Proceedings of the International MultiConference of Engineers and Computer Scientists 2017 Vol I,
IMECS 2017, March 15 - 17, 2017, Hong Kong

On the Autarky Structure of Linear CNF's

Stefan Porschen [*]

Abstract—**Autarkies are an important tool in algorithmic CNF-SAT for simplifying formulas by removing sovable substructures. In this paper we investigate some structural aspects of autarkies relying on the concept of variable closures and the fibre-view on CNF formulas.**

Keywords: satisfiability, autarky, linear formula, hypergraph, variable closure

1 Introduction

A fundamental problem in mathematics is the NP versus P problem. The genuine and one of the most important NP-complete problems is the propositional satisfiability problem (SAT) for conjunctive normal form (CNF) formulas [5], thus SAT forms the core of computational complexity theory. SAT also plays a fundamental role in the theory of designing exact algorithms, and it has a wide range of applications because many problems can be encoded as a SAT problem via reduction [10, 9] due to the rich expressiveness of the CNF language. Important areas where SAT plays a vital role are formal verification [19], bounded model checking [4], and artificial intelligence. In industrial applications most often the modelling CNF formulas are of a specific structure. And therefore it would be desirable to have fast algorithms for such instances. The applicational area is pushed by the fact that meanwhile several powerful solvers for SAT have been developed. Also from a theoretical point of view one is interested in classes for which SAT can be solved in polynomial time. There are known several classes, for which SAT can be tested in polynomial time, such as quadratic formulas, Horn formulas, matching formulas, nested formulas etc. [1, 3, 7, 11, 12, 18, 20]. So, the structure of the set of all CNF formulas is of great importance. Specifically from the point of view of algorithmics autarkies are of quite importance. The concept of autark assignments within the context of CNF-SAT has been introduced in [13]. Such assignments satisfy independent parts of a formula preserving its overall satisfiability status. The concept has been successfully used in automated theorem proving and was also subject of theoretical investigations. However there are many questions open concerning the structure of autarkies in a given CNF formula and also concerning their algorithmical usefulness. The present paper is devoted to provide some insight into structural aspects of autark assignments where the case of linear formula classes is considered also. Specifically we are interested in the fraction of unsatisfiable formulas admitting no autarky at all. Moreover the concept of the variable closure is exploited and several complexity issues of computational problems w.r.t. autarky are discussed.

[*]Mathematics Group, Department 4, HTW Berlin, D-10313 Berlin, Germany, Email: porschen@htw-berlin.de.

