetic progression (that is, for every E there exist $a, d \in \{1,\ldots,n\}$ with $E = \{a+i\cdot d : i \in \{0,\ldots,m-1\}\}$); 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 $\mathrm{F_W}(k,m,n) := F_{\{1,\ldots,k\}}(\mathrm{WH}(m,n))$, 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)]). 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 [41]).

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

We notice that in the same vein 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, 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

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

[4)] The translation is the core of two translations discussed in [14] — the additional constraints used in [14] just express the structural property of a generalised clause-set, that every variable gets exactly one value of its domain.

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 $C_{i,\vec{x},\vec{y}} := \{\vec{x}_i \neq \vec{y}_i : i \in \{1,\ldots,m\}\}$. 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 translation $F(G_1, G_2, L)$ as well as $F(\mathcal{A},\mathcal{B})$ is "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 $\vec{x} \in R_i$, with domain R'_i. The constraints are the unit clauses $\{(i,\vec{x}) \neq \vec{y}\}_{i\in\{1,\ldots,m\}}$ expressing that a variable $(i,\vec{x})$ with $\vec{x}_k = \vec{x}_{k'}$ for some k, k' must not get a value $\vec{y} \in R'_i$ with $\vec{y}_k \neq \vec{y}_{k'}$, and the binary clauses $\{(i,\vec{x}) \neq \vec{y}, (i',\vec{x}') \neq \vec{y}'\}$ expressing that if variable $(i,\vec{x})$ gets a value $\vec{y}$, then a variable $(i',\vec{x}')$ with $\vec{x}_k = \vec{x}'_k$ for some k must not get a value $\vec{y}'$ with $\vec{y}_k \neq \vec{y}'_k$.

1.3 Hypergraph inequalities

In this article we consider problems from hypergraph theory related to the notion of deficiency. In [47] it was proven that a hypergraph which is minimally non-2-colourable and has no isolated vertices has at least as many edges as vertices. By recognising that this property only needs the non-existence of certain *autarkies*, as in the case of Tarsi's lemma, we can generalise this result to the statement, that if a hypergraph without isolated vertices is minimally non-k-colourable for some $k \geq 2$, then it has at least as many edges as vertices. A similar application (but without adding something new) is given to the Fisher inequality in design theory. Both inequalities can be derived from a meta-theorem, which allows to translate back leanness properties of generalised clause-sets encoding hypergraph colouring problems.

I hope that these (first) examples can demonstrate the potential of considering hypergraph colouring problems as special generalised satisfiability problems. The virtue of embedding hypergraph colouring problems into the richer structure of generalised satisfiability problems can be seen in the substitutional closedness of generalised satisfiability problems, which is noticeable missing from the notion of hypergraph colouring, while on the other hand generalised satisfiability as presented in this paper seems close enough to hypergraph colouring problems, so that techniques can be transferred.[5)]. So I see great potential for the study of generalised satisfiability problems as a unifying framework for combinatorics, while on the other hand for example many treasures of (hyper)graph theory are still waiting to be discovered for the SAT community (see Subsection 8.3 for an intriguing example from [47]).

[5)] According to my personal experience researchers from graph theory and combinatorics sometimes view the satisfiability problem as "pure complexity theory", dismissing its combinatorial nature. For example in [42], where (boolean) resolution is transferred to the 2-colouring problem for hypergraphs, the point is stressed that satisfiability is a special case of the 2-colouring problem for hypergraphs by reducing the SAT problem for boolean conjunctive normal form in a relatively simple way to the hypergraph 2-colouring problem — however that the inverse transformation is even simpler (not using any sort of "gadget") is not mentioned in this paper.

1.4 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 [20] (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 [44], 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 when splitting on a variable v, each branch fixing a value of v (the corresponding form of resolution has been studied in some depth in [37] (generalised there through the use of oracles)). 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 (i.e., splitting on the negative literal "$v \neq \varepsilon$") and a domain restriction $D_v \mapsto D_v \setminus \{\varepsilon\}$ in the other branch (splitting on the positive literal "$v = \varepsilon$"): 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).[6)] 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 in 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

[6)] Note that 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.

the reason why in the upcoming `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).[7)]

1.5 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 (while the hypergraphs notions and notations are collected in Subsection 7.2). 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 (first studied in [19]) is generalised in a natural way to generalised clause-sets, based on the generalised notion of deficiency. Theorem 4.7 in Subsection 4.2 as the first main result, guaranteeing the existence of satisfying assignments "close enough" to matching satisfying assignments, is established for generalised clause-sets, so that in Corollary 4.8 poly-time satisfiability decision for generalised clause-sets with bounded maximal deficiency can be derived, generalising and strengthening the approach from [16] (proving fixed parameter tractability with respect to the maximal deficiency has to wait until the next section, where further tools are provided). 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 4.21 (every generalised minimally unsatisfiable clause-set has deficiency at least one). In Subsection 4.4 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 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

[7)] This discussion shows in my opinion the main reason, while generalising boolean reasoning proved to be difficult in the past, and (boolean) SAT solvers have an edge: 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.

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 [18]; 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 6.14 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).

The main result of **Section 7** on linear autarkies for generalised clause-sets is Theorem 7.8, providing a general criterion for the existence of a matching between the vertices and the hyperedges of a hypergraph, such that all vertices are covered, and generalising one of the main results from [1], namely that for a minimally non-2-colourable hypergraph there exists a matching in the associated bipartite graph which covers all vertex nodes. It is based on Lemma 7.2, which exploits balanced linear autarkies, together with basic properties of (linear) autarkies regarding the translation of hypergraphs into generalised clause-sets studied in Subsection 7.3. As a first application we show in Lemma 7.10 how to derive the (generalised) Fisher inequality for pairwise balanced designs.

A second application of Theorem 7.8 is given in **Section 8**, where we consider hypergraph colouring. In Corollary 8.2 we generalise a well-known early result of Seymour (similar to Tarsi's Lemma, and also based on autarky theory now), which yields a lower bound on the deficiency of minimally unsatisfiable colouring clause-sets, and we discuss the relation to "crown decomposition". Also a refinement of the chromatic number, namely the "autarky number", is outlined. We conclude in Subsection 8.3 by interpreting a characterisation of Seymour of all intersecting critical 3-colourable hypergraphs as (essentially) a classification of minimally unsatisfiable multihitting colouring clause-sets with (relative) minimal deficiency; see Theorem 8.14 for the full classification.

Finally we present a collection of open problems in Section 9.

2 Preliminaries

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 [37], 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 $v \in \mathcal{VA}$ by $\boldsymbol{D_v}$ we denote the associated (value-)domain (thus variables have fixed (value-)domains, and change of domain (for example removal of values) must be performed by renaming). 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; by the above cardinality assumption there is a bijection between $\mathcal{VA}$ and $\mathcal{VA}_{\{0,1\}}$).

A **partial assignment** is a map φ with finite domain $\boldsymbol{\mathrm{var}(\varphi)} := \mathrm{dom}(\varphi) \subseteq \mathcal{VA}$, such that for all $v \in \mathrm{var}(\varphi)$ we have $\varphi(v) \in D_v$. The domain size of a partial assignment φ is denoted by $\boldsymbol{n(\varphi)} := |\mathrm{var}(\varphi)| \in \mathbb{N}_0$. 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 $\langle v_1 \to \varepsilon_1, \ldots, v_m \to \varepsilon_m \rangle$ 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 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 the above representation, 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 first representation of partial assignments.

2.2 Graphs

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 *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 $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ resp. $f(A \cup B) + f(A \cap B) \geq f(A) + f(B)$, 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 *neighbourset* $\Gamma_G(A)$ is defined as the set of vertices adjacent to at least one element of A. The function $A \subseteq V(G) \mapsto |\Gamma(A)|$ 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. If G is a bipartite graph with bipartition (A, B) (also called "colour classes"), then the maximal number of vertices of A which can be covered by a matching is $|A| - \delta^*(A)$, where $\delta^*(A) := \max_{A' \subseteq A} \delta(A')$ (see for example Theorem 22.2 in [46], where the notion of "transversals" or "systems of distinct representatives" of set system is used (not to be mixed up with "transversals" in hypergraphs), and where the set system is $(\Gamma_G(\{a\}))_{a \in A}$).

See Subsection 7.2 for notions and notations regarding *hypergraphs*.

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 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 [37, 32] for a general, axiomatic framework for "generalised satisfiability problems", while in Subsection 2.3 of [37] generalised clause-sets are discussed, and in Section 2 of [35] boolean multi-clause-sets are considered (see also [34] 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 always 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 $\mathbf{var}(v, \varepsilon) := v$ and $\mathbf{val}(v, \varepsilon) := \varepsilon$. The set of all literals is denoted by $\mathcal{LIT}$, and for any $\mathcal{VA}' \subseteq \mathcal{VA}$ we write $\mathcal{LIT}(\mathcal{VA}') := \{x \in \mathcal{LIT} : \mathrm{var}(x) \in \mathcal{VA}'\}$ 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 $\mathcal{CL}$. 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 $\mathcal{CL}(\mathcal{VA}') := \{C \in \mathcal{CL} : \mathrm{var}(C) \subseteq \mathcal{VA}'\}$ 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 $\varphi_C \in \mathcal{PASS}$ 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 $C_\varphi \in \mathcal{CL}$ 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.[8)]

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 $C \in F$ 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 $\mathbf{val}_v(F) := \{\varepsilon \in D_v \mid \exists\, C \in F : (v, \varepsilon) \in C\}$. 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 $\top$.

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

1. $\#_{(v,\varepsilon)}(F) := \sum_{C \in F, (v,\varepsilon) \in C} F(C) \in \mathbb{N}_0$ measures the number of occurrences of a literal;
2. $\#_v(F) := \sum_{\varepsilon \in D_v} \#_{(v,\varepsilon)}(F) = \sum_{C \in F, v \in \mathrm{var}(C)} F(C) \in \mathbb{N}_0$ measures the number of occurrences of a variable;
3. $s_{(v,\varepsilon)}(F) := \sum_{\varepsilon' \in D_v \setminus \{\varepsilon\}} \#_{(v,\varepsilon)}(F) = \#_v(F) - \#_{(v,\varepsilon)}(F) \in \mathbb{N}_0$ 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. $n(F) := |\mathrm{var}(F)| \in \mathbb{N}_0$ measures the number of variables;
5. $c(F) := \sum_{C \in F} F(C) \in \mathbb{N}_0$ measures the number of clauses;

[8)] 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).

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;
7. $\mathbf{mds(}\boldsymbol{F}\mathbf{)} := \max_{v \in \mathrm{var}(F)} |D_v| \in \mathbb{N}_0$ measures the maximal domain size.

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 $\boldsymbol{\mathcal{C}_{f \leq b}}$ for some $b \in \mathbb{R}$ we denote the set of all $F \in \mathcal{C}$ with $f(F) \leq b$; analogously we define $\mathcal{C}_{f=b}$, $\mathcal{C}_{f \geq b}$ 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 $\mathcal{MCLS}_{f \leq b}^{g \leq b'}$. Finally, for $p \in \mathbb{N}_0$ we denote by $\boldsymbol{p}\text{-}\boldsymbol{\mathcal{MCLS}}$ resp. $\boldsymbol{p}\text{-}\boldsymbol{\mathcal{CLS}}$ the set of multi-clause-sets resp. clause-sets F such that for $C \in F$ we have $|C| \leq p$.

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

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

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 $\boldsymbol{\mathfrak{S}_V(F)}$ be the set of $\varphi \in \mathcal{PASS}(V)$ with $\varphi * F = \top$, while $\boldsymbol{\mathfrak{F}_V(F)}$ is the set of $\varphi \in \mathcal{PASS}(V)$ with $\bot \in \varphi * F$. Thus F is satisfiable iff $\mathfrak{S}_V(F) \neq \emptyset$; and for any clause C with $\mathrm{var}(C) \subseteq V$ we have

$$|\mathfrak{F}_V(\{C\})| = |\mathcal{PASS}(V \setminus \mathrm{var}(C))| = \prod_{v \in V \setminus \mathrm{var}(C)} |D_v|.$$

Obviously $\mathfrak{S}_V(F) \cap \mathfrak{F}_V(F) = \emptyset$ and $\mathfrak{S}_V(F) \cup \mathfrak{F}_V(F) = \mathcal{PASS}(V)$. By definition we have

$$\mathfrak{F}_V(F) = \bigcup_{C \in F} \mathfrak{F}_V(\{C\}).$$

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 $\mathfrak{F}_V(F_2) \subseteq \mathfrak{F}_V(F_1)$. 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 very injective h, replacing v by w in F using h is injective. Note that injective replacements alter the meaning 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 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.

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

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

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 [32] for a general theory of autarkies and autarky systems, while in Section 4 of [32] 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 [31], Section 3, while autarky systems for boolean clause-sets have been introduced in [33] (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 $\mathbf{Auk}(\boldsymbol{F})$; 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 $F' \leq F$, 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 $V \subseteq \mathcal{VA}$.

