Journal of Artificial Intelligence Research 51 (2014) 707-723 Submitted 07/14; published 12/14

On Minimum Representations of Matched Formulas

Ondřej Čepek ONDREJ.CEPEK@MFF.CUNI.CZ
Štefan Gurský STEVKO@MAIL.RU
Petr Kučera KUCERAP@KTIML.MFF.CUNI.CZ
Charles University in Prague
Faculty of Mathematics and Physics
Department of Theoretical Computer Science
Malostranské nám. 25, 118 00 Praha 1, Czech Republic

Abstract

A Boolean formula in conjunctive normal form (CNF) is called matched if the system of sets of variables which appear in individual clauses has a system of distinct representatives. Each matched CNF is trivially satisfiable (each clause can be satisfied by its representative variable). Another property which is easy to see, is that the class of matched CNFs is not closed under partial assignment of truth values to variables. This latter property leads to a fact (proved here) that given two matched CNFs it is co-NP complete to decide whether they are logically equivalent. The construction in this proof leads to another result: a much shorter and simpler proof of Σ_2^p-completeness of Boolean minimization for matched CNFs. The main result of this paper deals with the structure of clause minimum CNFs. We prove here that if a Boolean function f admits a representation by a matched CNF then every clause minimum CNF representation of f is matched.

1. Introduction

In this paper we study the class of matched formulas introduced by Franco and Van Gelder (2003). Given a formula φ in conjunctive normal form (CNF) we consider its *incidence graph* $I(\varphi)$ defined as follows. $I(\varphi)$ is a bipartite graph with one part consisting of clauses of φ and the other part containing the variables of φ. An edge $\{C, x\}$ for a clause C and variable x is in $I(\varphi)$ if x appears in C. It was observed by Aharoni and Linial (1986), and Tovey (1984) that if $I(\varphi)$ admits a matching (i.e. a set of pairwise disjoint edges) of size m (where m is the number of clauses in φ), then φ is satisfiable. Later the formulas satisfying this condition were called *matched formulas* by Franco and Van Gelder (2003). Since a matching of maximum size in a given graph can be found in polynomial time (e.g., Lovász & Plummer, 1986), we can check whether a given formula is matched. Given an arbitrary CNF φ we can measure how far it is from being matched by considering its maximum deficiency $\delta^*(\varphi)$, the number of clauses which remain unmatched in a maximum matching of $I(\varphi)$. A formula φ is thus matched iff $\delta^*(\varphi) = 0$. A weaker notion of deficiency $\delta(\varphi) = m - n$ (where m is the number of clauses and n the number of variables in φ) is also often considered.

Since their introduction, matched formulas were considered as a base class in parameterized algorithms for satisfiability (e.g., for an overview of parameterized algorithms theory see Flum & Grohe, 2006). In particular, Fleischner, Kullmann, and Szeider (2002) show that satisfiability of the formulas whose maximum deficiency is bounded by a constant can be decided in polynomial time. This result was later improved by Szeider (2003) to an algorithm for satisfiability parameterized with maximum deficiency of a formula. Parameterization based on backdoor sets with respect to matched formulas were considered by Szeider (2007).

Several generalizations of matched formulas were considered in the literature, too. Kullmann (2000) generalized the class of matched formulas into the class of linearly satisfiable formulas and Kullmann (2003) studied autarkies based on matchings. Another generalization was considered by Szeider (2005) as classes of bi-clique satisfiable and var-satisfiable formulas. Unfortunately, for both bi-clique and var-satisfiable formulas it is hard to check if a formula falls into one of these classes (Szeider, 2005).

The results listed in the previous paragraphs show that matched formulas play a significant role in the theory of satisfiability solving which is, without any doubt, one of the most studied problems in theoretical computer science that has many practical applications. Despite this fact, little is known about the structure of matched CNFs. We say that a CNF φ representing a function f is *irredundant* if it is set-minimal representation of f, i.e. if for any clause C in φ we have that $\varphi' = \varphi \setminus \{C\}$ does not represent f. We say that φ is a *prime CNF* representing f if all clauses in φ are prime implicates of f, where a clause C is an *implicate* of f if it is satisfied by all assignments satisfying f and it is a *prime implicate* if it is a set-minimal implicate of f (considering a clause as a set of literals). If we only say that a CNF φ is prime without mentioning the function f in question we implicitly consider f to be the function represented by φ. It is not hard to come up with examples of matched CNFs such that some logically equivalent prime and irredundant CNFs are not matched. This is quite an interesting phenomenon which does not occur in most classes with polynomial time satisfiability testing such as quadratic CNFs (also 2-CNFs, i.e. CNFs consisting of clauses with at most two literals), Horn CNFs (a CNF is Horn if every clause in it contains at most one positive literal), and their various generalizations, for which once a CNF is in the class, all logically equivalent prime CNFs are guaranteed to be in the class as well. This brings an interesting question: given a (nonprime) matched CNF does there exist at least one equivalent prime CNF which is also matched? In this paper we give an affirmative answer to this question. This answer may seem to be quite intuitive because usually it is enough to consider prime implicates of a function when studying its CNF representations, however, the proof of this answer is not easy.

Another problem we study in this paper is Boolean minimization of matched CNFs. The Boolean minimization problem (BM) can be stated as follows: given a CNF find a logically equivalent CNF with a minimum possible number of clauses. The number of clauses can be viewed as a number of rules or implicates in a representation of a knowledge base and it is a standard measure used in this context. One can also consider the length of the formula, i.e. the total number of literal occurrences in the formula, as a measure of optimality, in this paper we use the number of clauses as a measure and we leave as an open question whether the results in this paper could be extended to the case of the formula length as well. The optimization version of the Boolean minimization problem can be turned into a decision version by adding a number k and asking whether there exists a logically equivalent CNF with at most k clauses. Umans (2001) showed that the decision version of BM is Σ_2^p-complete (e.g., for related results see the review paper Umans, Villa, & Sangiovanni-Vincentelli, 2006). Buchfuhrer and Umans (2011) later showed that BM is

Σ_2^p-complete when considering general formulas of constant depth as the input and output to the Boolean minimization problem.

It is also long known that BM is NP-hard already for some classes of CNFs where SAT is solvable in polynomial time. Maybe a best known example is the class of Horn CNFs where the NP-hardness with respect to both output measures was proved (Ausiello, D'Atri, & Sacca, 1986; Boros & Čepek, 1994; Čepek, 1995; Hammer & Kogan, 1993; Maier, 1980). There exists a hierarchy of tractable subclasses of Horn CNFs for which there are polynomial time minimization algorithms, namely acyclic and quasi-acyclic Horn CNFs (Hammer & Kogan, 1995), and CQ Horn CNFs (Boros, Čepek, Kogan, & Kučera, 2009). There are also few heuristic minimization algorithms for Horn CNFs (Boros, Čepek, & Kogan, 1998).