2 Preliminaries

A Boolean variable x taking values from $\{0,1\}$ induces a positive literal (variable x) or a negative literal (negated variable $\overline{x}$). Let $L(U)$ denote the set of all literals determined by the members of a finite, non-empty set U of Boolean variables. A *clause* c is a non-empty, finite disjunction of different literals, and is represented as a finite, non-empty set $c = \{l_1, \ldots, l_{|c|}\}$. Setting a literal to 1 means to set the corresponding variable accordingly. A *(CNF) formula* C is a non-empty, finite conjunction of different clauses and is considered as a finite, non-empty set of its clauses $C = \{c_1, \ldots, c_{|C|}\}$. Let CNF denote the set of all such non-empty formulas consisting of non-empty clauses, and r-CNF $:= \{C \in \text{CNF} : |c| \leq r\}$, for fixed positive integer r. For a formula C (clause c), by $(V(C)$ $(V(c))$ denote the set of variables occurring in C (c). A clause containing no negated literal is called *monotone*. Let CNF$_+$ denote the collection of all *monotone* clause sets. A prominent concept in hypergraph research is the *linear* hypergraph (V,E) [2]. Here each pair $e_1 \neq e_2 \in E$ of hyperedges by definition fulfills $|e_1 \cap e_2| \leq 1$. We call a hypergraph *exact linear* if for all distinct hyperedges e_1, e_2 one has $|e_1 \cap e_2| = 1$. Obviously monotone formulas can be regarded as hypergraphs in which the hyperedges are the clauses. Thus a generalization of (exact) linear hypergraphs is given by *(exact) linear* CNF formulas, where $C \in$ CNF is called *linear* if $|V(c_1) \cap V(c_2)| \leq 1$ and *exact linear* if $|V(c_1) \cap V(c_2)| = 1$ for all $c_1, c_2 \in C$, $c_1 \neq c_2$ [17]. Let LCNF denote the class of all linear formulas, and XLCNF the collection of all exact linear formulas. For a finite set M, let 2^M denots its powerset. Given $C \in$ CNF, SAT asks whether there is a truth assignment $t : V(C) \to \{0,1\}$ such that there is no $c \in C$ all literals of which are set to 0. If such an assignment exists it is called a *model* of C. Let SAT $\subseteq$ CNF denote the collection of all formulas for which there is a model, and UNSAT $:=$ CNF $\setminus$ SAT. r-SAT considers only instances from r-CNF. Without loss of generality it is assumed throughout that clauses only contain different variables, hence pairs like $x, \bar{x}$ are excluded because such clauses are always satisfied and could be removed. Furthermore the existence of clauses containing exactly one literal is excluded in formulas, because such clauses also could be removed by either satisfying them all or obtaining an unsatisfiable formula, immediately. Exactly those clauses c of a formula $C \in \text{CNF}$ which all have the same variable set $b = V(c) \subseteq V(C)$ yield the *fibre* $C_b = \{c \in C : V(c) = b\}$ of C over b [14]. The *base hypergraph* $\mathcal{H}(C) = (V(C), B(C))$ of C is given by the vertex set $V(C)$ and the hyperedge set $B(C) := \{V(c) : c \in C\} \in \text{CNF}_+$. Conversely, we can start with a fixed arbitrary hypergraph $\mathcal{H} = (V, B)$ serving as a base hypergraph if its vertices $x \in V \neq \emptyset$ are regarded as Boolean variables such that for every $x \in V$ there is a $b \in B \neq \emptyset$ containing x. By $W_b := \{c \subseteq L(V) : V(c) = b\}$ denote the collection of all possible clauses over a fixed $b \in B$, similarly W_V denotes the set of all clauses over V of cardinality $|V|$. Observe that $W_V \in \text{UNSAT}$. To a formula $C \in \text{CNF}$ its *formula graph* G_C has a vertex for each clause, and the vertices c, d are joined by an edge in G_C if $V(c) \cap V(d) \neq \emptyset$. For $C \in \text{CNF}$ and $\emptyset \neq U \subseteq V(C)$, let $C(U) = \{c \in C : V(c) \cap U \neq \emptyset\} \in \text{CNF}$ denote the subformula of all clauses c containing a variable in U, where in case $U = \{x\}$ we write $C(x)$. Let C_U be the substructure called the *U-retraction* of C as introduced in [14] and defined as $C_U := \{c \cap L(U) : c \cap L(U) \neq \emptyset, c \in C\}$. Note that $C_U \in \text{CNF}$ only if it contains no unit clause.

3 (Co-)Autarkies in Linear Formulas

For $C \in \text{CNF}$, as introduced in [13] $\emptyset \neq U \in 2^{V(C)}$ is called an *autark set (of variables)*, if there exists a (partial) truth assignment $\alpha : U \to \{0, 1\}$ satisfying $C(U)$, in which case α is called an *autark assignment or autarky.* Clearly α, U as well as $C(U)$ as above can be identified when speaking of an autarky. Hence, an autarky $C(U)$ can be removed from a formula without affecting its satisfiability status.

Lemma 1 *Let U be an autarky of $C \in \text{CNF}$ then $W_b \neq C_b \subseteq C(U)$ for all $b \in B(C)$ with $b \cap U \neq \emptyset$.*

Proof. Let $b \in B$ with $b \cap U \neq \emptyset$ then $C_b \subseteq C(U)$ and as U is autark implying $C(U) \in \text{SAT}$ it follows $C_b \neq W_b$ for all $b \in B$. $\Box$

Let $\text{AUT} \subseteq \text{CNF}$ denote the collection of all formulas for which an autarky exists. Obviously one has $\text{SAT} \subseteq \text{AUT}$. A CNF formula admitting no autarky is refered to as a *co-autarky*, let the collection of such formulas be denoted as $\text{AUT}' := \text{CNF} \setminus \text{AUT}$ then clearly $\text{AUT}' \subseteq \text{UNSAT}$. Next we state that $\text{AUT}' \neq \emptyset$, moreover $\text{AUT} \setminus \text{SAT} \neq \emptyset$, because $\text{AUT}' \neq \text{UNSAT}$.