An **autarky reduction** is a reduction $F \to \varphi * F$ 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 [33], a special case of Lemma 3.7 in [32]), and thus the result of iterated autarky reductions until no further reductions are possible is uniquely determined; we denote it by $\mathbf{N_{Auk}}(\boldsymbol{F}) \leq F$. It is $\mathbf{N_a} := \mathrm{N_{Auk}}$ a "kernel operator", that is, $\mathrm{N_a}(F) \leq F$, $\mathrm{N_a}(\mathrm{N_a}(F)) = \mathrm{N_a}(F)$, and $F_1 \leq F_2 \Rightarrow \mathrm{N_a}(F_1) \leq \mathrm{N_a}(F_2)$; 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 $\mathrm{N_a}(F) \leq \varphi * F$). 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 $F' \leq F$ 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 $F_1, F_2 \leq F$ 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

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

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 [32], 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 [32] 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 **$\bot$-invariant**, if always $\mathcal{A}(F) = \mathcal{A}(F + \{\bot\})$ holds (in [32, 33] 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, $\bot$-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 [32, 33] "normal autarky systems" have been called "strong autarky systems", but meanwhile the above properties seem not so strong anymore to me, but quite "normal" ("ab-normality" is a defect which can be repaired; see for example Lemma 8.4 in [33], 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 [32] $\mathcal{A}$-reduction is confluent, and the result of applying $\mathcal{A}$-reductions as long as possible is uniquely determined, yielding a normal form $\mathbf{N_\mathcal{A}(}F\mathbf{)} \leq F$. As before, the operator $\mathrm{N}_\mathcal{A}$ is a kernel operator, that is, $\mathrm{N}_\mathcal{A}(F) \leq F$, $\mathrm{N}_\mathcal{A}(\mathrm{N}_\mathcal{A}(F)) = \mathrm{N}_\mathcal{A}(F)$ and $F_1 \leq F_2 \Rightarrow \mathrm{N}_\mathcal{A}(F_1) \leq \mathrm{N}_\mathcal{A}(F_2)$. 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 $\mathcal{SAT}_\mathcal{A}$, the set of all $\mathcal{A}$-lean multi-clause-sets by $\mathcal{LEAN}_\mathcal{A}$. 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.

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 $F_1, F_2 \leq F$ 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 [33]. 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* $\mathrm{var}(F) = \{v_1, \ldots, v_{n(F)}\}$.

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* $1 \leq i \leq n(F)$ *with*

$$\{v_1, \ldots, v_{i-1}\} * F \notin \mathcal{LEAN}_\mathcal{A} \quad \text{and} \quad \{v_1, \ldots, v_i\} * F \in \mathcal{LEAN}_\mathcal{A}.$$

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 $F_i := \{v_1, \ldots, v_{i-1}\} * F$ 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.

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 [37], 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" $C_1, \ldots, C_{|D_v|}$ are called **resolvable with resolution variable** v, if $\mathrm{val}_v(\{C_1, \ldots, C_{|D_v|}\}) = D_v$ 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 $i, j \in \{1, \ldots, |D_v|\}$ 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 [37], where, translating branching trees into resolution trees, the existence of a resolution tree with at most $\mathrm{mds}(F)^{n(F)}$ many leaves for unsatisfiable generalised clause-sets F is shown. Also stated in [37] 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 [31] 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.[9] This theorem can be immediately generalised to generalised clause-sets, using exactly the proof from [31] (together with the proof transformation tools provided in [37]). In [32], 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 [31] 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 [31] 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 [32], 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).

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 $\mathbf{DP_v(F)} := \{C \in F : v \notin \mathrm{var}(C)\} \cup F_v$. From the completeness results for (generalised) resolution in [37] 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

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

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 $|D_v| = 1$, then for $D_v = \{\varepsilon\}$ reduce $F \mapsto \langle v \to \varepsilon \rangle * F$.

Elimination of "pure literals" is now better called **elimination of pure variables**: If there is $v \in \mathrm{var}(F)$ with $|\mathrm{val}_v(F)| < |D_v|$, 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 [32].

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. In our context, where we fixed the domain of each variable, thus unit-clause elimination for $\{(v, \varepsilon)\} \in F$ replaces all $C \in F$ by $C \setminus \{(v, \varepsilon)\}$ in case of $D_v = \{\varepsilon\}$, while otherwise we eliminate all clauses containing the literal (v, ε) from F, and replace variable v in the remaining occurrences by a new variable v' with $D_{v'} = D_v \setminus \{\varepsilon\} \neq \emptyset$ (using any value transfer which is injective on D_v — this will ensure that the replacement in injective (recall Subsection 3.3)). The effect of unit-clause elimination in the boolean case is obtained when combining this generalised form of unit-clause elimination with trivial domain reduction. Repeated unit-clause elimination (aka unit-clause *propagation*) in the boolean case is confluent when combined with subsumption elimination in case the empty clause is created; now the generalised form of unit-clause elimination combined with trivial-domain reduction is confluent *modulo renaming* (again using subsumption elimination if the empty clause is created by trivial-domain reduction); generalising the well-known linear time algorithm for unit-clause propagation in the boolean case, this normal form can be computed in linear time.

Considering clauses $C \in \mathcal{CL}$ as constraints of scope $\mathrm{var}(C)$ (see [12], Subsection 2.1.1), and thus clause-sets $F \in \mathcal{CLS}$ as *constraint networks* (or *constraint satisfaction problems*), F is arc-consistent ([12], Definition 3.6) iff for all $C \in F \setminus \{\bot\}$

we have $|C| \geq 2$, while F is relational arc-consistent ([12], Definition 8.1) iff for all $v \in \mathrm{var}(F)$ we have $|D_v| \geq 2$.

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

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

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). 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), or v is a non-degenerated DP-variable with $c(\mathrm{DP}_v(F)) = c(F)$; in any case $\mathrm{DP}_v(F)$ is the result of applying trivial domain reduction to F. 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 elimination of pure variables 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 which is not pure for F 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 $\#_{(v,\varepsilon')}(F) \leq 1$. 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 $|D_v| \neq 1$. If a singular DP-variable v is non-degenerated then we have $c(\mathrm{DP}_v(F)) = c(F) - |D_v| + 1$. If v is a degenerated singular DP-variable, 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 3.3 *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 $\mathrm{DP}_v(F)$. ∎

In the boolean case, such applications of DP-reduction are used at many places in the literature (in [28], Appendix B, the application of DP_v for non-degenerated singular DP-variables v is called "$(1,\infty)$-reduction" (the boolean case)). We conclude by another reduction arising from the DP-operator. The notion of **blocked clauses** for boolean clause-sets (see [29, 30]) 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 [34, 35] and continued with [21, 36]. We generalise here only a very few simple notions used later in this article.

The **conflict graph cg(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 hd(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 ***r*-uniform 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-uniform hitting for $r \geq 1$, then F is hitting), while a **uniform hitting clause-set** is an r-uniform hitting clause-set for some $r \geq 0$, and we denote the set of uniform hitting clause-sets by $\mathcal{UHIT}$.

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 $\mathcal{MHIT}$ 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 hitting iff F is $c(F)$-multihitting). For a multihitting clause-set F the **multihitting number mh(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 [32], 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 4.4, 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 Section 4.1 is to generalise the notion of deficiency $\delta(F)$, which has been introduced for boolean clause-sets F in [19] 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. 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 Section 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 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.[10)] As an application we obtain in Corollary 4.9 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 4.19 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

[10)] This additional property is also new for the boolean case; it is implicitly contained in the proofs from [16], which are not only generalised here, but also simplified in such a way, that the construction becomes more lucid.

deficiency of F (actually, this condition characterises matching leanness, as shown in Lemma 4.16), 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 4.18), 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).

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 procedure in this article (regarding autarkies, all autarky systems studied here are stable for unused values).

4.1 Matching satisfiable generalised clause-sets

We whish to generalise the notion of "matching satisfiable clause-sets", introduced in [33] 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 $i \in \{1, \ldots, m\}$ there are variables $v_i \in \mathrm{var}(F_i)$ such that for all $C \in F_i$ we have $v_i \in \mathrm{var}(C)$;

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

(iii) for all $i \in \{1, \ldots, m\}$ we have $|D_{v_i}| > |\mathrm{val}_{v_i}(F_i)|$.

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 $\langle v_i \to \varepsilon_i : i \in \{1, \ldots, m\}\rangle$ 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 $\mathrm{var}(\varphi) = \{v_1, \ldots, v_m\}$, and set F_i for $i \in \{1, \ldots, m\}$ 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 $i \in \{1, \ldots, m\}$ we have $|D_{v_i}| > c(F_i)$

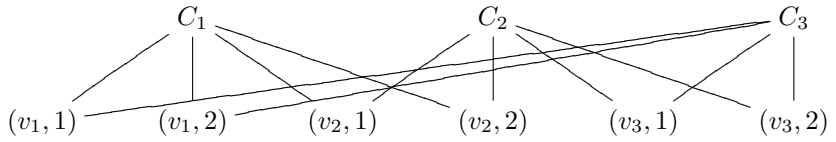
strengthens condition (iii). We call multi-clause-sets $F \in \mathcal{MCLS}$ having a decomposition $F = F_1 + \cdots + F_m$ fulfilling conditions (i), (ii) and (iii)' **matching satisfiable**, and the set of all matching satisfiable (generalised) multi-clause-sets is denoted by $\boldsymbol{\mathcal{MSAT}}$.

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

- Let
$$\begin{array}{rcl} \underline{F} & := & \{\, (C,i) : C \in F,\, i \in \{1,\ldots,F(C)\} \,\} \\ \underline{V} & := & \{\, (v,j) : v \in \mathrm{var}(F),\, j \in \{1,\ldots,|D_v|-1\} \,\}. \end{array}$$
- the vertex set of $B(F)$ is defined as $V(B(F)) := \underline{F} \uplus \underline{V}$; (the elements of $\underline{F}$ are called the *clause-nodes*, while the elements of $\underline{V}$ are called the *variable-nodes*)
- the edge set $E(B(F))$ is the set of all (undirected) edges $\{(C,i),(v,j)\}$ over $V(B(F))$ such that $v \in \mathrm{var}(C)$.

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 $v \in \mathrm{var}(F)$, while edges connect copies of variables v with copies of clauses C such that $v \in \mathrm{var}(C)$. The canonical bipartition of $B(F)$ is $(\underline{F}, \underline{V})$. 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 $\mathrm{var}(C) \cap V \neq \emptyset$).

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

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

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

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 $\delta^*(F) \leq c(F)$. 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 [33]):

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 [49], 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 [26] 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 [46]), 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 or not is decidable in polynomial time. And due to $c(F') = c(F) - \delta^*(F)$ the maximal deficiency $\delta^*(F)$ is computable in polynomial time.*

4.2 Satisfying assignments versus matching satisfying assignments

Consider a (generalised) multi-clause-set $F \in \mathcal{MCLS}$ and a partial assignment $\varphi \in \mathcal{PASS}$. The partial subgraph $B_\varphi(F)$ of $B(F)$ is obtained from $B(F)$ by eliminating all edges $\{(C,i),(v,j)\}$ such that for the literal $(v,\varepsilon) \in C$ we have $\varphi((v,\varepsilon)) = 0$ (i.e., $\varphi(v) = \varepsilon$). Now φ is called a **matching-satisfying assignment** for F if $B_\varphi(F)$ contains a matching covering all clause-vertices (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. The following two lemmas give simple basic properties regarding this notion.

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 $n(\varphi') \leq c(F)$ such that also φ' is satisfying for F.*
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.*
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.*

Proof: Assertions 1 and 2 follow by definition, while assertion 3 follows by Hall's criterion. ∎

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 $\psi * F$.*

The main result of this subsection is the following theorem.

Theorem 4.7 *For a satisfiable generalised multi-clause-set $F \in \mathcal{MCLS}$ there exists a satisfying assignment φ for F and a sub-multi-clause-set $F' \leq F$ with $c(F') = c(F) - \delta^*(F)$, such that φ is matching-satisfying for F'.*

We obtain the following generalisation of Theorem 7.16 in [33]:

Corollary 4.8 *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 Theorem 4.7 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 it is φ_0 matching-satisfying for $\varphi * F'$, where $\varphi * F = \varphi * (F' + F'') = \varphi * F' + \varphi * F'' = \varphi * F'$, and thus φ_0 is matching-satisfying for $\varphi * F$. ∎

Corollary 4.9 *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).*

In [48] 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 and simplifying the results on "admissible matchings" in [16]). A *maximal matching* in a graph is one which can not be extended, while a *maximum matching* is a matching of maximal size. The vertices covered by a matching M are the vertices incident to one of the edges in M. We begin with two auxiliary lemmas, using the following notions: Consider partial assignments φ, φ' with $\mathrm{var}(\varphi), \mathrm{var}(\varphi') \supseteq \mathrm{var}(F)$; we call φ' a *good neighbour* of φ w.r.t. F, if there is exactly one variable $v \in \mathrm{var}(F)$ with $\varphi(v) \neq \varphi'(v)$, and every clause of F falsified by φ' is also falsified by φ.