The complexity of BM for matched CNFs does not fit the above picture. Despite the fact that SAT is trivial for matched CNFs, BM for this class is Σ_2^p-complete, i.e. as hard as for the general case. This fact was proved by Gurský (2011), where the proof modifies the proof for the general case by Umans (2001). In this paper we give a much simpler proof of the same fact which is based on an observation, that equivalence testing is co-NP-complete for matched CNFs. We also study the structure of clause minimum CNFs. Based on the above mentioned result concerning prime CNFs we prove that if a Boolean function f admits a representation by a matched CNF then every clause minimum CNF representation of f is matched. This is the main result of the current paper.

The paper is structured as follows. We start by introducing the necessary notation, definitions and basic results in Section 2. In Section 3 we prove that testing logical equivalence for matched CNFs is co-NP-complete and further use the idea from this proof to show that BM for matched formulas is Σ_2^p-complete. This is a known fact, but the current proof is much shorter and simpler. Section 4 studies prime representations of functions defined by matched CNFs (possibly nonprime). We prove that every such function has at least one prime representation which is matched. Finally, in Section 5 we study the structure of clause minimum representations of functions defined by matched CNFs. Using the result from Section 4 we prove that if a Boolean function f admits a representation by a matched CNF then every clause minimum CNF representation of f is matched. Section 6 concludes the paper with closing remarks.

2. Definitions and Results

We shall start with the basic definitions of the notions we need in this paper. We shall also recall the results we shall use in this paper.

2.1 Boolean Functions

A *Boolean function of n variables* is a mapping $f : \{0,1\}^n \to \{0,1\}$. A *literal* is either a variable (x, called *positive literal*) or its negation ($\neg x$ or $\overline{x}$, called *negative literal*). A *clause* is a disjunction of literals. We assume that no clause contains both positive and negative literals of the same variable. Formula φ is in *conjuntive normal form (CNF)* if it is a conjuction of clauses (we also say that φ is a CNF formula). We shall often treat a clause as a set of its literals and a CNF formula as a set of its clauses. Thus $|\varphi|$ will denote the number of clauses in φ. It is a well known fact that every Boolean function can be represented by a CNF formula (e.g., Genesereth & Nilsson, 1987). If two CNF formulas φ_1 and φ_2 define the same function, we say that they are *equivalent* and we denote this fact with $\varphi_1 \equiv \varphi_2$. A CNF ψ is called *clause minimum* if for every CNF φ such that $\psi \equiv \varphi$ we have $|\psi| \leq |\varphi|$.

Clause C is called an *implicate of* f if every assignment $\vec{x} \in \{0,1\}^n$ satisfying f (i.e. $f(\vec{x}) = 1$) also satisfies C (i.e. $C(\vec{x}) = 1$). We say that *a clause* C_1 *subsumes a clause* C_2, if every literal from C_1 occurs also in C_2 (i.e. $C_1 \subseteq C_2$). C is a *prime implicate* of a function f if it is an implicate of f and there is no other implicate C' of f subsuming C (i.e. C is a set-minimal implicate of f). We say that CNF formula φ representing function f is a *prime* representation of f if it contains only prime implicates of f (if we refer to a prime CNF φ without specifying a function f we consider the function which is represented by φ). A CNF formula φ is *irredundant* if there is no sub-CNF $\varphi' \subset \varphi$ which represents the same function as φ.

An assignment t which assigns values to only a subset of (possibly to all) variables of a function f on n is called a *partial assignment*. Formally, a partial assignment can be viewed as a mapping $t : Y \mapsto \{0,1\}$ where Y is a subset of variables of f. Given a CNF φ, $\varphi(t)$ denotes the CNF after applying a partial assignment t. In particular $\varphi(t)$ is produced from φ in the following way: Clauses which contain some literal which is satisfied by t (assigned value 1) are removed from φ, occurences of literals on variables from Y which are not satisfied by t are removed from all clauses in φ.

2.2 Resolution

We say that two clauses have a *conflict in variable* x if there is a positive occurrence of x in one clause and a negative occurrence in the other. Two clauses $C_1 = (\widetilde{C_1} \vee x)$ and $C_2 = (\widetilde{C_2} \vee \overline{x})$ are *resolvable over* x if $\widetilde{C_1}$ and $\widetilde{C_2}$ do not have a conflict in any variable. We write $R(C_1, C_2) = \widetilde{C_1} \vee \widetilde{C_2}$ and this disjunction is called a *resolvent* of the *parent clauses* C_1 and C_2.

Let φ be a CNF formula representing a Boolean function f, we say that C can be derived from φ by a series of resolutions if there is a sequence of clauses $C_1, \ldots, C_k = C$ such that every C_i, $1 \leq i \leq k$, either belongs to φ, or $C_i = R(C_{j_1}, C_{j_2})$, where $j_1, j_2 < i$. Such a series of resolutions is also called a *resolution proof of* C *from* φ. A resolution proof of an empty clause (denoted as $\perp$) from an unsatisfiable formula is called *refutation*. The *length* of a resolution proof is the number of clauses in the sequence.

It is a well known fact that for any Boolean function the resolvent of two implicates is again an implicate (e.g., Büning & Lettmann, 1999). Another well known fact is that every prime implicate of f can be derived from any CNF representation φ of f by a series of resolutions (e.g., Büning & Lettmann, 1999).

We shall also use the notions of regular and tree-like resolution proofs. A resolution proof is a *tree-like resolution proof* if every occurrence of a clause in the proof is used at most once as the premise of a resolution where the only clause not used as a premise of a resolution is the conclusion; a tree-like resolution proof can be represented as a tree, where the leaves are labelled with input clauses, and the root of the tree with the conclusion of the proof. The *depth* of a tree-like resolution proof T is then the length of a longest path from a leaf to the root in T. A resolution proof is *regular* if in any path in the proof from an input clause to the conclusion, no variable is resolved more than once.

It can be observed that if φ is an unsatisfiable CNF, then it has a regular tree-like refutation (Urquhart, 2011), basically we can turn any resolution derivation to a tree-like resolution derivation by repeating clauses if necessary. Then any tree-like refutation can be turned into a regular tree-like refutation (Tseitin, 1983; Urquhart, 1995, 2011). It can be further observed that if C is an implicate which can be derived by a series of resolutions from φ, then there is a clause $C' \subseteq C$ which can be derived by a regular tree-like resolution T satisfying that no variable from C' is resolved in T. Indeed, if C is an implicate of φ, then let t be the partial assignment which assigns false to all literals in C. It follows that $\varphi(t)$ is an unsatisfiable formula, and thus it has a regular tree-like refutation T'. If we put back the falsified literals from C and the clauses satisfied by t, we get a resolution derivation of a sub-CNF C' from φ. The following proposition now follows immediately.

Lemma 2.1 *Let φ be a CNF and let C be a prime implicate of φ, then C can be derived by a regular tree-like resolution from φ.*

2.3 Exclusive Sets of Implicates of a Boolean Function

In this section we recall the definition of exclusive sets of implicates of a Boolean function and we state some of their properties, which were shown by Boros, Čepek, Kogan, and Kučera (2010).