Lemma 2 *Let $V \neq \emptyset$ be a finite set of Boolean variables and $C \in \text{CNF}$. Then $W_V \in \text{AUT}'$ and $C \cup W_V \in \text{UNSAT} \setminus \text{AUT}'$ if $C \in \text{SAT}$ with $V(C) \cap V = \emptyset$.*

Proof. For every $\emptyset \neq U \in 2^V$ one directly obtains $W_V(U) = W_V \in \text{UNSAT}$. So $W_V \in \text{AUT}'$ for every V. The second claim follows because $V(C) \neq \emptyset$ is autark for $C \cup W_V \in \text{UNSAT}$. $\Box$

The next result provides a characterization of co-autarkies.

Lemma 3 *$C \in \text{AUT}'$ if and only if $C(x) \in \text{UNSAT}$ for every $x \in V(C)$.*

Proof. Let $C \in \text{AUT}'$ and assume there is $x \in V(C)$ such that $C(x) \in \text{SAT}$ then $U = \{x\}$ is an autark set yielding a contradiction. Reversely let $C \in \text{AUT}$ and let $\emptyset \neq U \in 2^{V(C)}$ be an autarky then there is $x \in U$ such that $C(x) \subseteq C(U) \in \text{SAT}$ verifying the assertion. $\Box$

Regarding Lemma 2 one might guess that formulas in AUT' always contain a complete fibre W_b. But this is not true. The next result shows that even some linear formulas lie in AUT'. Observe that the fibres of a linear formula C have the property that $|C_b| = 1$ for all $b \in B(C)$ because unit clauses are excluded.

Theorem 1 *$\text{XLCNF} \subseteq \text{AUT}$ and $\text{LCNF} \cap \text{AUT}' \neq \emptyset$.*

Proof. According to [17] an exact linear formula without complementary unit clauses is satisfiable thus one has directly $\text{XLCNF} \subseteq \text{AUT}$ because unit clauses are excluded in any case. Regarding the second assertion assume the contrary, i.e., $\text{LCNF} \subseteq \text{AUT}$. According to [17] there exists $C^{(0)} \in \text{LCNF} \cap \text{UNSAT}$ admitting an autark set $U^{(0)} \neq \emptyset$. Clearly $\emptyset \neq C^{(1)} := C^{(0)} \setminus C^{(0)}(U^{(0)}) \in \text{LCNF} \cap \text{UNSAT}$ admitting an autark set $U^{(1)}$ and so on up to an unsatisfiable linear formula $\emptyset \neq C^{(i)} := C^{(i-1)} \setminus C^{(i-1)}(U^{(i-1)}) \in \text{AUT}$, for positive integer i. Since $C^{(0)}$ was finite in the next step $i+1$ either the empty set is obtained which is impossible as $C^{(0)} \in \text{UNSAT}$, or $\emptyset \neq C^{(i+1)} \in \text{AUT}'$ yielding a contradiction and proving the claim. $\Box$

As a direct consequence of the last proof the next result follows.

Corollary 1 *$C \in \text{LCNF} \cap \text{AUT} \cap \text{UNSAT}$ if and only if there is $\emptyset \neq C^{(0)} \in \text{SAT} \cap \text{LCNF}$ and $\emptyset \neq D \in \text{AUT}' \cap \text{LCNF}$ such that $C = C^{(0)} \cup D$.*

Proof. If $C \in \text{AUT} \cap \text{UNSAT}$ is linear then remove iteratively autark subformulas as long as possible; the union of these subformulas defines $C^{(0)} \in \text{SAT}$. Since the remaining formula D is neither the empty set nor lies in AUT we have the only if direction. The reverse direction is clear because the presence of $C^{(0)} \in \text{SAT} \cap \text{LCNF}$ states that $C \in \text{AUT}$ and the presence of $D \in \text{AUT}' \cap \text{LCNF}$ verifies that $C \in \text{UNSAT}$, finally $C \in \text{LCNF}$ follows. $\Box$

Another property of linear formulas w.r.t. autarky is stated next which does not hold for arbitrary members in CNF, because W_V serves as a counterexample. Let $\mathrm{LCNF}_{\neq} := \{C \in \mathrm{LCNF} : \forall x, y \in V(C), x \neq y : C(x) \neq C(y)\}$ and $\mathrm{LCNF}_{=} := \mathrm{LCNF} \setminus \mathrm{LCNF}_{\neq}$.