Lemma 4.10 *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 $\{v, C\} \in E(B_\varphi(F))$ such that neither v nor C is covered by M. We assume furthermore, that if there exists a matching $M^* \supset M$ in $B_\varphi(F)$, then M^* does not cover v. Then there exists a good neighbour φ' of φ w.r.t. F, such that $M' := M \cup \{\{v, C\}\}$ is a matching in $B_{\varphi'}(F)$.*

Proof: Let v_0 be the underlying variable of variable-node v, and C_0 the underlying clause of clause-node C. Let E be the set of values $\varepsilon \in D_{v_0}$, such that there is an edge $\{(v_0, i), (A, j)\} \in M$ with $(v_0, \varepsilon) \in A$. Since v is not covered by M, we have $|E| \leq |D_{v_0}| - 2$, and thus there are $\varepsilon_1, \varepsilon_2 \in D_{v_0} \setminus E$, $\varepsilon_1 \neq \varepsilon_2$. W.l.o.g. we can assume $\varphi(v_0) = \varepsilon_1$ and $(v_0, \varepsilon_1) \in C_0$ (otherwise M' would be a matching in $B_\varphi(F)$ covering v). Set $\varphi' := \varphi \circ \langle v_0 \to \varepsilon_2 \rangle$. By definition it is M' a matching in $B_{\varphi'}(F)$. Now consider a clause $A \in F$ falsified by φ', and assume that A is not falsified by φ. Thus $(v_0, \varepsilon_2) \in A$, and the literal (v_0, ε_2) is the only literal in A satisfied by φ. So no clause-node covered by M has clause A associated with it. It follows that $M^* := M \cup \{\{v, (A, 1)\}\}$ is a matching in $B_\varphi(F)$ extending M and covering v, contradicting the assumption. ∎

Lemma 4.11 *Consider a generalised multi-clause-set $F \in \mathcal{MCLS}$, a partial assignment $\varphi \in \mathcal{PASS}$, a maximal matching M' in $B_\varphi(F)$ and an edge $\{v, C\} \in E(B_\varphi(F))$ such that v is not covered by M', while C is covered by M' and thus there is a (unique) edge $\{C, w\} \in M'$. Then there exists a partial assignment φ', such that either $\varphi' = \varphi$ or φ' is a good neighbour φ' of φ w.r.t. F, and such that $M'' := (M' \cup \{\{v, C\}\}) \setminus \{\{C, w\}\}$ is a matching in $B_{\varphi'}(F)$ with $|M''| = |M'|$, variable-node w is not covered by M'', and the clause-nodes covered by M'' are exactly the clause-nodes covered by M'.*

Proof: Let v_0 be the underlying variable of variable-node v, C_0 the underlying clause of clause-node C, and let $(v_0, \varepsilon) \in C_0$. If $\varphi((v_0, \varepsilon)) = 1$, then let $\varphi' := \varphi$ and we are done. Otherwise let $M := M' \setminus \{\{C, w\}\}$ and apply Lemma 4.10. ∎

We further remind at the notion of an *M-augmenting path* for a matching M in a graph G (see Section 16.1 in [46]), which is a path of odd length with endpoints not covered by M and whose edges are alternatingly out of and in M.

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 M-augmenting path in $B(F)$ (note that here we can use all edges). Then we can construct (in polynomial time) partial assignments $\varphi_0, \ldots, \varphi_m$, $m \in \mathbb{N}_0$, and a matching M^+ in $B_{\varphi_m}(F)$ with $|M^+| = |M| + 1$, such that $\varphi_0 = \varphi$ and φ_i is a good neighbour of φ_{i-1} for $i \in \{1, \ldots, m\}$.*

Proof: In the following construction we will construct $\varphi_0, \ldots, \varphi_{m'}$ for some $m' \in \mathbb{N}_0$, fulfilling the above conditions but allowing $\varphi_i = \varphi_{i+1}$ for some i; $m \leq m'$ and the final list of partial assignments is obtained by removing identical neighbours from the list $\varphi_0, \ldots, \varphi_{m'}$. W.l.o.g. we can assume that the augmenting path P is of the form $P = (v_1, C_1, v_2, C_2, \ldots, v_k, C_k)$, $k \in \mathbb{N}$, where the v_i are (distinct) variable-nodes, and the C_i are (distinct) clause-nodes. We construct now partial assignments φ_i and matchings M_i in $B_{\varphi_i}(F)$ for $i = 0, \ldots, m'$, where $0 \leq m' < k$ is determined by the following construction:

1. Set $\varphi_0 := \varphi$ and $M_0 := M$.

2. Set $i := 1$.

3. While $i < k$ do

   (a) if matching M_{i-1} is not maximal in $B_{\varphi_{i-1}}(F)$, then extend M_{i-1} by one edge to obtain M^+, and with $m' := i - 1$ we are done with the whole construction;

   (b) otherwise apply Lemma 4.11 with $M' := M_{i-1}$, $\varphi := \varphi_{i-1}$ and $v := v_i$, $C := C_i$, and let $M_i := M''$ and $\varphi_i := \varphi'$;

   (c) set $i := i + 1$.

4. After completing the while-loop (we now have $i = k$), apply Lemma 4.10 with $M := M_{i-1}$, $\varphi := \varphi_{i-1}$ and $v := v_i$, $C := C_i$, and we set $m' := i$, $\varphi_i := \varphi'$ and $M^+ := M'$. ∎

Using the fact, that if a matching in a graph is not maximum, then it has an augmenting path (see Theorem 16.1 in [46]), we prove Theorem 4.7 as follows: Start with any satisfying assignment φ_0 for F and the empty matching M_0. Apply Lemma 4.12 repeatedly until a maximum matching M^+ in $B(F)$ is obtained, and set $\varphi := \varphi_m$.

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 [33] we get

Lemma 4.13 *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 $F' \leq F$ 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 4.14 *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 [33], we obtain the fundamental relation between tight sub-multi-clause-sets and matching autarkies:

Lemma 4.15 *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 $\delta(\varphi * F) \geq \delta(F)$.*

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 [33], we now can characterise matching lean multi-clause-sets:

Lemma 4.16 *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. *$\forall\, F' \subsetneq F : \delta(F') < \delta(F)$;*
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 4.15, Part 3b. Obviously, Part 4 implies Part 3, and Part 3 implies Part 2. Finally, Part 1 follows from Part 2 by Lemma 4.15, Part 2. ∎

By Lemma 4.16, Part 2 we get

Corollary 4.17 *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 4.18 *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 4.16, Part 4 together with Lemma 4.15, Part 3b we get

Corollary 4.19 *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 4.16, Part 3 we get the following generalisation of "Tarsi's Lemma" (see [1]):

Corollary 4.20 *If the generalised multi-clause-set $F \neq \top$ is matching lean, then $\delta(F) \geq 1$.*

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

Corollary 4.21 *If a generalised clause-set $F \in \mathcal{CLS}$ is minimally unsatisfiable, then we have $\delta(F) \geq 1$.*

In [10], 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 4.21.

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 4.22 $\mathcal{MUSAT}_{\delta=1} = \mathcal{MLEAN}_{\delta=1} \cap \mathcal{USAT}$.

4.4 Comparison with an earlier version of "matching autarkies"

In [32] 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 $F_{\mathrm{var}(\varphi)}$.

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

(i)' for $i \in \{1, \ldots, m\}$ there are variables $v_i \in \mathrm{var}(F_i)$ such that for all clause-occurrence C in F_i there are literals $x_C \in C$ with $\mathrm{var}(x_C) = v_i$, 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 [45]), 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 [45] has been coined the "multivalued encoding", which is the "standard translation", but without AMO ("at most one") clauses (since they would destroy the combinatorial structure): For every literal (v, ε) we consider a boolean variable $\tau((v, \varepsilon))$ expressing that v shall not get value ε, and 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 could be chosen).[11)]

In [32] 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 [32] only a restricted notion of "matching autarkies" has been used (recall Subsection 4.4) 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 [20] (using the notion of "nb-formulas" (for "non-boolean")).

It is worth to mention here, that in [44] it has been shown, that resolution which works only with generalised clause-sets, that is, where in the corresponding

[11)] 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.

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 [37]) 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 (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). 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 $\mathbf{ALO_v} := \{\overline{\tau((v,\varepsilon))} : \varepsilon \in D_v\} \in \mathcal{CL}(\mathcal{VA}_{\{0,1\}})$ be the (negative, boolean) clause expressing that v gets assigned at least one of the values $\varepsilon \in D_v$, 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 [21]). If F contains no pure variables (recall Subsection 3.8), then the sub-clause-sets of $\Theta(F)$ given by $\Theta(F')$ for $F' \subseteq F$ are exactly the sub-clause-sets of $\Theta(F)$ 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 $\varphi(v) = \varepsilon$. 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 extension** 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 extension 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 $\varepsilon_0 \in D_v$ with $\varphi(\tau((v,\varepsilon_0))) = 0$, and setting $\psi(\tau((v,\varepsilon'))) := 1$ for $\varepsilon' \in D_v \setminus \{\varepsilon_0\}$, while $\psi(\tau((v,\varepsilon_0))) := 0$.

The purpose of standard extensions is, that they allow to "lift" an autarky φ for $\Theta(F)$ to an autarky ψ for $\Theta(F)$, such that ψ is a standard extension of φ — now by Lemma 5.1 we can reflect ψ back to a an autarky for F. The following lemma (with obvious proofs) states the basic properties of standard extensions.

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 extension $\psi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ 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 [32] 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 $\tau(\varphi)$ matching satisfies $\Theta(F)$, then 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 [32] rephrased in the terminology of Subsection 4.4 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 [32], 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 $\mathrm{N}_{\mathrm{ma}}(\Theta(F_1)) = \tau(F_1)$ (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 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 $\delta^*(\Theta(F_2)) = 1$.

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 $\mathrm{N}_{\mathrm{ma}}(\Theta(F_3)) = \tau(2 \cdot \{(v, 2)\})$ (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 a boolean variable w, since here

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

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 [48] 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 [48]), 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 important) notion of minimally unsatisfiable clause-sets. In this section we will 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). In the boolean case, irredundant clause-sets have been studied in [9] (using the notion of a "clause minimal formula"), concentrating on complexity theoretical questions; we consider the basic question of preservation of irredundancy under application of partial assignments.

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. Finally 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.

6.1 Irredundant clause-sets

A clause $C \in F$ is called **redundant** for clause-set $F \in \mathcal{CLS}$ if $F \setminus \{C\} \models C$ holds, while otherwise C is called **irredundant** for F. It is C redundant for F if and only if the set $\mathfrak{F}_{\mathrm{var}(F)}(C)$ of falsifying assignments for C is covered by $(\mathfrak{F}_{\mathrm{var}(F)}(C'))_{C' \in F \setminus \{C\}}$, the set of falsifying assignments for the remaining clauses. We are interested here in the question, given a partial assignment φ and a clause $C \in F$ with $\varphi * \{C\} \neq \top$, 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 $\mathfrak{F}_{\mathrm{var}(F)}(C)$ is covered

by $(\mathfrak{F}_{\mathrm{var}(F)}(C'))_{C' \in F \setminus \{C\}}$, 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 $\mathfrak{F}_{\mathrm{var}(F)}(C)$ which consists of assignments compatible with φ is covered by $(\mathfrak{F}_{\mathrm{var}(F)}(C'))_{C' \in F \setminus \{C\}}$.

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 $F \setminus \{C\} \not\models C \cup \{(v, \varepsilon)\}$.*

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. 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 $(\mathfrak{F}_{\mathrm{var}(F)}(C'))_{C' \in F \setminus \{C\}}$ (i.e., these two sets are disjoint). 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 $\mathfrak{F}_V(C) \cap \mathfrak{F}_V(C') = \emptyset$ 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) $\mathfrak{F}_{\mathrm{var}(F)}(C) \cap \bigcup_{C' \in F \setminus \{C\}} \mathfrak{F}_{\mathrm{var}(F)}(C') = \emptyset$.

(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 [35]:

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

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. An irredundant core of an unsatisfiable clause-set is called a **minimally unsatisfiable core**.