Let us first introduce the following notation. By $\mathcal{I}^p(f)$ we shall denote the set of all prime implicates of a function f. By $\mathcal{I}(f)$ we shall denote the resolution closure of $\mathcal{I}^p(f)$, i.e. an implicate C of f belongs to $\mathcal{I}(f)$ if it can be derived by a series of resolutions from $\mathcal{I}^p(f)$.

Definition 2.2 (Boros et al., 2010) Let f be a Boolean function and let $\mathcal{X} \subseteq \mathcal{I}(f)$ be a set of clauses. We shall say, that $\mathcal{X}$ is an *exclusive set of clauses of f* if for every pair of resolvable clauses $C_1, C_2 \in \mathcal{I}(f)$ the following implication holds:

$$R(C_1, C_2) \in \mathcal{X} \Longrightarrow C_1 \in \mathcal{X} \text{ and } C_2 \in \mathcal{X},$$

i.e. the resolvent belongs to $\mathcal{X}$ only if both parent clauses are in $\mathcal{X}$. If the function f is clear from the context, we shall simply say that $\mathcal{X}$ is an exclusive set.

We shall recall some of the properties of exclusive sets, which were proved by Boros et al. (2010) and which we will use in this paper.

Lemma 2.3 (Boros et al., 2010) *Let $\mathcal{A}, \mathcal{B} \subseteq \mathcal{I}(f)$ be exclusive sets of implicates of f, then both $\mathcal{A} \cup \mathcal{B}$ and $\mathcal{A} \cap \mathcal{B}$ are also exclusive sets of implicates of f.*

Theorem 2.4 (Boros et al., 2010) *Let f be an arbitrary Boolean function, let $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathcal{I}(f)$ be two distinct sets of clauses which both represent f, and let $\mathcal{X} \subseteq \mathcal{I}(f)$ be an exclusive set of clauses. Then $\mathcal{C}_1 \cap \mathcal{X} \equiv \mathcal{C}_2 \cap \mathcal{X}$, i.e. both represent the same function.*

Based on this proposition we define an exclusive component of a Boolean function.

Definition 2.5 (Boros et al., 2010) Let f be an arbitrary Boolean function, $\mathcal{X} \subseteq \mathcal{I}(f)$ be an exclusive set of clauses of f, and $\mathcal{C} \subseteq \mathcal{I}(f)$ be a set of clauses which represents f. The Boolean function $f_{\mathcal{X}}$ represented by the set $\mathcal{C} \cap \mathcal{X}$ is called the *$\mathcal{X}$-component* of the function f. We shall simply call a function g an *exclusive component of f*, if $g = f_{\mathcal{X}}$ for some exclusive subset $\mathcal{X} \subseteq \mathcal{I}(f)$.

Theorem 2.4 guarantees that the $\mathcal{X}$-component $f_{\mathcal{X}}$ is well defined for every exclusive set $\mathcal{X} \subseteq \mathcal{I}(f)$. Theorem 2.4 has the following corollary.

Corollary 2.6 (Boros et al., 2010) *Let $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathcal{I}(f)$ be two distinct sets of clauses such that $\mathcal{C}_1 \equiv \mathcal{C}_2 \equiv f$, i.e. such that both sets represent f, and let $\mathcal{X} \subseteq \mathcal{I}(f)$ be an exclusive set of clauses. Then $(\mathcal{C}_1 \setminus \mathcal{X}) \cup (\mathcal{C}_2 \cap \mathcal{X})$ also represents f.*

2.4 Autarkies

Autark assignments were introduced by Monien and Speckenmeyer (1985) and they are defined as follows.

Definition 2.7 Let ψ be a CNF on the set V of variables, let $Y \subseteq V$ be a subset of variables, let $L = \{x \mid x \in Y\} \cup \{\overline{x} \mid x \in Y\}$ be the corresponding set of literals, and let $t : Y \mapsto \{0, 1\}$ be a partial assignment on ψ. Then t is an *autarky* on ψ if for every clause $C \in \psi$ either $C \cap L = \emptyset$ or C is satisfied by t.

An autarky is a special type of partial assignment which satisfies every clause in which it substitutes a value for some literal. We shall prove two simple lemmas about autarkies which will be needed later in this paper. The first lemma was (in a different notation) shown by Kullmann (2000) as Lemma 3.13, but we will give a short proof here as well to make this paper self-contained.

Lemma 2.8 *Let ψ be a CNF on the set V of variables which represents a function f. Let $Y \subseteq V$ be a subset of variables, and let $t : Y \mapsto \{0, 1\}$ be an autarky on ψ. Then t is an autarky on $\mathcal{I}(f)$.*

Proof : Since all clauses in $\mathcal{I}(f)$ can be derived from ψ by resolution, it will suffice to show that resolution preserves the autarky properties, namely that if the parent clauses are satisfied by t whenever they contain a literal from $L = \{x \mid x \in Y\} \cup \{\overline{x} \mid x \in Y\}$, then so does the resolvent. Let $C, C_1, C_2 \in \mathcal{I}(f)$ be clauses such that $C = R(C_1, C_2)$. Let $\ell \in L$ be a literal in C. If ℓ is satisfied by t we are done, so let us asssume that ℓ is not satisfied by t. Clause C inherited ℓ from one of its parent clauses so let us assume without loss of generality $\ell \in C_1$. By the autarky property, C_1 must be now satisfied by t, so it must contain another literal $\ell' \in L$ satisfied by t. Now there are two possibilities: either $\ell' \in C$ (clause C inherited both ℓ and ℓ' from C_1) in which case C is satisfied by t and we are done, or $\ell' \notin C$ which means that $R(C_1, C_2)$ resolves over ℓ'. That implies $\overline{\ell'} \in C_2$, i.e. C_2 contains a literal from L not satisfied by t. Thus, by the autarky property, C_2 must be satisfied by t, so it must contain another literal $\ell'' \in L$ satisfied by t. However, in this case C inherits ℓ'' from C_2, which finishes the proof. ■

Corollary 2.9 *Let ψ be a CNF on the set V of variables which represents a function f. Let $Y \subseteq V$ be a subset of variables, and let $t : Y \mapsto \{0, 1\}$ be an autarky on ψ. Let $\varphi \subseteq \mathcal{I}(f)$ be an arbitrary representation of f. Then t is an autarky on φ.*

Let us mention that Corollary 2.9 would not hold without the assumption $\varphi \subseteq \mathcal{I}(f)$. Consider e.g. a CNF $\psi = (x \vee y) \wedge z$ and a CNF $\varphi = (x \vee y) \wedge (z \vee y) \wedge (z \vee \overline{y})$. It should be obvious that $\psi \equiv \varphi$, i.e. both CNFs represent the same function f, but $\varphi \not\subseteq \mathcal{I}(f)$. Now if t

is an assignment which sets y to 1, t is autarky on ψ. On the other hand t is not autarky on φ because $(z \vee \overline{y})$ is not satisfied by t.