Theorem 2 *Let $C \in \mathrm{LCNF}$. Then $C \in \mathrm{AUT}'$ implies $C \in \mathrm{LCNF}_{\neq}$.*

Proof. Take $C \in \mathrm{LCNF}$ arbitrarily then as C by definition is non-empty, free of unit clauses and moreover clauses are assumed to be free of complementary literals, $V(C)$ contains at least two variables $x \neq y$. Assume that $C(x) = C(y)$ which is equivalent with $x, y \in V(c)$, for all $c \in C(x) = C(y)$ which because of the linearity of C is equivalent with the existence of exactly one clause $c \in C$ such that $x, y \in V(c)$, and $C(x) = C(y) = \{c\} = C(\{x, y\})$. Therefore $U = \{x, y\}$ is an autark set of C which thus cannot lie in AUT'. $\square$

In view of the last result one has $\mathrm{LCNF}_{\neq} \cap \mathrm{AUT}' \neq \emptyset$ and $\mathrm{LCNF}_{=} \subseteq \mathrm{AUT}$. Note that the reverse direction in the previous theorem does not hold in general as is indicated by $C_0 \cup D \in \mathrm{LCNF}_{\neq} \cap \mathrm{AUT}$ where $C_0 := \{\{xy\}, \{yz\}, \{xz\}\} \in \mathrm{CNF}_{+}$ with clauses represented as literal strings over the variables x, y, z, and $D \in \mathrm{LCNF} \cap \mathrm{AUT}'$ is chosen arbitrarily.

4 Autarkies and Variable Closures

Given $C \in \mathrm{CNF}$, then for every $U \subseteq V(C)$ the set $H_C(U) := \{x \in V(C) : x \notin V(C \setminus C(U))\} \subseteq V(C(U))$ is called the *variable closure* of U as introduced in [14], for short we refer to $H_C(U)$ as the *hull* of U. Clearly $U \subseteq H_C(U)$, moreover one can show that H_C is a finite closure operator on $2^{V(C)}$ [15]. Let $\mathfrak{H}(C) := \{H_C(U) : U \in 2^{V(C)}\}$ denote the image of H_C, i.e., the collection of all hulls of C. Useful properties of the hull concept are stated next.

Lemma 4 *Let $C \in \mathrm{CNF}$ then $C(U) = C(H_C(U))$. Moreover one has:*
(1) $C(U) = C(U')$ if and only if $H_C(U) = H_C(U')$.
(2) $H_C(U)$ is an autarky of $C \in \mathrm{CNF}$ if $U \subseteq V(C)$ is an autarky of C.

Proof. Statements (1) and (2) are shown in the proof of Lemma 3 in [15]. The first claim above follows from (1) and $H_C(H_C(U)) = H_C(U)$ which is the idempotence property of a closure operator. $\square$

The converse of the claim (2) in Lemma 4 does not hold in general: Let $U = \{x, y\}$ and consider $C := \{c \cup \{z\} : c \in W_U\} \cup D$ for any $D \in \mathrm{CNF}$ such that $x, y, z \notin V(D)$. Then $C(U) = \{c \cup \{z\} : c \in W_U\}$ cannot be solved on basis of U but on basis of $H_C(U) = \{x, y, z\}$. Hence $H_C(U)$ is an autarky but not U itself. Relying on the hull concept we have another characterization of AUT', respectively of AUT.