Lemma 6.8 *Consider a generalised clause-set $F \in \mathcal{CLS}$ 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 $\mathfrak{F}_V(F_1) \cap \mathfrak{F}_V(F_2) = \emptyset$.*
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 $\sum_{C \in F} |\mathfrak{F}_V(\{C\})| = |\mathcal{PASS}(V)|$.*

Proof: Part 1 follows by definition. In Part 2 it is $\mathfrak{F}_V(\{C\})$ covered by $\mathfrak{F}_V(F')$, and thus by Part 1 in fact $\mathfrak{F}_V(\{C\})$ is covered by $\mathfrak{F}_V(F' \cap F_C)$, 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.*

Using $|\mathfrak{F}_{\mathrm{var}(F)}(C)| = \prod_{v \in \mathrm{var}(F) \setminus \mathrm{var}(C)} |D_v|$ for $C \in F$ it follows that satisfiability for generalised hitting clause-sets is decidable in polynomial time (generalising the well-known special case for boolean clause-sets). For boolean *bihitting* clause-sets (where the conflict graph is bipartite) in [21] 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); 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) multi-hitting clause-sets have efficient satisfiability decision (see [36] for more information in the boolean case).

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. 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 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 6.11 *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 $\pi : F \to F^*$ 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 [28] 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 $\langle v \to \varepsilon \rangle * F$ 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 6.12 *Consider a generalised clause-set $F \in \mathcal{MUSAT}$ and a literal $(v, \varepsilon) \in \mathcal{LIT}$.*

1. *If $\langle v \to \varepsilon \rangle * F \in \mathcal{MUSAT}$, 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 $\langle v \to \varepsilon \rangle * F \in \mathcal{MUSAT}$.*

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 $\langle v \to \varepsilon\rangle * F' \in \mathcal{USAT}$ with $\langle v \to \varepsilon\rangle * F' = (\langle v \to \varepsilon\rangle * F) \setminus \{C\}$, and thus C would be redundant in $\langle v \to \varepsilon\rangle * F$.

For Part 2 assume that $\langle v \to \varepsilon\rangle * F$ 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 6.13 *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 6.13 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 $F \in \mathcal{SMUSAT}$ (as can be verified directly), while $\langle a \to 2\rangle * F = \{\bot, \{b \neq 2\}\} \notin \mathcal{MUSAT}$ (as well as $\langle b \to 2\rangle * F = \{\bot, \{a \neq 2\}\} \notin \mathcal{MUSAT}$).

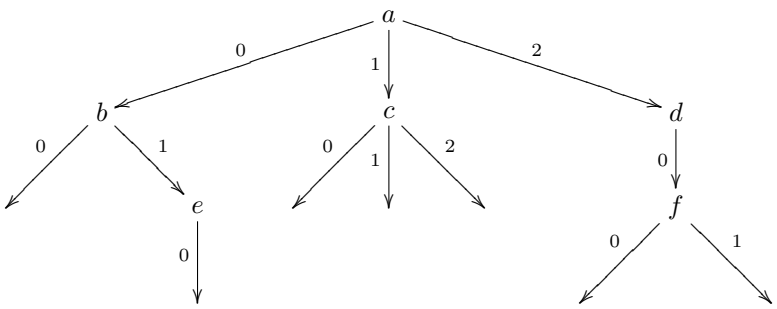
An important application of the process of saturation for *boolean* clause-sets is given by Lemma C.2 in [28], proving that for every $F \in \mathcal{MUSAT}$, $F \neq \{\bot\}$ there is a variable $v \in \mathrm{var}(F)$ such that for $\varepsilon \in D_v = \{0, 1\}$ we have $\#_{(v,\varepsilon)}(F) \leq \delta(F)$. The proof is based on the characterisation of saturated minimally unsatisfiable boolean clause-sets in Corollary 6.13 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 clear to me how to obtain a full generalisation for generalised clause-sets (the problem is that saturation is not that powerful anymore); in Lemma 6.14 we obtain the generalisation of Lemma C.2 ([28]) to generalised clause-sets in the special case of deficiency one.

6.4 Characterisation of the basic case of deficiency one

Generalising the tree construction from [28] (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.



This tree representation R uses six variables $a, \ldots, f$ 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 $w_0, e_1, w_1, \ldots, e_m, w_m$ 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 $C_w := \{(v(w_i), \varepsilon(e_{i+1})) : i \in \{0, \ldots, m-1\}\}$. 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

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

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 $C_r = \bot$ we see that $F(T, r, v, \varepsilon)$ is unsatisfiable.

2. $F(T, r, v, \varepsilon)$ is a 1-uniform 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 subtree 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 $\mathcal{UHIT}$ for the set of uniform hitting clause-sets and hd for the hitting degree, as introduced before, we have $F(T,r,v,\varepsilon) \in \mathcal{UHIT}^{\mathrm{sat}=0}_{\mathrm{hd}=1,\delta=1}$.

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 [28] 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 6.14 *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 $\Theta(F)$ has a 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). 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). ∎

For the proof of the main results, we need to generalise some well-known basic facts about DP-reduction, which are obvious by the remarks made before Lemma 3.3:

Lemma 6.15 *Consider a generalised clause-set $F \in \mathcal{CLS}$ and a singular DP-variable v w.r.t. F.*

1. $\delta(\mathrm{DP}_v(F)) \le \delta(F)$.
2. *If v is non-degenerated, then $\delta(\mathrm{DP}_v(F)) = \delta(F)$.*

Finally we are able to generalise Lemma C.5 in [28]:

Theorem 6.16 *The class $\mathcal{MUSAT}_{\delta=1}$ of minimally unsatisfiable (generalised) clause-sets of deficiency 1 is exactly 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: 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 6.14 there exists a variable $v \in \mathrm{var}(F)$ such that for all $\varepsilon \in D_v$ we have $\#_{(v,\varepsilon)}(F) = 1$. Thus v is a singular DP-variable w.r.t. F. Let $G := \mathrm{DP}_v(F)$. By Lemma 3.3 and Lemma 6.15 it is $G \in \mathcal{MUSAT}_{\delta=1}$, and we can apply the induction hypothesis to G; the assertion follows now immediately by extending the tree representation of G at the (single) new resolvent (obtained by the DP-reduction) by adding leaves corresponding to the occurrences of v in F. ∎

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

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).

7 Linear autarkies and hypergraph inequalities

The main result of this section is Theorem 7.8, which covers two applications of linear algebra in hypergraph theory, namely the (generalised) Fisher inequality (in design theory; see Lemma 7.10) and the Seymour inequality (for minimally non-2-colourable hypergraphs; see Corollary 8.2). "Balanced linear autarkies" provide here the link, similar to the role played by matching autarkies for establishing minimal deficiency one for minimally unsatisfiable clause-sets; the core result here is Lemma 7.2.

7.1 Linear and balanced linear autarkies

The notion of a "linear autarky", introduced for boolean clause-sets in [31] and further investigated in [50, 33], has the following natural generalisation to generalised clause-sets: A partial assignment $\varphi \in \mathcal{PASS}$ is called a **linear autarky** for $F \in \mathcal{CLS}$ if there is a weight function $w : \mathrm{var}(F) \to \mathbb{Q}_{>0}$, assigning to each variable in F a positive rational number, such that for all clauses $C \in F$ we have

$$\sum_{x \in C, \varphi(x)=1} w(\mathrm{var}(x)) \geq \sum_{x \in C, \varphi(x)=0} w(\mathrm{var}(x)), \tag{2}$$

that is, if for every clause the sum of the weights of the underlying variables of literals satisfied by φ is not smaller than that sum over the literals falsified by φ.[12)] Obviously, every linear autarky for F is an autarky for F (if a clause contains a falsified literal, then by (2) it must also contain a satisfied literal). Choosing a constant weight function we see that every satisfiable generalised clause-set $F \in 2\text{–}\mathcal{CLS}$ is satisfiable by a linear autarky, and thus, since for example graph colouring can be expressed with clause-sets from $2\text{–}\mathcal{CLS}$ (see Subsection 8.1), we see that deciding the existence of a linear autarky for generalised clause-sets is NP-hard even for inputs from $2\text{–}\mathcal{CLS}$ (while in the boolean case the (general) decision problem is solvable in polynomial time as shown in [31]).

Linear autarkies fulfil all conditions for a normal autarky system except that the iteration condition is not fulfilled (recall Subsection 3.6; the problem with the

[12)] Literals with underlying variables not in the domain of φ are not taken into account.

iteration condition is that if φ is a linear autarky for F, and ψ a linear autarky for $\varphi * F$, then ψ might behave irresponsible on the clauses already satisfied by φ, and thus $\psi \circ \varphi$ might not be a linear autarky for F). In this article we do not further investigate the notion of linear autarkies for generalised clause-sets (see [32] for some hints on further developments), but we take a closer look at "balanced linear autarkies".

As introduced in [33] (Section 6), a **balanced linear autarky** for a generalised clause-set $F \in \mathcal{CLS}$ is a linear autarky for F where in (2) for all clauses equality holds. Again, balanced linear autarky yield an autarky system which fulfils all normality conditions except of the iteration condition.

In the boolean case, the existence problem for balanced linear autarkies can be decided in polynomial time. The existence problem for linear autarkies for (generalised) clause-sets was above shown to be NP-hard even for inputs from 2–$\mathcal{CLS}$, and so let us examine balanced linear autarkies for $F \in 2\text{–}\mathcal{CLS}$. For such F by definition balanced linear autarkies are the partial assignments which in every clause of F they touch satisfy one literal and falsify the other literal (in general, balanced linear autarkies cannot touch unit clauses). It follows that balanced linear autarkies on 2–$\mathcal{CLS}$ in fact yield a normal autarky system, and satisfiability by balanced linear autarkies generalise the NOT-ALL-EQUAL-2-SAT problem from boolean clause-sets to generalised clause-sets. Satisfiability of generalised 2-clause-sets by balanced linear autarkies is in NP due to this characterisation, and the guess is that it is also NP-complete, though at present we do not have a proof (it is well known that the boolean case is poly-time decidable, and this follows also from the efficient procedure for finding balanced linear autarkies for boolean clause-sets mentioned above).

As remarked at the end of Section 6 in [33], we have the following fundamental property of balanced linear autarkies for boolean clause-sets:

Lemma 7.1 *A boolean clause-set F has no non-trivial balanced linear autarky (i.e., F is **lean w.r.t. balanced linear autarkies**) iff the columns of the clause-variable matrix $M(F)$ of F are linearly independent (where the clause-variable matrix $M(F)$ of a multi-clause-set F is a $c(F) \times n(F)$ matrix over $\{-1, 0, +1\}$, containing the natural encoding of the clauses in the rows).*

It follows that if F is balanced-linearly lean, then $\delta(F) \geq 0$. We will later see, that this is the heart of the statement proven in [47], that a minimally non-2-colourable hypergraph without non-covered vertices has as least as many edges as vertices. In Theorem 2 of [1] this statement was strengthened to the statement, that in a minimally non-2-colourable hypergraph G without non-covered vertices there exists a matching in the bipartite graph $B(G)$ corresponding to G (the vertices and the hyperedges of G constitute the bipartition of $B(G)$, and a vertex node v is joined with a hyperedge node H if $v \in H$) covering all vertices. We will see that the following lemma generalises this strengthening.

Lemma 7.2 *Consider a boolean clause-set $F \in \mathcal{CLS}(\mathcal{VA}_{\{0,1\}})$ which is balanced linearly lean. Then $\delta^*(F) = \delta(F)$ holds, i.e., there is a matching in $B(F)$ covering all variable nodes. (Since $\delta^*(F) \geq 0$, this statement includes $\delta(F) \geq 0$.)*

Proof: Assume $\delta^*(F) > \delta(F)$, and consider a tight F' for F. By Lemma 4.15, Part 3b there is a matching autarky φ for F with $\varphi * F = F'$. Let $V := \mathrm{var}(\varphi)$. Since $\delta(F') > \delta(F)$, by Lemma 4.15, Part 1 we get $\delta(F[V]) < 0$. In general a multi-clause-set $G \in \mathcal{MCLS}$ has a non-trivial balanced linear autarky iff $M(G) \cdot \vec{x} = 0$

has a non-trivial solution (over the rational or over the real numbers), and this is guaranteed if this system of linear equations is under-constrained, that is, if $\delta(G) < 0$ holds. Thus $F[V]$ has a non-trivial balanced linear autarky, which yields a (non-trivial) balanced linear autarky for F. ∎

For balanced-linearly lean boolean clause-sets F we thus know $\forall\, F' \subseteq F : \delta(F') \leq \delta(F)$, which is weaker than $\forall\, F' \subset F : \delta(F') < \delta(F)$, shown in Lemma 4.16 to be equivalent to F being matching lean. But the notion of balanced linear autarkies is incomparable to the notion of matching autarkies, as the following examples show.

1. Consider three (different) boolean variables $x, y, z \in \mathcal{VA}$ and a clause-set F containing 6 (different) clauses C with $\mathrm{var}(C) = \{x, y, z\}$ (and nothing else). Consider (different) boolean variables $a, b \in \mathcal{VA} \setminus \{x, y, z\}$, and add the literals a, b to all 6 clauses in F. The partial assignment $\langle a \to 1, b \to 0\rangle$ is a balanced linear autarky for F (choose some constant weighting) satisfying F, while F is matching lean, since $\delta(F) = 6 - 5 = 1$, while for $\top \neq F' \subset F$ we have $n(F') = 5$ and $c(F') \leq 5$, and thus $\delta(F') \leq 0$.
2. The most trivial example for a matching satisfiable boolean clause-set which is balanced-linearly lean is $\{\{a\}\}$, while an example not containing unit-clauses is $\{\{a, b\}, \{a, \overline{b}\}\}$.