Lemma 2.10 *Let ψ be a CNF on the set V of variables, let $Y \subseteq V$ be a subset of variables, let $L = \{x \mid x \in Y\} \cup \{\overline{x} \mid x \in Y\}$ be the corresponding set of literals, and let $t : Y \mapsto \{0, 1\}$ be an autarky on ψ. Then $\psi(t)$ represents an exclusive component $f_{\mathcal{X}}$ of f defined by the exclusive set of clauses*

$$\mathcal{X} = \{C \in \mathcal{I}(f) \mid C \cap L = \emptyset\}.$$

Proof : It suffices to show that $\mathcal{X}$ is an exclusive subset of $\mathcal{I}(f)$ Let $C, C_1, C_2 \in \mathcal{I}(f)$ be clauses such that $C = R(C_1, C_2)$ and $C \in \mathcal{X}$. Let us assume by contradiction that one of the parent clauses, say C_1, does not belong to $\mathcal{X}$, which means that there exists $\ell \in C_1 \cap L$. Since $\ell \notin C$, $R(C_1, C_2)$ must resolve over ℓ, which implies $\overline{\ell} \in C_2$. However, one of $\ell, \overline{\ell}$ is not satisfied by t, so the corresponding clause (one of C_1, C_2) must contain some other literal $\ell' \in L$ which is satisfied by t because t is an autarky. But now we get $\ell' \in C$ contradicting the assumption $C \in \mathcal{X}$.

Since t is an autarky on ψ it follows that $\psi(t) = \psi \cap \mathcal{X}$ and since $\psi \equiv \mathcal{I}(f)$ and $\mathcal{X}$ is an exclusive set of implicates of f, it follows by Theorem 2.4 that $\psi(t) = \psi \cap \mathcal{X} \equiv \mathcal{I}(f) \cap \mathcal{X}$ defines an exclusive compontent $f_{\mathcal{X}}$. ■

2.5 Matched Formulas

In this subsection we shall define the key concept of this paper. This concept is based on graph properties, so to this end we shall use standard graph terminology (e.g., see Bollobás, 1998). Given an undirected graph $G = (V, E)$, a subset of edges $M \subseteq E$ is a *matching* in G if the edges in M are pairwise disjoint. A *bipartite graph* $G = (A, B, E)$ is an undirected graph with disjoint sets of vertices A and B, and the set of edges E satisfying $E \subseteq A \times B$. For a set W of vertices of G, let $\Gamma(W)$ denote the *neighbourhood* of W in G, i.e. the set of all vertices adjacent to some element of W. We shall use the following well-known result on matchings in bipartite graphs:

Theorem 2.11 (Hall's Theorem – Hall, 1935; Lovász & Plummer, 1986) *Let $G = (A, B, E)$ be a bipartite graph. A matching M of size $|M| = |A|$ exists if and only if for every subset S of A we have that $|S| \leq |\Gamma(S)|$.*

Now we are ready to define matched formulas.

Definition 2.12 Let $\varphi = C_1 \wedge \ldots \wedge C_m$ be a CNF on n variables $X = \{x_1, \ldots, x_n\}$. We shall associate a bipartite graph $I(\varphi) = (\varphi, X, E)$ with φ (also called the incidence graph of φ), where the vertices correspond to clauses in CNF φ and the variables X. A clause C_i is connected to a variable x_j (i.e. $\{C_i, x_j\} \in E$) if C_i contains x_j or $\overline{x_j}$. A CNF φ is *matched* if $I(\varphi)$ has a matching of size m, i.e. if there is a matching which pairs each clause with a unique variable.

Note that a matching of maximum size in a given graph can be found in polynomial time (e.g., Lovász & Plummer, 1986) and thus we can test in polynomial time whether given CNF is matched. The fact that a clause C_i is matched to a variable x_j in a matching M will be denoted as $\{C_i, x_j\} \in M$. A variable which is matched to some clause in matching

M is called *matched* in M, it is *free* in M otherwise. Note, that a matched CNF is trivially satisfiable. If a clause C_i is matched to variable x_j, then we can simply assign x_j a value which will satisfy C_i. The name "matched" was given to these formulas by Franco and Van Gelder (2003), although they were already considered by Aharoni and Linial (1986), and Tovey (1984).

3. Equivalence Testing and Hardness of Clause Minimization of Matched Formulas

Following a definition given by Čepek, Kučera, and Savický (2012) a class of CNFs $\mathcal{X}$ is called *tractable* if it satisfies the following four properties.

- *Recognition:* Given an arbitrary CNF φ it is possible to decide in polynomial time with respect to the size of φ whether $\varphi \in \mathcal{X}$.
- *Satisfiability:* Given an arbitrary CNF $\varphi \in \mathcal{X}$ it is possible to decide in polynomial time with respect to the size of φ whether φ is satisfiable.
- *Partial assignment:* Given an arbitrary CNF $\varphi \in \mathcal{X}$, if ψ is produced from φ by fixing some variables to 0 or 1 and substituting these values into φ, then $\psi \in \mathcal{X}$.
- *Prime representations:* Given an arbitrary CNF $\varphi \in \mathcal{X}$, if φ represents a function f then all prime CNF representations of f belong to $\mathcal{X}$.

It was shown by Čepek et al. (2012) that given two CNFs from a tractable class, it can be tested in polynomial time, whether these two CNFs are logically equivalent or not. The class of matched CNFs clearly satisfies the first two tractability conditions, but fails to satisfy the remaining two. Costructing a counterexample to the third property is easy. The CNF

$$(x \vee y \vee z) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee \overline{z})$$

is clearly matched, but a partial assignment $x \leftarrow 0$ creates a CNF

$$(y \vee z) \wedge (\overline{y} \vee z) \wedge (y \vee \overline{z})$$

that is not matched. We defer the counterexample to the fourth property to the next section. In the light of these findings it is an interesting question what is the complexity of equivalence testing for matched CNFs. Despite the fact that satisfiability is trivial for matched CNFs, equivalence testing is co-NP-complete.

Matched equivalence	
Instance :	Two matched CNFs φ and ψ
Question :	Is $\varphi \equiv \psi$?

Theorem 3.1 *The problem* Matched equivalence *is co-NP-complete.*

Proof : A nondeterministic polynomial procedure checking that the two CNFs are not equivalent simply guesses an assignment t and checks whether $\varphi(t) \neq \psi(t)$. The problem Matched equivalence is thus in co-NP.

To show co-NP-hardness we reduce from the problem of checking that a given CNF α is unsatisfiable (this problem is a prominent example of a co-NP-complete problem as its complement is the satisfiability problem, e.g., see Garey & Johnson, 1979). Let α be an arbitrary CNF on n variables and m clauses, in particular let $\alpha = C_1 \wedge C_2 \wedge \ldots \wedge C_m$. Let us define a clause $D = (a_1 \vee a_2 \vee \ldots \vee a_m)$ on m new variables not occuring in α. Now let us define two CNFs:

$$\begin{aligned} \varphi &= (C_1 \vee D) \wedge (C_2 \vee D) \wedge \ldots \wedge (C_m \vee D) \\ \psi &= D \end{aligned}$$

Both φ and ψ are obviously matched, since each clause $C'_i = (C_i \vee D)$ can be matched to a variable a_i. Now we have that $\varphi \equiv \psi$ iff $\alpha \equiv \perp$, i.e. iff α is unsatisfiable. This follows directly from the fact that $\varphi \equiv \alpha \vee D \equiv \alpha \vee \psi$. We have reduced a co-NP-complete problem of unsatisfiability to the problem Matched Equivalence and thus the problem Matched Equivalence is co-NP-complete as well. ■

The fact, that equivalence testing is co-NP-hard, is probably the principal reason behind the fact proved by Gurský (2011) that clause minimization of matched CNFs is Σ_2^p-complete. The proof by Gurský (2011) basically follows the proof of Σ_2^p-completeness of general CNFs presented by Umans (1999), and Buchfuhrer and Umans (2011) and is quite long and complicated. Here we present a much shorter and simpler proof based on a similar idea as the proof for the hardness of equivalence testing.