Theorem 3 *$C \in \mathrm{AUT}$ if and only if there is an autark set $U \in \mathfrak{H}(C)$.*

Proof. The if-part is obvious. For the reverse direction assume that $C \in \mathrm{AUT}$ hence there is an autark set $U \in 2^{V(C)}$ thus $H_C(U) \in \mathfrak{H}(C)$ is autark for C by Lemma 4 part (2). $\square$

According to [14] a variable hull is called *free* if it is minimal with respect to the inclusion relation. Observe that in general $V(C(U)) \neq H_C(U)$. Besides clarifying when equality is true here, the next result states several connections between $C(U)$, $H_C(U)$ and C_U.

Theorem 4 *For $C \in \mathrm{CNF}$ and $U \subseteq V(C)$ one has:*
(a) $G_{C(U)}$, $G_{C \setminus C(U)}$ are distinct connected components of G_C if and only if $V(C(U)) = H_C(U)$.
(b) U is an autarky if and only if C_U is satisfiable.
(c) Let $U \in \mathfrak{H}(C)$ then $C_U \subseteq W_U$ if and only if U is free.
(d) $U \in \mathfrak{H}(C)$ if $C_U = C(U)$.

Proof. First, $V(C(U)) = H_C(U)$ is equivalent with $V(C \setminus C(U)) \cap V(C(U)) = \emptyset$ which is the same as that for every clause $c \in C \setminus C(U)$ and every clause $d \in C(U)$ we have $V(c) \cap V(d) = \emptyset$; meaning that there exists no egde between the vertices in $G_{C(U)}$ and the vertices in $G_{C \setminus C(U)}$. So, (a) is proven. Obviously $C_U = \{c \cap L(U) : c \in C(U)\}$, therefore if C_U is satisfiable there is $\alpha : U \to \{0, 1\}$ satisfying C_U and also $C(U)$ hence U is an autarky of C. Conversely, if U is an autarky then specifically $C(U) \in \mathrm{SAT}$ for a partial model on U hence specifically C_U is satisfiable proving (b). Observe that the if-part of (c) is already proven in [14], namely in the proof of Theorem 2 there. So it suffices to verify the converse direction for which we let $C_U \subseteq W_U$ meaning that $U \subseteq V(c)$ for all $c \in C(U)$. Thus $C(x) = C(U)$ for all $x \in U$, otherwise there are $x \in U$ and $c \in C(U)$ such that $x \notin V(c) \supseteq U$ which is impossible. Because of Lemma 4 (1) and $U \in \mathfrak{H}(C)$ one obtains $H_C(x) = H_C(U) = U$, for all $x \in U$. According to Lemma 2 (1) in [14], a hull U is free if and only if $H_C(x) = U$, for all $x \in U$, so the proof of (c) is finished. Finally, $C_U = C(U)$ implies $U = V(C_U) = V(C(U)) \supseteq H_C(U)$, thus $U = H_C(U)$, hence (d). $\square$

Note that given $C \in \mathrm{CNF}$ and $U \in \mathfrak{H}(C)$ for the hull structure of the subformula $C' := C \setminus C(U)$ in general it does not hold that $\mathfrak{H}(C') \subseteq \mathfrak{H}(C)$. In fact, for $Q = \{a, b, c\}$ and $U = \{v, x, y, z\}$ consider $C = W_Q \cup W_{\{x,y,z\}} \cup \{c\}$ with $c := \{avxyz\}$ represented as a literal string. Then $C(U) = W_{\{x,y,z\}} \cup \{avxyz\}$, $C' = W_Q$ and $U \in \mathfrak{H}(C)$, however $H_{C'}(Q) = Q \neq H_C(Q) = Q \cup \{v\}$. Instead the hull structure of C' can be characterized as follows also providing a co-autarky criterion discussed later.