A typical criterion to establish that a boolean clause-set is balanced linearly lean is given by the following lemma (which follows from elementary linear algebra together with Lemma 7.1).

Lemma 7.3 *Consider a boolean clause-set $F \in \mathcal{CLS}(\mathcal{VA}_{\{0,1\}})$. If $M(F)^t \cdot M(F)$ is non-singular, then F is balanced linearly lean.*

Besides Lemma 7.2, our use of balanced linear autarkies is based on the following lemma (which follows directly from the definitions), allowing to conclude from a boolean clause-set $F \cup \{\overline{C} : C \in F\}$ being linearly lean, that F is balanced linearly lean (suitably generalised in the next subsection). For a boolean literal (x, ε) we use $\overline{(v, \varepsilon)} := (v, 1 - \varepsilon)$, while for a boolean clause C we use $\overline{C} := \{\overline{x} : x \in C\}$, and finally for a boolean clause-set F we set $\overline{F} := \{\overline{C} : C \in F\}$.

Lemma 7.4 *Consider a boolean clause-set $F \in \mathcal{CLS}(\mathcal{VA}_{\{0,1\}})$. A partial assignment $\varphi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$ is a balanced linear autarky for F if and only if φ is a linear autarky for $F \cup \overline{F}$. Thus F is balanced linearly lean if and only if $F \cup \overline{F}$ is linearly lean.*

7.2 Preliminaries on hypergraphs

A (finite) *hypergraph* G is a pair $G = (V, E)$, where the finite set $V(G) = V$ is the set of "vertices" of G, while $E(G) = E \subseteq \mathbb{P}(V)$ is the set of "hyperedges" of G (often also just called "edges"). Every graph is a hypergraph. An *non-covered vertex* v of a hypergraph G is an element of $V(G) \setminus \bigcup E(G)$. A hypergraph G is *simple* if there are no $H, H' \in E(G)$ with $H \subset H'$ (i.e., there are no subsumed hyperedges).

The underlying **variable hypergraph** of a multi-clause-set F has vertex set $\mathrm{var}(F)$, while the hyperedges are the sets $\mathrm{var}(C)$ for $C \in F$.

A C-*colouring* of a hypergraph G for some "colour-set" C is a map $f : V(G) \to C$ such that for all hyperedges $e \in E(G)$ there exist vertices $v, w \in e$ with $f(v) \neq f(w)$; a k-*colouring* for $k \in \mathbb{N}_0$ is a $\{1, \dots, k\}$-colouring. A hypergraph G containing a hyperedge $e \in E(G)$ with $|e| \leq 1$ has no C-colouring for any C, and thus is called *uncolourable*, while otherwise the identity yields a $V(G)$-colouring for every hypergraph G, and G is called *colourable*.[13)] The *chromatic number* $\chi(G)$ of a colourable hypergraph G is the minimal cardinality of a set C such that a C-colouring of G exists. We have $0 \leq \chi(G) \leq |V(G)|$, where $\chi(G) = 0$ iff $V(G) = E(G) = \emptyset$ and $\chi(G) = 1$ iff $V(G) \neq \emptyset$ and $E(G) = \emptyset$. A (colourable) hypergraph G is called k-*chromatic critical* (or *critical* k-*colourable*) for $k \in \mathbb{N}_0$ if $\chi(G) = k$, and for every edge $e \in E(G)$ we have $\chi((V(G), E(G) \setminus \{e\})) < k$; note that if $\chi(G) \leq 1$, then G is automatically critically $\chi(G)$-colourable, since there are no edges. A hypergraph G is *minimally non-*k-*colourable* for $k \in \mathbb{Z}_{\geq -1}$ if G is not k-colourable, but for all edges $e \in E(G)$ the hypergraph $(V(G), E(G) \setminus \{e\})$ is k-colourable; if G is $(k+1)$-chromatic critical, then G is also minimally non-k-colourable, while the converse holds exactly for the colourable hypergraphs.[14)]

An *independent set* in a hypergraph G is a subset $S \subseteq V(G)$ such that there is no hyperedge $H \in E(G)$ with $H \subseteq S$, while a *transversal* of G is a subset $T \subseteq V(G)$ such that for all hyperedge $H \in E(G)$ we have $T \cap H \neq \emptyset$. It is S an independent subset of G iff $V(G) \setminus S$ is a transversal of G. A hypergraph G is 2-colourable iff G has an independent transversal.

A (finite) *general hypergraph* is a triple $(V, E, \mathcal{E})$, where E, V are finite sets and $\mathcal{E} = \mathcal{E}(G) : E \to \mathbb{P}(V)$ assigns to each hyperedge $H \in E(G)$ the vertex set $\mathcal{E}(H) \subseteq V$. A hypergraph G can be implicitly promoted to a general hypergraph $(V(G), E(G), \mathcal{E})$ with $\mathcal{E}(H) := H$ for $H \in E(G)$, while the underlying hypergraph of a general hypergraph G is $(V(G), \{\mathcal{E}(H) : H \in E(G)\})$, and this conversion is applied only if necessary. The *dual* G^{t} of a general hypergraph is given by $V(G^{\mathrm{t}}) = E(G)$ and $E(G^{\mathrm{t}}) = V(G)$, while to each hyperedge in G^{t}, which is a vertex of G, the set of incident hyperedges of G (vertices of G^{t}) is assigned. A general hypergraph G is called *square* if $|E(G)| = |V(G)|$. A general hypergraph G is called *intersecting* if for all $H, H' \in E(G)$ we have $\mathcal{E}(H) \cap \mathcal{E}(H') \neq \emptyset$ (thus a hypergraph without hyperedges is intersecting, while in the presence of some hyperedge an intersecting general hypergraph cannot contain an empty hyperedge).

A *pairwise balanced design with index* $\lambda \in \mathbb{N}_0$ is a general hypergraph G such that for every two vertices $v, w \in V(G)$, $v \neq w$, there are exactly λ hyperedges $H \in E(G)$ with $v, w \in \mathcal{E}(H)$; in the context of design theory vertices are called "points" and hyperedges "blocks", and points are "incident" with blocks if they are contained in the associated vertex set, and then this condition states that every two points are incident with exactly λ blocks. A *pairwise balanced design* is a pairwise balanced design with some index (in $\mathbb{N}_0$). We use general hypergraphs here to

[13)] Typically, in hypergraph theory colourings ignore hyperedges of size at most one (see for example Chapter 7 in [24]), which unnecessarily breaks the natural connection to the theory of generalised clause-sets. It seems better to me to depart from this tradition. In this way we are also consistent with the notion of colourings for graphs: If a (general) graph has a loop, then it has no colouring at all, and in the same spirit, if a hypergraph contains the empty edge or an edge of size one, then it has no colouring at all. When considering the canonical translation of hypergraph colouring problems into satisfiability problems for generalised clause-sets, then hypergraphs with hyperedges of size one or zero translate into clause-sets which are ("automatically") unsatisfiable.

[14)] If G is uncolourable, then G is minimally non-k-colourable for some k iff $|E(G)| = 1$, where in case of $V(G) = \emptyset$ (and thus $E(G) = \{\emptyset\}$) it is G minimally non-k-colourable for all $k \geq -1$, while otherwise (that is, $V(G) \neq \emptyset$ and $E(G) = \{\{v\}\}$ for some $v \in V(G)$) G is minimally non-k-colourable for all $k \geq 0$.

allow blocks with the same point set. We call a pairwise balanced design G *non-degenerated*, if G has at least two points (thus we have at least λ blocks in G), and every block contains at least two points. A *trivial pairwise balanced design* with index $\lambda \in \mathbb{N}_0$ is a general hypergraph G with $|E(G)| = \lambda$, while for all $H \in E(G)$ we have $\mathcal{E}(H) = V(G)$ (i.e., G consists just of λ full blocks; a non-degenerated pairwise balanced designs is trivial iff it has the minimal number of blocks).

A (finite) *linear design* is a pairwise balanced design with index 1. Blocks of linear designs are also called "lines". A *dual linear design* is the dual of a linear design. A (finite) *projective incidence plane* is a linear design G which is also a dual linear design, and such that there exist four (different) points in G so that for any three of them there exists no line incident with all of them. A linear design which is also a dual linear design but which is not a projective incidence plane is called a *degenerated projective incidence plane*. If G is a projective incidence plane, then we have

1. G is an intersecting hypergraph (without repeated blocks) with at least 7 points and 7 lines;
2. there is $n \in \mathbb{N}$, $n \geq 2$, the *order of* G, such that each line of G is incident with (exactly) $n+1$ points, and each point of G is incident with (exactly) $n+1$ lines (in other words, in G as well as in G^{t} every hyperedge has length $n+1$);
3. $|V(G)| = |E(G)| = (n+1)^2 - n = n^2 + n + 1$, and thus G is square.

We do not make (full) use of the following facts (and non-facts) for projective incidence planes, but they seem instructive to me (especially since the open existence problems might be approachable (in the future) for SAT solvers): For p a prime number and $e \in \mathbb{N}$ it is known that there are projective incidence planes $\mathrm{PG}(2, p^e)$ of order p^e given by the usual construction of projective geometry: Let F be a field of order p^e (thus F is necessarily commutative), let the points of $\mathrm{PG}(2, p^e)$ be the 1-dimensional linear subspaces of the F-vector space F^3, let the lines be the 2-dimensional linear subspaces of the F^3, and let a point of $\mathrm{PG}(2, p^e)$ be incident with a line if the point is a subset of the line (as subsets of F^3). Projective incidence planes isomorphic to some $\mathrm{PG}(2, p^e)$ are called *desarguesian*. It is known that for $p^e \geq 9$ and $e \geq 2$ always non-desarguesian projective incidence planes of order p^e exist, while for $n \leq 8$ every projective incidence plane of order n is desarguesian (see Section 3.2 in [13]). It is open whether there are non-desarguesian projective incidence planes of order $p \geq 11$, and it is open whether there are projective incidence planes of order $n \geq 12$ not a prime power (i.e., with two different prime divisors).

7.3 Applications to hypergraph inequalities

We make use of the following connection between vertices and variables: All hypergraphs G fulfil $V(G) \subseteq \mathcal{U}$ for some fixed universe $\mathcal{U}$, and for every domain D there is a bijection $\mathrm{var}_D : \mathcal{U} \to \mathcal{VA}_D$, i.e., for a fixed domain D there is a one-to-one correspondence between "vertices" (at all) and variables with domain D; in this way we have an easy way of translation between vertices and variables (with a given domain), and also an easy way of changing the domain of variables. Now back to the first problem of generalising Lemma 7.4.

Since for generalised clause-sets we do not have complementation (negation), in order to generalise Lemma 7.4, we need to fix in some sense the signs of the

variables. Since a sign-less clause-set is nothing else than a hypergraph, this is most easily established by considering a hypergraph G, a domain D and a sub-domain $\emptyset \neq D' \subseteq D$, and defining the clause-set $\boldsymbol{F_{[D',D]}(G)} \in \mathcal{CLS}(\mathcal{VA}_D)$ by

$$F_{[D',D]}(G) := \bigcup_{\varepsilon \in D'} \{\mathrm{var}_D(H) \times \{\varepsilon\} : H \in E(G)\}.$$

In words, $F_{[D',D]}(G)$ is obtained from G by collecting for each edge $H \in E(G)$ and each value $\varepsilon \in D'$ the clauses $\{(\mathrm{var}_D(v), \varepsilon) : v \in H\}$ (recall that $\mathrm{var}_D(v)$ is the variable with domain D corresponding to vertex v). We have $n(F_{[D',D]}(G)) = |V(G)|$ and $\mathrm{wn}(F_{[D',D]}(G)) = (|D|-1) \cdot |V(G)|$. The variable hypergraph of $F_{[D',D]}(G))$ is G. Parameter D' is used for technical purposes, so that we can express

$$F_{[D',D]}(G) = \bigcup_{\varepsilon \in D'} F_{[\{\varepsilon\},D]}(G). \tag{3}$$

The most important case is $D' = D$; let $\boldsymbol{F_{[D]}(G)} := F_{[D,D]}(G)$. The union in (3) is disjoint if $\emptyset \notin E(G)$. The clause-set $F_{[D',D]}(G)$ depends only on the edge set of G, ignoring non-covered vertices in G. It is $F_{[D]}(G)$ multihitting iff G is intersecting, where if an intersecting G has at least one edge, then $F_{[D]}(G)$ is $|D|$-multihitting, while for G without an edge $F_{[D]}(G)$ is 1-multihitting if G has vertices, and 0-multihitting otherwise.