Matched minimization	
Instance :	A matched CNF φ and an integer k
Question :	Is there a CNF ψ equivalent to φ with at most k clauses?

Theorem 3.2 *The problem* Matched minimization *is Σ_2^p-complete.*

Proof : Since Matched Minimization is a special case of Boolean minimization, and it is known that Boolean minimization is Σ_2^p-complete, Matched Minimization must be in Σ_2^p. To see that this problem is Σ_2^p-hard, we reduce the Σ_2^p-complete Boolean minimization to it.

Let (α, k) be an instance of Boolean minimization where $\alpha = C_1 \wedge C_2 \wedge \ldots \wedge C_m$. Now let us repeat the construction from the proof of Theorem 3.1. Let $a_1, a_2, \ldots, a_m$ be new variables that do not occur in α. Let then φ be a matched CNF defined by $\alpha \vee (a_1 \vee a_2 \vee \ldots \vee a_m)$, that is $\varphi = C'_1 \wedge C'_2 \wedge \ldots \wedge C'_m$ where $C'_i = (C_i \vee a_1 \vee a_2 \vee \ldots \vee a_m)$ for $1 \leq i \leq m$. The instance of Matched minimization is now (φ, k).

Let (α, k) be a positive instance of Boolean minimization. Then there exists CNF $\beta = D_1 \wedge D_2 \wedge \ldots \wedge D_{k'}$ (with $k' \leq k$) that is equivalent to α. Let ψ be a CNF equivalent to $\beta \vee (a_1 \vee a_2 \vee \ldots \vee a_m)$ that is let $\psi = D'_1 \wedge D'_2 \wedge \ldots \wedge D'_{k'}$ where $D'_i = D_i \vee a_1 \vee a_2 \vee \ldots \vee a_m$

for $1 \leq i \leq k'$. Clearly ψ is equivalent to φ and has at most k clauses. Therefore (φ, k) is a positive instance of Matched minimization.

To see the other direction let (φ, k) be a positive instance of Matched minimization and let ψ be a CNF equivalent to φ with at most k clauses. Let β be a CNF originating from ψ by a partial assignment that sets all a-variables to zero and sets no other variable. Since ψ is equivalent to φ and that is equivalent to $\alpha \vee (a_1 \vee a_2 \vee \ldots \vee a_m)$, we have that β is equivalent to α. Clearly $|\beta| \leq |\psi| \leq k$ and since β is equivalent to α we conclude that (α, k) is a positive instance of Boolean minimization. ∎

Remark 3.3 *Since all occurrences of a-variables in φ in the proof of Theorem 3.2 were positive, every resolution from φ keeps all a-variables in every derived clause. We can assume that ψ is prime and therefore every clause of ψ also contains all a-variables positively and thus in fact β has the same number of clauses as ψ.*

4. Prime Representations of Matched Formulas

It is not difficult to see that unlike some well-behaved classes of CNFs (such as e.g. Horn CNFs or quadratic CNFs) for which all prime and irredundant CNFs lie inside the class, this is not the case for matched CNFs. Consider the CNF

$$(\overline{a} \vee b) \wedge (\overline{b} \vee c) \wedge (\overline{c} \vee a)$$

which is matched and a logically equivalent CNF

$$(\overline{a} \vee b) \wedge (\overline{b} \vee a) \wedge (\overline{c} \vee b) \wedge (\overline{b} \vee c)$$

which is not matched despite being prime and irredundant. Thus it is a legitimate question, whether given a (nonprime) matched CNF, there exists at least one logically equivalent prime and irredundant CNF which is also matched. In the rest of this section we will prove an affirmative answer to this question. Let us start with a simple but useful observation.

Observation 4.1 *Let $\varphi = C_1 \wedge \ldots \wedge C_m$ be a matched CNF and let C be a clause derived by a regular tree-like resolution derivation T from φ. Let $C = R(D_1, D_2)$, where $D_1 = (A_1 \vee z)$ and $D_2 = (A_2 \vee \overline{z})$, i.e. the resolution is over z. Let T_1 denote the subtree of T rooted at D_1, and let T_2 denote the subtree of T rooted at D_2. If $C_i \in \varphi$, $i \in \{1, \ldots, m\}$ is a leaf clause in both T_1 and T_2, then C_i contains neither z nor $\overline{z}$ and thus it cannot be matched to z in any matching for φ.*

Proof : Since T is regular, the only resolution over z in T is $C = R(D_1, D_2)$. Thus there can be no $\overline{z}$ in T_1 and no z in T_2, since then they would be in D_1 and D_2 respectively as well. Thus C_i which is in both T_1 and T_2 cannot contain z at all. ∎

The following lemma will allow us to exchange one free variable with a matched variable in any matching for a given matched CNF φ.

Lemma 4.2 *Let $\varphi = C_1 \wedge \ldots \wedge C_m$ be a matched CNF and let M be a matching for φ. Let D be a clause derived by a regular tree-like resolution T from φ. Let $x \in D$ be a variable which is free in M, then there is a variable $y \in D$ matched in M and a matching M' for φ which satisfy the following property: If X denotes the set of variables which are matched in M, and X' denotes the set of variables matched in M', then $X' = (X \setminus \{y\}) \cup \{x\}$.*

Proof : We shall proceed by induction on the depth of the regular tree-like resolution proof T of D. If D is in φ, then we set y to be the variable matched with D in M and

$$M' := \Big(M \setminus \Big\{\{D, y\}\Big\}\Big) \cup \Big\{\{D, x\}\Big\}.$$

Now let us assume that $D = R(D_1, D_2)$, where D_1 and D_2 are either clauses in φ, or they are derived by a regular tree-like resolution from φ. Suppose the resolution is over a variable z and let us denote $D_1 = (A_1 \vee z)$, $D_2 = (A_2 \vee \overline{z})$. For $i = 1, 2$ let T_i denote the subtree rooted at D_i, L_i the set of leaf clauses in T_i, φ_i the sub-CNF of φ formed by clauses in L_i, M_i the sub-matching of M on clauses from φ_i, and X_i the set of variables matched in M_i.