Lemma 5 *Let $C \in$ CNF and $U \in \mathfrak{H}(C)$ then $\mathfrak{H}(C \setminus C(U)) = \{U' \setminus U : U' \in \mathfrak{H}(C)\}$.*

Proof. Again set $C' := C \setminus C(U)$ and define $\mathcal{D} := \{U' \setminus U : U' \in \mathfrak{H}(C)\}$. Consider the inclusion from left to right and let $Q \in \mathfrak{H}(C')$ then $Q = H_C(Q) \setminus U \in \mathcal{D}$. Indeed, if $Q \in \mathfrak{H}(C)$ then $Q = H_C(Q)$ and we are done. So, assume that $Q \notin \mathfrak{H}(C)$. First, $Q \cap U = \emptyset$ otherwise there is $x \in Q \cap U$ and $c \in C'$ with $x \in V(c)$ implying $c \in C(U)$ yielding a contradiction. Next, for every $x \in H_C(Q) \setminus Q =: S \neq \emptyset$ there is a clause $c \in C(U) \cap C(Q)$ such that $V(c) \cap Q \neq \emptyset$ with $x \in V(c) \setminus Q$. Suppose that there also is a clause $c' \in C'$ with $x \in V(c')$. As $x \notin Q \in \mathfrak{H}(C')$, x must occur in c' and also in a clause $c'' \in C' \setminus C'(Q)$. The latter implies $x \in V(C \setminus C(Q))$ while $x \in V(C(Q))$ thus $x \notin H_C(Q)$ contradicting the main assumption. Now, as $U \in \mathfrak{H}(C)$ and $x \notin V(C')$ we must have $x \in U$, therefore $S \subseteq U$ in summary verifying that $Q = H_C(Q) \setminus S = H_C(Q) \setminus U \in \mathcal{D}$. For the reverse inclusion we show that $\mathcal{D} \ni Q := U' \setminus U \in \mathfrak{H}(C')$ for every pair $U, U' \in \mathfrak{H}(C)$. If $U = \emptyset$ one has $C' = C$ and $\mathfrak{H}(C') = \mathfrak{H}(C)$. So, let $U \neq \emptyset$ and assume $Q \notin \mathfrak{H}(C')$. Then $\emptyset \neq Q \subsetneq H_{C'}(Q)$ meaning that there is $c \in C'(Q)$ with $x \in V(c)$ such that neither $x \in V(C' \setminus C'(Q))$ nor $x \in Q$ but $V(c) \cap U' \neq \emptyset$ hence $c \in C(U')$. Now, either one of the following situations can occur:
Case 1, $x \in U'$: Then $x \in U \cap U'$ because $U \neq \emptyset$, but as $U \in \mathfrak{H}(C)$ one obtains $x \notin V(C')$ yielding a contradiction to $x \in V(C'(Q))$.
Case 2, $x \notin U'$: Then either $x \in U \setminus U'$ yielding a contradiction as above. Or $x \notin U$ thus $x \notin U' \cup U$. As $\emptyset \neq U \in \mathfrak{H}(C)$, $x \in V(C(U))$ implies $x \notin V(C'(U))$ yielding a contradiction to $x \in V(C'(Q))$, hence $x \notin C(U)$. And we can focus on C'. As $U' \in \mathfrak{H}(C)$ it follows that there is a further clause $c' \in C' \setminus C'(U') \subseteq C' \setminus C'(Q)$ with $x \in V(c')$. On the other hand, since $x \in H_{C'}(Q)$ there can be no clause in $C' \setminus C'(Q)$ containing x establishing a contradiction also in this case and finishing the proof. □

The next result characterizes the co-autarky of a superhull which especially holds if the underlying co-autark hull is free.

Theorem 5 *For $C \in$ CNF, let $\emptyset \neq U \in \mathfrak{H}(C)$.
(1) If U is co-autark, then any superhull $\hat{U}$ of U is also co-autark if one of the following holds (i) $(\hat{U} \setminus U) \cap V(C(U)) = \emptyset$ or (ii) $\hat{U} \setminus U$ is co-autark for $C(\hat{U}) \setminus C(U)$.
(2) If U is free then U can be checked for co-autarky in linear time.*

Proof. In case (1), (i), no variable of $\hat{U} \setminus U$ appears in $C(U)$ thus $C(U)$ cannot be satisfied by variables in $\hat{U}$ outside U. In case (1), (ii), let $C' := C \setminus C(U)$ then by Lemma 5, $Q := \hat{U} \setminus U$ is a hull in $\mathfrak{H}(C')$. Clearly $C'(\hat{U}) = C(\hat{U}) \setminus C(U)$, thus if Q is not autark then $C'(\hat{U})$ cannot be satisfied over Q, hence there is no way for satisfying $C(\hat{U})$ over $\hat{U}$. The proof of claim (2) can be found in [14]. □

Clearly the previous theorem specifically holds in the case of linear formulas. The following two lemmas collect some further properties of autarkies that are not hard to verify, so the proofs are omitted.