For the ease of reference we collect some obvious properties related to (balanced) linear autarkies for 2-colourability problems in the following lemma.

Lemma 7.5 *Consider a hypergraph G.*

1. *For $\varepsilon \in \{0,1\}$ we have*
$$\overline{F_{[\{\varepsilon\},\{0,1\}]}(G)} = F_{[\{1-\varepsilon\},\{0,1\}]}(G).$$
2. *Thus $F_{[\{0,1\}]}(G)$ is linearly lean iff $F_{[\{0\},\{0,1\}]}(G)$ is balanced linearly lean (by Lemma 7.4).*
3. *A partial assignment $\varphi \in \mathcal{PASS}(\mathcal{VA}_{\{0,1\}})$, considered here as a partial map assigning "colours" $0,1$ to (some) vertices of G, is a balanced linear autarky for $F_{[\{0\},\{0,1\}]}(G)$ if and only if for every hyperedge H of G the number of vertices assigned colour 0 equals the number of vertices assigned colour 1.*
4. *More generally, for every domain D and $\varepsilon \in D$ we have, that $\varphi \in \mathcal{PASS}(\mathcal{VA}_D)$ is a balanced linear autarky for $F_{[\{\varepsilon\},D]}(G)$ if and only if for every hyperedge H of G the number of vertices assigned colour ε equals the number of vertices assigned a colour from $D \setminus \{\varepsilon\}$.*
5. *Thus $F_{[\{0\},\{0,1\}]}(G)$ is balanced linearly lean if and only if $F_{[\{\varepsilon\},D]}(G)$ for any domain D with at least two elements and for any $\varepsilon \in D$ is balanced linearly lean.*[15)]

We remark that $B(F_{[\{0\},\{0,1\}]}(G))$ is isomorphic to $B(G)$ via the bijection mapping vertex nodes resp. hyperedge nodes of $B(G)$ to variable nodes resp. clause nodes of $B(F_{[\{0\},\{0,1\}]}(G))$.

[15)] Actually, for every autarky system $\mathcal{A}$ which is invariant under renaming and stable for unused values we have the same equivalence, that is, $F_{[\{0\},\{0,1\}]}(G)$ is $\mathcal{A}$-lean iff $F_{[\{\varepsilon\},D]}(G)$ is $\mathcal{A}$-lean.

In general, if $\emptyset \notin E(G)$ then $c(F_{[D',D]}(G)) = |D'| \cdot |E(G)|$, and thus

$$\delta(F_{[D]}(G)) = |D| \cdot (|E(G)| - |V(G)|) + |V(G)|. \tag{4}$$

A generalised clause-set $F \in \mathcal{CLS}$ is called **colouring** (see Section 8 for the motivation for this naming), if there is a domain D and a hypergraph G such that $F = F_{[D]}(G)$. Note that G is the variable hypergraph of F here (since $D \neq \emptyset$), while we can recover $D =: D(F)$ from F in case we have at least one variable (that is, G contains at least one non-empty edge); if $\mathrm{var}(F) = \emptyset$, then we set $D(F) := \{0\}$. Thus, colouring clause-sets F are characterised by having a universal domain $D(F)$ and consisting of $|D(F)|$ many copies of some hypergraph G, where each copy has one of the values of $D(F)$ as the constant value of all literals in it. We denote the set of all colouring clause-sets by $\boldsymbol{\mathcal{CCLS}}$. A clause-set $F \in \mathcal{CLS}$ is colouring iff $F \setminus \{\bot\}$ is iff $F \cup \{\bot\}$ is. If for $F \in \mathcal{CCLS}$ we have $|D(F)| \leq 1$, then F is lean.

Lemma 7.6 *Consider a hypergraph G, domains D_1, D_2 with $\emptyset \neq D_1 \subseteq D_2$, and a partial assignment $\varphi_1 \in \mathcal{PASS}(\mathcal{VA}_{D_1})$. Obtain $\varphi_2 \in \mathcal{PASS}(\mathcal{VA}_{D_2})$ from φ_1 by replacing variables $v_1 \in dom(\varphi_1)$ by the corresponding variable $v_2 \in \mathrm{var}_{D_2}(\mathrm{var}_{D_1}^{-1}(v))$ and setting $\varphi_2(v_2) := \varphi_1(v_1)$.*

1. *If φ_1 is an autarky for $F_{[D_1]}(G)$, then φ_2 is an autarky for $F_{[D_2]}(G)$. Thus if $F_{[D_2]}(G)$ is lean, then so is $F_{[D_1]}(G)$.*
2. *If φ_1 is a linear autarky for $F_{[D_1]}(G)$, then φ_2 is a linear autarky for $F_{[D_2]}(G)$. Thus if $F_{[D_2]}(G)$ is linearly lean, then so is $F_{[D_1]}(G)$.*

Proof: Part 1 follows by the observation, that for the clauses in $F_{[D_2]}(G)$ corresponding to the clauses in $F_{[D_1]}(G)$ (obtained by replacing variables $\mathrm{var}_{D_1}(v)$ by $\mathrm{var}_{D_2}(v)$ for $v \in V(G)$) nothing changes, while the additional clauses all get satisfied. For Part 2 we additionally note, that actually all literals in the additional clauses become true, and thus φ_1 is always a linear autarky w.r.t. the additional clauses. ∎

Lemma 7.7 *Consider a hypergraph G and a domain D with $0, 1 \in D$.*

1. *Consider some $\varepsilon \in D$. If a partial assignment φ is a balanced linear autarky for $F_{[\{\varepsilon\},D]}(G)$, then φ is a linear autarky for $F_{[D]}(G)$.*
2. *Thus, if $F_{[D]}(G)$ is linearly lean, then $F_{[\{0\},\{0,1\}]}(G)$ is balanced linearly lean.*

Proof: For Part 1 note, that if φ is a balanced linear autarky for $F_{[\{\varepsilon\},D]}(G)$, then for every hyperedge touched by φ (using the correspondence between vertices and variables of domain D) there is a vertex (variable) v in H with $\varphi(v) = \varepsilon$, and thus φ satisfies all clauses in $F_{[D\setminus\{\varepsilon\},D]}(G)$ it touches. Part 2 follows from Part 1 and Lemma 7.5, Part 5. ∎

We remark that if G has no hyperedge of length at most one, then $F_{[D]}(G)$ is linearly satisfiable for every domain D with $|D| \geq |V(G)|$.

Theorem 7.8 *Consider a hypergraph G without non-covered vertices. If there exists a domain D with $|D| \geq 2$ such that $F_{[D]}(G)$ is linearly lean, then there is a matching in $B(G)$ covering all vertex nodes, whence $|E(G)| \geq |V(G)|$.*

Proof: W.l.o.g. $0, 1 \in D$. By Lemma 7.7, Part 2 it follows that $F_{[\{0\},\{0,1\}]}(G)$ is balanced linearly lean. By Lemma 7.2 then we obtain a matching as required. ∎

Some remarks:

1. By virtue of Lemma 7.5, in case of $D = \{0, 1\}$ Theorem 7.8 is a special case of Lemma 7.2; so the point of Theorem 7.8 is the generalised domain. In Lemma 7.10 we give an application of Theorem 7.8 which boils down to the boolean case, exploiting (essentially) Lemma 7.3 and thus actually establishing balanced linear leanness directly; however in Corollary 8.2 we will infer linear leanness from minimally unsatisfiability, and there the generalisation of Theorem 7.8 plays its role.
2. Consider a hypergraph G without non-covered vertices and without the empty edge, and a domain D. Then by (4) we have
   (a) $|E(G)| \geq |V(G)| \Leftrightarrow \delta(F_{[D]}(G)) \geq n(F)$
   (b) $|E(G)| = |V(G)| \Leftrightarrow \delta(F_{[D]}(G)) = n(F)$.

   Thus G is square iff $F_{[D]}(G)$ has the minimal possible deficiency.
3. If we allow G with non-covered vertices, then the these vertices are ignored by the matching and the counting of vertices in Theorem 7.8.

7.4 The inequality of Fisher

To conclude this section, we give an application of Theorem 7.8 to a well-known inequality from design theory, in order to illustrate the role of autarky theory for this kind of investigations (there is no really new element in the proof, but the main point is the application of the general framework we developed). First a (well-known) lemma from elementary linear algebra:

Lemma 7.9 *For $n \in \mathbb{N}$ let A be a square matrix of dimension n and let $a \in \mathbb{R}_{>0}$, such that all non-diagonal entries of A are equal to a, while all diagonal entries are strictly greater than a. Then A is non-singular.*

Now we prove the (generalised) inequality of Fisher (see Section II.2 in [4], or Section 2 in [23], or Theorem 5.12 in [15]):

Lemma 7.10 *Consider a non-trivial and non-degenerated pairwise balanced design G, and let G_0 be the underlying hypergraph of G. Then $B(G_0)$ contains a matching covering all points of G_0, whence $|E(G)| \geq |E(G_0)| \geq |V(G_0)| = |V(G)|$.*

Proof: We show that $F_{[\{0,1\}]}(G_0)$ is linearly lean, which proofs the assertion by Theorem 7.8. Let $V(G) = \{v_1, \ldots, v_n\}$, $n = |V(G)|$ and $E(G) = \{H_1, \ldots, H_m\}$, $m = |E(G)|$. Consider the point-adjacency matrix A of G, which is a square matrix of dimension n, having entries at position (i, j) equal to the number of blocks incident with points v_i, v_j (simultaneously). By definition all non-diagonal entries are equal to the index λ of G, while elementary reasoning shows that all diagonal entries are strictly greater than λ (otherwise G would be a trivial). Thus by Lemma 7.9 we obtain, that A is non-singular. Now consider the point-block incidence matrix M of G, which is an $n \times m$-matrix over $\{0, 1\}$ with the entry at position (i, j) equal to 1 iff v_i is incident with H_j. By definition we have $M \cdot M^{\mathrm{t}} = A$, and thus the

column-rank of M must be at least n (by elementary linear algebra). It follows that the column-rank of M_0, the point-block incidence matrix of G_0, must be at least n as well, since removal of repeated columns does not change the column rank. By equality of column-rank and row-rank M_0 thus has linearly independent rows, or, in other words, the clause-variable-matrix $M(F_{[\{0\},\{0,1\}]}(G_0))$ has linearly independent columns. By Lemma 7.1 it follows that $F_{[\{0\},\{0,1\}]}(G_0)$ is balanced linearly lean, which in turn is equivalent to $F_{[\{0,1\}]}(G_0)$ being linearly lean (see Lemma 7.5). ∎

Since we used only a boolean domain in the proof of Lemma 7.10, we could have used Lemma 7.2 instead of Theorem 7.8, however it seems that the general "interface" for access to hypergraph inequalities "$|E(G)| \geq |V(G)|$" is given by Theorem 7.8, and so we wanted to emphasise the use of this general tool. In the next section a genuine application of Theorem 7.8 is presented.

8 Applications to/of hypergraph colouring

Like the graph colouring problem, the hypergraph colouring problems lacks a notion of "substitution", which for example would allow to split a hypergraph k-colouring problem into k hypergraph k-colouring subproblems by giving some distinct vertex one of the k possible colours. By embedding the hypergraph colouring problem into the richer context of satisfiability problems for (generalised) clause-sets we gain such closure under substitution, while the translation is very direct, and does not mask structure. In a certain sense this translation "annotates" the hypergraph colouring problem, making available the rich set of operations on (generalised) clause-sets.

In Subsection 8.1 we discuss basic properties of the canonical translation (which was already introduced in Subsection 7.3), while the inequality of Seymour and related subjects (including "crown decompositions") one finds in Subsection 8.2. Finally in Subsection 8.3 the task of characterising minimally unsatisfiable colouring clause-sets of minimal deficiency is discussed, exploiting the central result of [47].

8.1 Translating hypergraph colouring problems

Consider a hypergraph G and $k \in \mathbb{N}$. We associate the generalised clause-set $\boldsymbol{F_{[k]}(G)} := F_{[\{1,\ldots,k\}]} \in \mathcal{CLS}$ to G (recall Subsection 7.3), that is, the variables of $F_{[k]}(G)$ are the vertices of G, while the clauses of $F_{[k]}(G)$ are the hyperedges of G in k versions corresponding to the k choices for the colour. We have that $n(F_{[k]}(G))$ is the number of vertices of G which are not non-covered, $D_v = \{1,\ldots,k\}$ for all $v \in \mathrm{var}(F_{[k]}(G))$, $c(F_{[k]}(G)) = k \cdot |E(G)|$, and $\ell(F_{[k]}(G)) = k \cdot \sum_{E \in E(G)} |E|$. By definition it is G k-colourable if and only if $F_{[k]}(G)$ is satisfiable, and moreover the set of k-colourings of G is identical to $\mathfrak{S}_{\mathrm{var}(F_{[k]}(G))}(F_{[k]}(G))$. The variable hypergraph of $F_{[k]}(G)$ is G. If actually we have a "list hypergraph colouring" problem, that is, for each vertex v only a subset of all colours is available, then we model this by restricting the domain D_v of variable v accordingly, and by removing all clauses C from $F_{[k]}(G)$ containing a literal (v, ε) with $\varepsilon \notin D_v$.