First let us assume that variable $x \in A_1 \setminus A_2$. Let us without loss of generality assume that x appears positively in D_1 (otherwise we can switch the polarity of x in φ), thus $D_1 = (A'_1 \vee x \vee z)$, where $A'_1 = A_1 \setminus \{x\}$. Now we can use the induction hypothesis on the subtree T_1 rooted at D_1, and matching M_1 on φ_1. It follows that there is a matching M'_1 on φ_1 in which x is a matched variable and there is a variable y_1 in D_1 which is matched in M_1 but free in M'_1. Moreover if X'_1 denotes the set of variables matched in M'_1, then $X'_1 = (X_1 \setminus \{y_1\}) \cup \{x\}$. We can extend the matching M'_1 to the whole φ by adding pairs matching clauses in $L_2 \setminus L_1$ to variables in $X_2 \setminus X_1$ as in M, let the new matching be denoted M''. If $y_1 \neq z$, then y_1 occurs in D and we can set $M' = M''$.

If $y_1 = z$, then it is a free variable in M''. Let M''_2 denote the sub-matching of M'' on clauses of φ_2. Now we can use induction hypothesis on φ_2, its resolvent D_2, and variable $y_1 = z$ (playing role of the free variable). By this we find a matching M^*_2 for φ_2 and a variable y in D_2 such that y_1 is matched in M^*_2, y is free in M^*_2 while it was matched in M''_2. In particular if X^*_2 denotes the set of matched variables in M^*_2 and X''_2 denotes the set of variables matched in M''_2, then $X^*_2 = (X''_2 \setminus \{y\}) \cup \{y_1\}$. We can now extend the matching M^*_2 to the whole formula φ by adding pairs matching clauses in $L_1 \setminus L_2$ to variables $X''_1 \setminus X''_2$ as in M'' (here X''_1 denotes the set of variables matched in M'' to leaves of T_1). In this way we obtain the desired matching M'. It is clear that $y \neq z$ and thus $y \in D$. Moreover $X' = (X \setminus \{y\}) \cup \{x\}$.

Now, let us assume $x \in A_1 \cap A_2$. By Observation 4.1 we have that $z \notin X_1 \cap X_2$. Let $i \in \{1, 2\}$ be such that $z \notin X_i$. Now let us use induction hypothesis on φ_i and variable x. We get a matching M'_i for formula φ_i which can be extended to the whole φ by adding the pairs matching clauses in $L_{3-i} \setminus L_i$ to variables in $X_{3-i} \setminus X_i$ as in M (the extended matching is then the desired matching M'). We also get a variable y which is matched in M_i, but which is free in M'_i. Since $y \in X_i$, we get that $y \neq z$ and thus y is in D. Clearly $X' = (X \setminus \{y\}) \cup \{x\}$. ■

Lemma 4.3 *Let $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be a matched CNF and let us assume that an implicate D is derived by a regular tree-like resolution derivation T from φ in which C_1 is used, then $\varphi' = D \wedge C_2 \wedge \ldots \wedge C_m$ is a matched CNF.*

Proof : The fact that C_1 is used in this resolution derivation implies that C_1 is a leaf clause in T. We shall proceed by induction on the depth of T. Let M be a matching for φ and let X denote the set of variables, which are matched in M. We shall preserve the following invariant:

(*) If M' is a matching for φ' constructed by the proof and X' denotes the matched variables in M', then $X' = X$.

Let us at first assume that $D = C_1$. Then the proposition trivially follows and invariant (*) is satisfied. Now let us suppose that $D = R(C_1, C_j)$, where $j \in \{2, \dots, m\}$. Let y be the variable which is matched to C_1 in M, i.e. $\{C_1, y\} \in M$. If $y \in D$, then we set

$$M' = \Big(M \setminus \Big\{\{C_1, y\}\Big\}\Big) \cup \Big\{\{D, y\}\Big\}.$$

If $y \notin D$, then it follows that C_1 and C_j resolve over y. Let us without loss of generality assume that y appears positively in C_1 and let us denote $C_1 = (A_1 \vee y)$ and $C_j = (A_j \vee \overline{y})$, hence $D = A_1 \vee A_j$. Then C_j is matched with another variable $z \in A_j$ in M, thus we can set

$$M' = \Big(M \setminus \Big\{\{C_1, y\}, \{C_j, z\}\Big\}\Big) \cup \Big\{\{C_j, y\}, \{D, z\}\Big\}.$$

Then M' is a matching for φ' with $X' = X$.

Now let us assume that $D = R(D_1, D_2)$ where D_1 and D_2 are themselves derived by resolution derivation from φ, or they belong to φ. Suppose the resolution is over a variable z and let us denote $D_1 = (A_1 \vee z)$, $D_2 = (A_2 \vee \overline{z})$. For $i = 1, 2$ let T_i denote the subtree of T rooted at D_i, L_i the set of leaf clauses in T_i, φ_i the sub-CNF of φ formed by clauses in L_i, let M_i be the sub-matching of M on clauses from φ_i, and X_i the set of variables matched in M_i.

Let us at first assume that $C_1 \in L_1 \cap L_2$. By Observation 4.1 we get that $z \notin X_1 \cap X_2$. Let $i \in \{1, 2\}$ be such that $z \notin X_i$. Let us use the induction hypothesis on T_i and D_i to find a matching for a formula φ'_i which is a CNF formed by clauses $(L_i \setminus \{C_1\}) \cup \{D_i\}$. By the induction hypothesis φ'_i is a matched formula, let M'_i be a matching constructed for φ'_i which satisfies invariant (*), i.e. the set of matched variables in M'_i is X_i. Let x be the variable matched with D_i in M'_i. Now since $z \notin X_i$ we have that $x \in A_i$, thus x belongs to D as well. We can now construct a matching M' for φ' by extending M'_i with pairs matching variables in $X_{3-i} \setminus X_i$ to clauses in $L_{3-i} \setminus L_i$ as in M. Moreover we replace the pair $\{D_i, x\}$ with the pair $\{D, x\}$. The result M' is a matching for φ' in which exactly the variables in X are matched and thus M' satisfies invariant (*).

In the rest of the proof we shall assume that $C_1 \in L_1 \setminus L_2$. We can use induction hypothesis on T_1 and D_1 to find a matching M'_1 for a formula φ'_1 which is again a CNF formed by clauses $(L_1 \setminus \{C_1\}) \cup \{D_1\}$. By induction hypothesis M'_1 satisfies invariant (*) and thus the matched variables in M'_1 are exactly the variables in set X_1. Let x be the variable to which clause D_1 is matched in M'_1. Now there are two cases to consider .

1. If $x \in A_1$, then we can construct a matching M' for the whole formula φ' by extending M'_1 with pairs matching clauses in $L_2 \setminus L_1$ to variables in $X_2 \setminus X_1$ as in M, and we replace pair $\{D_1, x\}$ with pair $\{D, x\}$. Then M' is a matching which again matches variables in X and thus it also satisfies invariant (*).