If we want to solve $F_{[k]}(G)$ by a (generalised) SAT solver, then it is useful to break the symmetry between the values $1,\ldots,k$ of the colouring as follows: Choose (different) variables $v_1,\ldots,v_{k-1} \in V(G)$ and use the restricted domains $D_{v_i} = \{1,\ldots,i\}$ for $i \in \{1,\ldots,k-1\}$; the clause-set obtained in this way is satisfiability-equivalent to $F_{[k]}(G)$, but the search space has been reduced. In practice, the savings obtained by this simple method are quite considerable, but for theoretical reasoning

we gain only unnecessary complications, and thus in this paper we do not use this elementary symmetry breaking, but we only use the canonical transformation $G \mapsto F_{[k]}(G)$.

Clause-sets expressing hypergraph colouring problems are up to renaming exactly the colouring clause-sets as defined in Subsection 7.3. A colouring clause-set F is satisfiable iff the variable-hypergraph of F is $|D(F)|$-colourable. Boolean colouring clause-sets are special cases of "PN-clause-sets" as introduced in [21] (boolean clause-sets, where each clause either is positive or negative), which are the heart of the class of bipartite clause-sets (clause-sets, where the conflict graph is bipartite).

Lemma 8.1 *A colouring clause-set $F \in \mathcal{CCLS}$ is minimally unsatisfiable if and only if the variable hypergraph of F is minimally non-$|D(F)|$-colourable. Put into the hypergraph perspective: A hypergraph G is minimally non-k-colourable for $k \in \mathbb{N}$ if and only if $F_{[k]}(G)$ is minimally unsatisfiable.*

Proof: Consider a colouring clause-set F, let $k := |D(F)| \in \mathbb{N}$, and let G denote the variable hypergraph of F. Obviously, if F is minimally unsatisfiable, then G is minimally non-k-colourable. Now assume that G is minimally non-k-colourable, but F is not minimally unsatisfiable. Thus there exists $C \in F$ such that $F \setminus \{C\}$ is unsatisfiable. If $C = \bot$, then $G = (\emptyset, \{\emptyset\})$, and thus $F = \{\bot\}$ would be minimally unsatisfiable; so let $\varepsilon \in D(F)$ be the common value of the literals in C. The hypergraph G minus the edge $\mathrm{var}(C)$ has a $D(F)$-colouring f. It must f colour all vertices in hyperedge $\mathrm{var}(C)$ with the same colour $\varepsilon' \in D(F)$. Consider a bijection $\pi \in S_{D(F)}$ with $\pi(\varepsilon') = \varepsilon$, and set $f' := \pi \circ f$. Now f' (which still is a $D(F)$-colouring for G) is a satisfying assignment for $F \setminus \{C\}$ contradicting the assumption. ∎

8.2 On an inequality of Seymour

Lemma 8.1 together with Theorem 7.8 yields immediately

Corollary 8.2 *Consider a hypergraph G without non-covered vertices, which is minimally non-k-colourable for some $k \geq 2$. Then there is a matching in $B(G)$ covering all vertex nodes, whence $|E(G)| \geq |V(G)|$.*

Corollary 8.2 shows the potential of the approach of translating hypergraph colouring problems into satisfiability problems for generalised clause-sets: If a hypergraph G is critical k-colourable, then G is not critical k-colourable for any $k' < k$, and thus there is no direct way to reduce the assertion for $k > 2$ to the known case $k = 2$. However from $F_{[k]}(G)$ being minimally unsatisfiable we can conclude that $F_{[k]}(G)$ is lean, and if $F_{[k]}(G)$ is lean, then also $F_{[k']}(G)$ is lean for all $k' < k$ (an autarky for $F_{[k']}(G)$ is also an autarky for $F_{[k]}(G)$ modulo the (injective) variable replacements involved). And the lower bound on the deficiency does not need a minimality condition, but only a certain leanness condition.

More precisely, we need that $F_{[2]}(G)$ is *linearly* lean, and since the lean kernel of $F_{[2]}(G)$ w.r.t. linear autarkies is computable in polynomial time (together with a corresponding sequence of linear autarkies; see [33]), we get the following strengthening of Corollary 8.2:

Corollary 8.3 *Consider a hypergraph G. In polynomial time a map $f : V' \to \{1, 2\}$ for some $V' \subseteq V(G)$ can be computed, such that f is a (proper) 2-colouring of*

$G' := (V', \{E \in E(G) : E \cap V' \neq \emptyset\})$ *(the hypergraph with vertex set* V' *and hyperedges the hyperedges from* G *which have some vertex in common with* V'*), and such that* $G'' := (V \setminus V', \{E \in E(G) : E \cap V' = \emptyset\})$ *fulfils* $|E(G'')| \geq |V(G'')|$*. (If* G *is minimally non-*k*-colourable for some* $k \geq 2$*, and has no non-covered vertices, then necessarily* $G'' = G$ *holds.)*

For general boolean clause-sets F for the computation of the lean kernel w.r.t. linear autarkies a series of linear programming problems needs to be solved. However, if F is colouring, every linear autarky is a balanced linear autarky, and thus instead of finding a non-trivial solution of $M(F) \cdot x \geq 0$ (where $M(F)$ is the clause-variable matrix) we only need to solve systems $M(F) \cdot x = 0$ of linear equations. So the polynomial-time computation for Corollary 8.3 is quite practical.

Corollary 8.3 is essentially the same as Corollary 6 in [43], but for their proof the authors used so-called "crown decomposition", which in this setting is closely related to reductions by matching autarkies (see Lemma 5 in [43]).

To conclude this subsection, I would like to point out the following interesting relaxation of the chromatic number of hypergraphs:

Consider a colourable hypergraph G. It is $F_{[1]}(G)$ lean, while $F_{[\chi(G)]}(F)$ is satisfiable, and thus not lean in case of $V(G) \neq \emptyset$. And if $F_{[k]}(G)$ is lean, then so is $F_{[k']}(G)$ for $k' \leq k$. So I propose to study the **autarky number** $\boldsymbol{\chi_{\mathrm{a}}(G)} \leq \chi(G)$, the maximal k so that $F_{[k]}(G)$ is lean. Corollary 8.2 then can be generalised as the assertion, that a hypergraph without non-covered vertices and with $\chi_{\mathrm{a}}(G) \geq 2$ has at least as many edges as vertices. Via the use of autarky systems the autarky number of hypergraphs can be generalised (considering for example the *matching autarky number*); by introducing an appropriate normal autarky system capturing "boolean balanced linear autarkies", and using the autarky number associated with that autarky system we then arrive at what looks as a very natural "most general condition" for having at least as many edges as vertices.

8.3 Characterising minimally unsatisfiable colouring clause-sets

For a clause-set F denote by $\boldsymbol{G(F)}$ the variable hypergraph of F. And let

$$\boldsymbol{\mathcal{MUCCLS}} := \mathcal{MUSAT} \cap \mathcal{CCLS}$$

be the set of minimally unsatisfiable colouring clause-sets. In this final section we look into the issue of characterising $\mathcal{MUCCLS}$, those minimally unsatisfiable clause-sets F which are determined by their variable graph $G(F)$ (and the chromatic number of $G(F)$). The following lemma recollects the basic information about (minimally unsatisfiable) colouring clause-sets.

Lemma 8.4 *The class* $\mathcal{CCLS}$ *of colouring clause-sets has the following properties:*

1. *(a) For all domains* D, D' *and all hypergraphs* G, G' *without non-covered vertices we have* $F_{[D]}(G) \in \mathcal{CCLS}$*, where* $F_{[D]}(G) = F_{[D']}(G')$ *if and only if* $(D = D'$ *and* $G = G')$ *or* $(G = G'$ *and* $V(G) = V(G') = \emptyset)$*.*
   (b) For every $F \in \mathcal{CCLS}$ *we have* $F = F_{[D(F)]}(G(F))$*.*
   (c) For every $F \in \mathcal{CCLS}$ *we have* $F \in \mathcal{MUSAT}$ *if and only if* $G(F)$ *is minimally non-*$|D(F)|$*-colourable.*

2. *For a clause-set $F \in \mathcal{CLS}$ we have:*

   (a) $F \in \mathcal{CCLS} \Leftrightarrow F \setminus \{\bot\} \in \mathcal{CCLS}$.

   (b) $\mathrm{var}(F) = \emptyset \Rightarrow F \in \mathcal{CCLS}$.

   (c) *Assume $\bot \notin F$ and $\mathrm{var}(F) \neq \emptyset$. Then $F \in \mathcal{CCLS}$ holds if and only if there is a domain D and $\varepsilon : F \to D$ with the following properties:*

      i. *For variables $v \in \mathrm{var}(F)$ we have $D_v = D$.*
      ii. *For $C \in F$ and $x \in C$ we have $\mathrm{val}(x) = \varepsilon(C)$.*
      iii. *For $C, C' \in F$ we have $C = C' \Leftrightarrow \mathrm{var}(C) = \mathrm{var}(C') \wedge \varepsilon(C) = \varepsilon(C')$.*
      iv. *$c(F) = |D| \cdot |E(G(F))|$.*

   D and ε are uniquely determined here.

3. *The set of all $F \in \mathcal{MUCCLS}_{|D|=1}$ is the set of all $F_{[D]}(G)$ for $|D| = 1$ and $|E(G)| = 1$ (and thus $\mathcal{MUCCLS}_{|D|=1} \subset \mathcal{MUSAT}_{\delta=1}$).*

4. *For $F \in \mathcal{MUCCLS}$ with $|D(F)| \geq 2$ we have $\delta(F) \geq n(F)$.*

5. *For $F \in \mathcal{MUCCLS}$ we have $\delta(F) = n(F)$ if and only if $G(F)$ is square.*

Proof: Follows by Remark 2 to Theorem 7.8, and with Lemma 8.1 and Corollary 8.2. ∎

We set out to classify $\mathcal{MUCCLS}_{\delta=n}$, the set of minimally unsatisfiable colouring clause-sets where the deficiency equals the number of variables, which by Lemma 8.4 amounts to classify all pairs (G, k) such that $k \in \mathbb{N}$ and G is a minimally non-k-colourable square hypergraph G. Except of the trivial instances of $\mathcal{MUCCLS}$ characterised in Lemma 8.4, Part 3, $\mathcal{MUCCLS}_{\delta=n}$ is the set of instances of $\mathcal{MUCCLS}$ with "relatively lowest" deficiency.

Now actually it is not clear classifying $\mathcal{MUCCLS}_{\delta=n}$ is feasible (at all); here we will only classify the class $\mathcal{MUCCLS}_{\delta=n} \cap \mathcal{MHIT}$ (the multihitting minimally unsatisfiable colouring clause-sets of minimal deficiency), which can be subdivided into the level $\mathcal{MUCCLS}^{\mathrm{mh}=k}_{\delta=n}$ for $k \in \mathbb{N}$. For fixed k the classification of $\mathcal{MUCCLS}^{\mathrm{mh}=k}_{\delta=n}$ amounts to classify all *intersecting* minimally non-k-colourable square hypergraphs.

From Corollary 6.9 we get, that a subsumption-free multihitting clause-set is already minimally unsatisfiable if it is just unsatisfiable. Thus

Corollary 8.5 *Consider a simple intersecting hypergraph G and $k \in \mathbb{N}$. Then G is minimally non-k-colourable if and only if G is not k-colourable.*

Simple intersecting hypergraphs are "almost" 2-colourable:

Lemma 8.6 *Consider a simple intersecting hypergraph G. For all hyperedges $H \in E(G)$ then $G - \{E\} = (V(G), E(G) \setminus \{E\})$ is 2-colourable.*

Proof: Define a map $f : V(G) \to \{1, 2\}$ by mapping the points of H to 1, and every other point to 2. f is a 2-colouring of $G - \{H\}$, since any hyperedge different from H has a non-empty intersection with H and contains vertices not in H. ∎

A hypergraph G is called a *square pencil* if G is square, intersecting and $|V(G)| = 1$ (thus for $v \in V(G)$ we have $E(G) = \{\{v\}\}$). Note that square pencils are not colourable.