2. If $x = z$, the situation is more complicated. In this case we can observe that $z \in X_1 \setminus X_2$ ($z \notin X_1 \cap X_2$ by Observation 4.1, on the other hand z is matched in M_1 and M'_1). Let M'_2 be a matching for the sub-CNF φ_2 formed by clauses in L_2 which is constructed

as follows. The clauses in $L_1 \cap L_2$ are matched to the same variables as in M'_1, the clauses in $L_2 \setminus L_1$ are matched to the same variables as in M. Note that M'_2 formed in this way is really a matching, in particular C_1 does not belong to L_2 and thus it does not matter that it is not matched to any variable in M'_1. Moreover, if X'_2 denotes the set of variables matched in M'_2, then each variable in X'_2 is matched to exactly one clause. This is because M'_1 did not change anything on clauses in $L_2 \setminus L_1$ and it did not match any of the variables in $X_2 \setminus X_1$. Note, that X'_2 is not necessarily equal to X_2, because M'_1 is allowed to use variables from $X_1 \setminus X_2$ for the clauses in $L_1 \cap L_2$, but we have that $X = X'_2 \cup X_1$. We also have that z is a free variable in M'_2, that is because $z \in X_1 \setminus X_2$, thus it is not matched to any clause in L_2 in M, and because z is matched to D_1 in M'_1, thus z is not matched to any clause in $L_1 \cap L_2$ in M'_1.

Now we have the following situation: We have a formula φ_2, we have a clause D_2 which is derived by a regular tree-like resolution T_2 from φ_2. We have a matching M'_2 for φ_2 which matches the variables in set X'_2. We have a variable z which is free in M'_2, thus we can use Lemma 4.2 to find another matching M''_2 for φ_2 and a variable y in D_2 such that z is matched to some clause in M''_2, and y is now free in M''_2 while it was matched in M'_2. Moreover, if X''_2 denotes the set of variables matched in M''_2, then $X''_2 = (X'_2 \setminus \{z\}) \cup \{y\}$. Necessarily $z \neq y$, and thus $y \in A_2$. Now we are ready to form desired matching M' for φ.

(a) A clause C from $L_1 \setminus (L_2 \cup \{C_1\})$ is matched to variable a in M' such that $\{C, a\} \in M'_1$.

(b) A clause C from L_2 is matched to a variable a in M' such that $\{C, a\} \in M''_2$.

(c) A clause D is matched to the variable y in M'.

M' defined in this way is indeed a matching, in particular if $C \in L_1 \setminus (L_2 \cup \{C_1\})$, then the matched variable a belongs to $X_1 \setminus (X'_2 \cup \{z\})$, and thus a is free in both M'_2 and M''_2, and it still can be used for C. Let us also observe that M' preserves invariant (*), in particular if X' denotes the set of variables matched in M', then $X' = X$. Let a be a variable from X', we shall show that $a \in X$ as well, because $|X'| = |X|$, it follows that $X' = X$.

(a) If a is matched to a clause $C \in L_1 \setminus (L_2 \cup \{C_1\})$, then this is because $\{C, a\} \in M'_1$ since by induction hypothesis invariant (*) is preserved for M'_1 we have that $a \in X_1$ and thus $a \in X$.

(b) If a is matched to a clause $C \in L_2$, then this is because $\{C, a\} \in M''_2$, then $a \in X'' = (X'_2 \setminus \{y\}) \cup \{z\}$. We know that $z \in X$ and that $X'_2 \subseteq X$.

(c) If a is matched to D, then $a = y \in X'_2 \subseteq X$.

Together we have that invariant (*) is satisfied for M'.

In any case we found a matching M' for CNF φ' which satisfies invariant (*), and the proof is finished. ∎

Theorem 4.4 *Let φ be a matched CNF representing function f, then there is a prime and irredundant representation of f which is also matched.*

Proof : This follows from Lemma 4.3. Firstly, we can drop any redundant clauses from φ without spoiling its matched property. If φ' is an irredundant representation of f originated from φ by dropping these redundant clauses, then we can turn it into a prime representation by using Lemma 4.3 as follows. If $\varphi' = C_1 \wedge \ldots \wedge C_m$, and $C_1' \subsetneq C_1$ is a prime implicate, then C_1' can be derived by a resolution derivation from φ'. Since φ' is already irredundant, in every resolution derivation of C_1' from φ' we have to use C_1. Thus by Lemma 4.3, the formula $\varphi'' = C_1' \wedge C_2 \wedge \ldots \wedge C_m$ is also matched. In this way we can replace every clause in φ' by a prime subimplicate. Thus we obtain a prime and irredundant representation of f. ■

5. Minimum Representations of Matched Formulas

In the previous sections we have seen that for a matched CNF there may be some logically equivalent prime and irredundant CNFs which are not matched but always at least one such CNF is matched. In this section we shall show that a stronger statement holds for CNFs which are not only prime and irredundant but also clause minimum. We shall prove that if a Boolean function f admits a matched CNF representation, then every clause minimum CNF representation of f is a matched CNF.

Theorem 5.1 *Let φ be a matched CNF representing function f on the set of variables V and let ψ be a clause minimum CNF representation of f. Then ψ is a matched CNF.*

Proof : Due to Theorem 4.4 we may assume that φ is prime and irredundant and thus $\varphi \subseteq \mathcal{I}(f)$. Let us assume by contradiction that ψ is not matched and that $\psi_X \subseteq \psi$ is a maximal (under inclusion) sub-CNF violating Hall's condition (such a subset must exist due to Theorem 2.11). Let us denote X the set of variables in the sub-CNF ψ_X, $Y = V \setminus X$ the set of remaining variables in ψ, and $\psi_Y = \psi \setminus \psi_X$ the remaining clauses in ψ (note that clauses in ψ_Y may contain variables not only from Y but also from X). Now the following holds:

- By the violation of Hall's condition we have $|\psi_X| > |X|$.
- By the maximality of ψ_X there exists a matching M of all clauses in ψ_Y to variables in Y, i.e. ψ_Y is a matched CNF even if we drop all variables in X from its clauses. This follows from the fact that every subset of ψ_Y must satisfy Hall's condition even with respect to the variables in Y, since otherwise any such violating subset could be added to ψ_X contradicting its maximality.

The existence of matching M implies that ψ_Y can be satisfied using only variables from Y (each clause can be satisfied by its matched variable). So let $t : Y \mapsto \{0, 1\}$ be some partial assignment satisfying all clauses in ψ_Y (t is not necessarily unique). Clearly, t is an autarky on ψ as it satisfies every clause containing an assigned literal.

It follows from Lemma 2.10 that $\psi(t) = \psi_X$ represents an exclusive component $f_{\mathcal{X}}$ of f defined by the exclusive set $\mathcal{X} \subseteq \mathcal{I}(f)$, which contains all clauses consisting only of variables from X. Since φ also represents f and we assumed $\varphi \subseteq \mathcal{I}(f)$, it follows from Corollary 2.9 that t is an autarky also on φ. Thus, similarly as for $\psi(t)$ above, we can conclude that $\varphi(t)$ is a sub-CNF of φ which represents the exclusive component $f_{\mathcal{X}}$ of f, i.e. $\psi(t) \equiv \varphi(t)$.

However, φ is matched, so every its sub-CNF (and in particular $\varphi(t)$) is matched, and thus $|\varphi(t)| \leq |X|$ while $|\psi(t)| = |\psi_X| > |X|$. But now, since both $\varphi(t)$ and $\psi(t) = \psi_X$ represent an exclusive component of f, also CNF $\psi' = (\psi \setminus \psi(t)) \cup \varphi(t)$ represents f by Corollary 2.6. However, we get $|\psi'| < |\psi|$ contradicting the assumed minimality of ψ. ■

6. Conclusions

In this paper we study the class of matched CNFs which is an important class of formulas in the theory of parametrized SAT algorithms. We focus on clause minimum CNF representations of Boolean functions which can be represented by matched CNFs. The results presented in this paper are of two types:

1. Complexity results. We show that testing logical equivalence of two matched CNFs is co-NP-complete. Then we use a similar construction to prove that Boolean minimization for matched CNFs is Σ_2^p-complete. This is an already known fact, but the presented proof is much shorter and simpler than the proof by Gurský (2011). Both results appear in Section 3.

2. Structural results. We prove that given a (non-prime) matched CNF representing function f, there may be some prime representations of f which are not matched, but there exists at least one prime representation of f which is matched. Furthermore we prove that in such a case all clause minimum CNFs of f are guaranteed to be matched. The latter result of course implies the former, however, the proof of the latter result uses the former one. Thus both subsequently appear in the text as the main results of Sections 4 and 5.

An interesting question for future research is whether the structural results from Sections 4 and 5 can be extended in some way from matched CNFs, i.e. CNFs with maximum deficiency zero, to CNFs with maximum deficiency bounded by a constant (see Section 1 for the definition of maximum deficiency).

In our paper we use the number of clauses in a CNF as a measure of optimality. It is an interesting question whether our result would hold if the total length of the formula (i.e. the total number of literal occurrences in a formula) would be considered.

Acknowledgments

The second author gratefully acknowledges the support of the Charles University Grant Agency (grant No. 1390213).

References

Aharoni, R., & Linial, N. (1986). Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *Journal of Combinatorial Theory, Series A*, *43*(2), 196 – 204.

Ausiello, G., D'Atri, A., & Sacca, D. (1986). Minimal representation of directed hypergraphs. *SIAM Journal on Computing*, *15*(2), 418–431.

Bollobás, B. (1998). *Modern Graph Theory*, Vol. 184 of *Graduate Texts in Mathematics*. Springer.

Boros, E., & Čepek, O. (1994). On the complexity of Horn minimization. Tech. rep. 1-94, RUTCOR Research Report RRR, Rutgers University, New Brunswick, NJ.

Boros, E., Čepek, O., & Kogan, A. (1998). Horn minimization by iterative decomposition. *Annals of Mathematics and Artificial Intelligence*, *23*, 321 – 343.

Boros, E., Čepek, O., Kogan, A., & Kučera, P. (2009). A subclass of Horn CNFs optimally compressible in polynomial time. *Annals of Mathematics and Artificial Intelligence*, *57*, 249–291.

Boros, E., Čepek, O., Kogan, A., & Kučera, P. (2010). Exclusive and essential sets of implicates of boolean functions. *Discrete Applied Mathematics*, *158*(2), 81 – 96.

Buchfuhrer, D., & Umans, C. (2011). The complexity of boolean formula minimization. *Journal of Computer and System Sciences*, *77*(1), 142 – 153.

Büning, H. K., & Lettmann, T. (1999). *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, New York, NY, USA.

Čepek, O. (1995). *Structural Properties and Minimization of Horn Boolean Functions*. Ph.D. dissertation, Rutgers University, New Brunswick, NJ, October 1995.

Čepek, O., Kučera, P., & Savický, P. (2012). Boolean functions with a simple certificate for CNF complexity. *Discrete Applied Mathematics*, *160*(4-5), 365 – 382.

Fleischner, H., Kullmann, O., & Szeider, S. (2002). Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theoretical Computer Science*, *289*(1), 503 – 516.

Flum, J., & Grohe, M. (2006). *Parameterized complexity theory*, Vol. 3. Springer.

Franco, J., & Van Gelder, A. (2003). A perspective on certain polynomial-time solvable classes of satisfiability. *Discrete Appl. Math.*, *125*(2-3), 177–214.

Garey, M., & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco.

Genesereth, M., & Nilsson, N. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA.

Gurský, Š. (2011). Minimization of matched formulas. In Šafránková, J., & Pavlů, J. (Eds.), *WDS'11 Proceedings of Contributed Papers: Part I – Mathematics and Computer Science*, pp. 101–105, Prague. Matfyzpress.

Hall, P. (1935). On representatives of subsets. *Journal of The London Mathematical Society-second Series*, *s1-10*, 26–30.

Hammer, P., & Kogan, A. (1993). Optimal compression of propositional Horn knowledge bases: Complexity and approximation. *Artificial Intelligence*, *64*, 131 – 145.

Hammer, P., & Kogan, A. (1995). Quasi-acyclic propositional Horn knowledge bases: Optimal compression. *IEEE Transactions on Knowledge and Data Engineering*, *7*(5), 751 – 762.

Kullmann, O. (2000). Investigations on autark assignments. *Discrete Applied Mathematics*, *107*(1–3), 99 – 137.

Kullmann, O. (2003). Lean clause-sets: generalizations of minimally unsatisfiable clause-sets. *Discrete Applied Mathematics*, *130*(2), 209 – 249.

Lovász, L., & Plummer, M. D. (1986). *Matching Theory*. North-Holland.

Maier, D. (1980). Minimal covers in the relational database model. *Journal of the ACM*, *27*, 664 – 674.

Monien, B., & Speckenmeyer, E. (1985). Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, *10*(3), 287 – 295.

Szeider, S. (2003). Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. In Warnow, T., & Zhu, B. (Eds.), *Computing and Combinatorics*, Vol. 2697 of *Lecture Notes in Computer Science*, pp. 548–558. Springer Berlin Heidelberg.

Szeider, S. (2005). Generalizations of matched CNF formulas. *Annals of Mathematics and Artificial Intelligence*, *43*(1-4), 223–238.

Szeider, S. (2007). Matched formulas and backdoor sets. In Marques-Silva, J., & Sakallah, K. (Eds.), *Theory and Applications of Satisfiability Testing – SAT 2007*, Vol. 4501 of *Lecture Notes in Computer Science*, pp. 94–99. Springer Berlin Heidelberg.

Tovey, C. A. (1984). A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, *8*(1), 85 – 89.

Tseitin, G. S. (1983). On the complexity of derivation in propositional calculus. In *Automation of Reasoning*, pp. 466–483. Springer.

Umans, C. (2001). The minimum equivalent DNF problem and shortest implicants. *J. Comput. Syst. Sci.*, *63*(4), 597–611.

Umans, C., Villa, T., & Sangiovanni-Vincentelli, A. L. (2006). Complexity of two-level logic minimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, *25*(7), 1230–1246.

Umans, C. M. (1999). Hardness of approximating Σ_2^p minimization problems. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pp. 465–474, Washington, DC, USA. IEEE Computer Society.

Urquhart, A. (1995). The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, *1*(4), pp. 425–467.

Urquhart, A. (2011). The depth of resolution proofs. *Stud. Log.*, *99*(1-3), 349–364.