Corollary 8.7 *An intersecting hypergraph G is 3-colourable if and only if G is not a square pencil.*

For the ease of reference we note the following direct implication of Lemma 8.6 (some designs will yield key examples):

Corollary 8.8 *Consider a dual linear design G with at least 2 points on each line. G is 2-colourable if any line is removed (only from the line set, while the point set is kept). Thus G is 3-colourable, while G is critical 3-colourable if and only if G is not 2-colourable (that is, has no blocking set).*

For multihitting colouring clause-sets we see, that regarding (un)satisfiability only boolean clause-sets are of real interest:

Corollary 8.9 *A multihitting colouring clause-set F with multihitting number at least three is unsatisfiable if and only if $n(F) = 1 = \delta(F)$.*

Regarding the classification of minimally unsatisfiable multihitting colouring clause-set we obtain, that all levels $\mathcal{MUCCLS}^{\mathrm{mh}=k}_{\delta=n}$ for $k \neq 2$ are trivial:

Lemma 8.10 *The elements of $\mathcal{MUCCLS}^{\mathrm{mh}=k}_{\delta=n}$ for $k \in \mathbb{N} \setminus \{2\}$ are the $F_{[D]}(G)$ for domains D with $|D| = k$ where G is a square pencils.*

Proof: The case $k = 1$ is trivial, while the remaining cases follow from Corollary 8.9. ∎

For the remaining case $k = 2$ we investigate projective incidence planes.

Lemma 8.11 *A projective incidence plane G of order $k \in \mathbb{N}_0$ is minimally non-2-colourable iff $k = 2$, and in this case $F_{[2]}(G) \in \mathcal{MUCCLS}^{\mathrm{mh}=2}_{\delta=n}$, while G is 2-colourable for $k \geq 3$. Among the degenerated projective incidence planes exactly the following types are square and minimally non-k-colourable for some $k \in \mathbb{N}$:*

1. *A square pencil P; here $F_{[k]}(P) \in \mathcal{MUCCLS}^{\mathrm{mh}=k}_{\delta=n}$ for all $k \in \mathbb{N}$.*
2. *A "near pencil", a hypergraph N with $|V(N)| \geq 3$ and $|E(N)| = |V(N)|$ containing a (unique) point $v \in V(N)$ such that $E(N) = \{V(N) \setminus \{v\}\} \cup \{\{v, w\}\}_{w \in V(N) \setminus \{v\}}$; these N are intersecting and minimally non-2-colourable. Thus $F_{[2]}(N) \in \mathcal{MUCCLS}^{\mathrm{mh}=2}_{\delta=n}$.*

Proof: As we have remarked, 2-colourability of G is equivalent to the existence of a blocking set (an independent transversal), and a projective plane contains a blocking set iff the order is at least 3 (this is well known, and follows also immediately from Corollary 8.13 below). The remaining assertions follow by Corollary 8.8 and elementary reasoning. ∎

Lemma 8.11 yields three key examples of intersecting minimally non-2-colourable square hypergraphs:

1. The square pencil $(\{1\}, \{\{1\}\})$.
2. The near pencil with three vertices, i.e., the triangle

$$C^3 = (\{1, 2, 3\}, \{\{1, 2\}, \{2, 3\}, \{1, 3\}\}).$$

3. The projective plane PG(2, 2) of order 2, i.e., the Fano plane

$$(\{1,\dots,7\}, \{\, \{1,2,3\},\{3,4,5\},\{1,5,6\},\{1,4,7\},\{2,5,7\},\{3,6,7\},\{2,4,6\} \,\}).$$

In [47] (Proposition 7) it was shown that these three hypergraphs plus a certain extension process exactly produce all intersecting minimally non-2-colourable square hypergraphs (solving the classification of $\mathcal{MUCCLS}^{\mathrm{mh}=2}_{\delta=n}$):

Theorem 8.12 *[47] The class of intersecting minimally non-2-colourable square hypergraphs is the class of hypergraphs containing square pencils and projective planes of order 2, and all "2-extensions of C^3", which is the class of hypergraphs containing near pencils with three vertices and if G belongs to this class, then also G' arising from G by choosing an edge $\{x_1, x_2\} \in E(G)$ of length 2 and some new vertex $z \notin V(G)$, and adding the edge $\{x_1, z\}$ as well as adding the vertex z to all edges of G except of $\{x_1, x_2\}$.*

Corollary 8.13 *Every intersecting simple square hypergraph G (by definition, this includes square dual linear designs) fulfilling at least one the following (sufficient) criterions is 2-colourable (and thus has a blocking set):*

1. *G has only hyperedges of size at least 4.*
2. *G has only hyperedges of size at least 3, and G has also hyperedges of size at least 4 or $|V(G)| \geq 8$.*
3. *G contains hyperedges of sizes 1 and 2.*

Proof: By Theorem 8.12 such a hypergraph is not minimally non-2-colourable, and thus by Corollary 8.5 it must be 2-colourable. ∎

By Lemma 8.10 and Theorem 8.12 we finally obtain:

Theorem 8.14 *The class $\mathcal{MUCCLS}_{\delta=n} \cap \mathcal{MHIT}$ is exactly the class of clause-sets F fulfilling the following conditions:*

1. *If F contains a unit-clause, then $G(F)$ is a square pencil.*

   Otherwise $|D(F)| = 2$ must hold, i.e, up to renaming now F is boolean for the remaining cases.
2. *If F contains a binary clause, then $G(F)$ is a 2-extension of C^3 (see Theorem 8.12).*
3. *Otherwise $G(F)$ is isomorphic to the Fano plane.*

Corollary 8.15 *For $F \in \mathcal{CCLS}_{\delta=n} \cap \mathcal{MHIT}$ each of the following conditions is sufficient to guarantee satisfiability of F:*

1. *F has only clauses of length at least 4.*
2. *F has only clauses of length at least 3, and F has also clauses of size at least 3 or $n(F) \geq 8$ or $c(F) \geq 15$.*
3. *G contains clauses of sizes 1 and 2.*

9 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 systems related to this notion, matching autarkies and balanced linear 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-free, 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.

Turning to hypergraph theory, we gave a general method for proving that a hypergraph has a matching between vertices and hyperedges covering all vertices, exploiting here linear algebra, while in proving the analogous result for minimally unsatisfiable clause-sets we used matching theory. We applied this general method to Fisher's inequality, and, using the canonical translation of hypergraph colouring problems into generalised satisfiability problems, we obtained a generalisation of a well-known result of Seymour. Regarding the project of classifying minimally unsatisfiable clause-sets, the results of Seymour have been interpreted as the characterisation of the first non-trivial level of minimally unsatisfiable colouring clause-sets of minimal deficiency (where colouring clause-sets are those generalised clause-sets characterised completely by their variable hypergraph and their (uniform) variable domain).

The embedding of the hypergraph colouring problem into the richer space of generalised satisfiability problems enables operations which are external to the hypergraph environment, and can be expressed at this level only in clumsy ways. (However, it seems essential that this richer space is still "close enough" to the original space.) One of these operations is the operation of autarkies. Autarky-freeness ("leanness") yields a natural and "smooth" extension of the notion of "minimality".

There is a host of open problems, which seem to be non-trivial and opening new pathways into structural investigations of clause-sets and hypergraphs, which we will discuss in the remainder.

9.1 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 [6]. 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).

In [28] it was shown 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 $\#_{(v,\varepsilon)}(F) \leq \delta(F)$. In Lemma 6.14 this property was shown to hold also for (generalised) minimally unsatisfiable clause-sets in case of $\delta(F) = 1$ — does it also hold for arbitrary (generalised) minimally unsatisfiable clause-sets ?

9.2 Uniform hitting clause-sets

Regarding the base case of deficiency 1 for minimally unsatisfiable clause-sets, a natural question is, whether every uniform unsatisfiable hitting clause-set has necessarily deficiency 1 ? (We remark here that obviously $\mathcal{UHIT}_{\mathrm{hd}=r}^{\mathrm{sat}=0}$ is empty for $r \geq 2$, since no resolution is possible here.) We conjecture that this is the case:

Conjecture 9.1 $\mathcal{UHIT}^{\mathrm{sat}=0} = \mathcal{UHIT}_{\delta=1}^{\mathrm{sat}=0}$.

From Conjecture 9.1 it would follow by Corollary 6.17, that the class of unsatisfiable hitting (generalised) clause-sets is equal to the class of saturated minimally unsatisfiable clause-sets of deficiency 1, generalising Corollary 34 in [35].

In [35] the property $\delta(F) \leq 1$ was actually shown for arbitrary (not necessarily unsatisfiable) boolean uniform 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 [35] 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 9.2 $\mathcal{UHIT} = \mathcal{UHIT}_{\delta \leq 1}$.

Note that Conjecture 9.2 implies Conjecture 9.1 (using Corollary 4.21). In the terminology of graph partitions, Conjecture 9.2 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 9.2 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).

9.3 Autarkies

Is the problem NOT-ALL-EQUAL-2-SAT for generalised (2-)clause-sets NP-complete? (As discussed in Subsection 7.1, this is equivalent to the problem of deciding whether a generalised 2-clause-sets is satisfiable by a balanced linear autarky.)

Is the concept of linear autarkies for generalised clause-sets useful? For example are there interesting poly-time computable subclasses (besides the boolean subcase)?

9.4 Hypergraphs

Are there further applications of Theorem 7.8? Can the autarky number of hypergraphs as discussed in Subsection 8.2 be made useful?

9.5 Characterising minimally unsatisfiable colouring clause-sets of minimal deficiency

What about the classification of $\mathcal{MUCCLS}_{\delta=n}$? Is one of the following classes poly-time decidable:

1. $\mathcal{MUCCLS}_{\delta=n}$?
2. $\mathcal{MUCCLS}_{\delta=n} \cap \mathcal{MHIT}$? (As we have seen, deciding this class boils down to deciding $\mathcal{MUCCLS}^{\mathrm{mh}=2}_{\delta=n}$).

Finally, we can wonder whether there might be a generalised structure theorem for classes of minimally unsatisfiable (generalised) clause-sets of "relative" minimal deficiency? Can for example the de Bruijn-Erdös classification of square linear designs be made fruitful here?

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 [27], 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] Thomas Beth, Dieter Jungnickel, and Hanfried Lenz. *Design Theory: Volume I*, volume 69 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, second edition, 1999. ISBN 0-521-44432-2.

[5] 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.

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

[7] Hans Kleine Büning and Xishun Zhao. The complexity of some subclasses of minimal unsatisfiable formulas. December 2003.

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

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

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

[11] 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.

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

[13] Peter Dembowski. *Finite Geometries*, volume 44 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer, 1997. ISBN 3-540-61786-8; reprint of the 1968 edition.

[14] 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.

[15] Pierre Duchet. Hypergraphs. In Graham et al. [24], chapter 7, pages 381–432. ISBN 0-444-82346-8; QA164.H33 1995.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] Nicola Galesi and Oliver Kullmann. Polynomial time SAT decision, hypergraph transversals and the hermitian rank. In Hoos and Mitchell [27], pages 89–104. ISBN 3-540-27829-X.

[22] 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.

[23] Chris D. Godsil. Tools from linear algebra. In Ronald L. Graham, Martin Grötschel, and László Lovász, editors, *Handbook of Combinatorics, Volume 2*, chapter 31, pages 1705–1748. North-Holland, Amsterdam, 1995. ISBN 0-444-82351-4; QA164.H33 1995.

[24] Ronald L. Graham, Martin Grötschel, and László Lovász, editors. *Handbook of Combinatorics, Volume 1.* North-Holland, Amsterdam, 1995. ISBN 0-444-82346-8; QA164.H33 1995.

[25] 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.

[26] 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.

[27] 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.

[28] Oliver Kullmann. An application of matroid theory to the SAT problem. In *Fifteenth Annual IEEE Conference on Computational Complexity (2003)*, pages 116–124.

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

[30] 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.

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

[32] 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.

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

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

[35] Oliver Kullmann. The combinatorics of conflicts between clauses. In Giunchiglia and Tacchella [22], pages 426–440. ISBN 3-540-20851-8.

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

[37] 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.

[38] Oliver Kullmann. Conjunctive normal forms with non-boolean variables, autarkies and hypergraph colouring. Technical Report CSR 5-2005, University of Wales Swansea, Computer Science Report Series (`http://www-compsci.swan.ac.uk/reports/2005.html`), March 2005.

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

[42] Nathan Linial and Michael Tarsi. Deciding hypergraph 2-colourability by H-resolution. *Theoretical Computer Science*, 38:343–347, 1985.

[43] Daniel Lokshtanov and Christian Sloper. Fixed parameter set splitting, linear kernel and improved running time. In Hajo Broersma, Matthew Johnson, and Stefan Szeider, editors, *Algorithms and Complexity in Durham 2005*, volume 4 of *Texts in Algorithms*, pages 105–113. Kings College London Publications, 2005.

[44] 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.

[45] Steven Prestwich. Local search on SAT-encoded colouring problems. In Giunchiglia and Tacchella [22], pages 105–119. ISBN 3-540-20851-8.

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

[47] P.D. Seymour. On the two-colouring of hypergraphs. *The Quarterly Journal of Mathematics (Oxford University Press)*, 25:303–312, 1974.

[48] 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.

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

[50] Hans van Maaren. A short note on some tractable cases of the satisfiability problem. *Information and Computation*, 158(2):125–130, May 2000.

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