Lemma 6 *Let $C \in$ CNF, have two autarkies $\alpha_i : U_i \to \{0,1\}$, $i = 1,2$. Then $\alpha_1|\alpha_2 : U_1 \cup U_2 \to \{0,1\}$ and $\alpha_2|\alpha_1 : U_1 \cup U_2 \to \{0,1\}$ defined by*

$$\alpha_i|\alpha_j(x) := \begin{cases} \alpha_i(x); x \in U_i \\ \alpha_j(x); x \in U_j - U_i \end{cases}$$

where $i \neq j \in \{1,2\}$, also are autarkies of C.□

Lemma 7 *Let $C \in$ CNF and $U \subseteq V(C)$ be arbitrary. If $\alpha : U' \to \{0,1\}$ is an autarky of C, then the restriction $\alpha|U' \cap V(C \setminus C(U))$ is an autarky of $C \setminus C(U)$.* □

5 Complexity Aspects

It is no surprise that both characterizations above do not provide an efficient algorithm for the computational problem AUT, which given $C \in$ CNF asks whether $C \in$ AUT. Since SAT $\subseteq$ AUT deciding AUT is at least as hard as deciding SAT hence AUT is NP-complete which below we prove rigorously. Let the formula class AUT$_k \subseteq$ AUT be defined as the collection of formulas such that there exists an autark set U of cardinality at most k, which is a fixed positive integer. Then AUT$_k$ corresponds to the following computational problem:
Given: $C \in$ CNF, $k \in \mathbb{N}$,
Question: $\exists U \subseteq V(C)$, $|U| \leq k$ autark?
For positive integer r, define r-AUT as AUT restricted to r-CNF formulas. Similarly, we proceed for r-AUT$_k$.

Theorem 6 *Let r, k be positive integers. (1) AUT and AUT$_k$ are NP-complete. (2) r-AUT and r-AUT$_k$ are NP-complete, for each fixed $r \geq 3$.*

Proof. Clearly AUT is in NP and assume it is not NP-complete. Then for $C \in$ CNF in polynomial time we can decide whether there is a set $U \subseteq V(C)$ which is autark. Assume that there is no such U, then clearly C is unsatisfiable . On the other hand, if there is an autark set U we can proceed with $C \setminus C(U)$ and so on until we achieve a subformula that has no autark set, then C is unsatisfiable, or we obtain the empty set. Obviously, this amounts to a polynomial time procedure for deciding SAT yielding a contradiction and the first claim. Since r-SAT is NP-complete we can proceed for r-AUT analogously. Next, obviously $\mathrm{AUT}_k \in \mathrm{NP}$ and its NP-completeness can be derived as follows. Let $\mathrm{AUT}_k^=$ be the special version of AUT_k for which U is required to have cardinality exactly k, then we claim CNF-SAT $\leq_p \mathrm{AUT}_k^=$ meaning the existence of a polynomial-time reduction. Indeed, if $C \in \mathrm{CNF}$ is a SAT input instance then we assign to it the instance $(C, |V(C)|)$ for $\mathrm{AUT}_k^=$. Now $C \in \mathrm{SAT}$ if and only if there is a (partial) truth assignment $\alpha : V(C) \to \{0,1\}$ satisfying C if and only if $(C, |V(C)|) \in \mathrm{AUT}_k^=$. In order to show that r-AUT_k is NP-complete, as above, we use the reduction from r-SAT to r-$\mathrm{AUT}_k^=$, for each fixed $r \geq 3$. □

Theorem 7 *Given $C \in \mathrm{CNF}$, positive integer k with $k < |V(C)| =: n$, then in time $O(n^k T(k\text{-SAT}_k))$, one can test whether $C \in \mathrm{AUT}_k$ and if, find a model for C, where $T(k\text{-SAT}_k)$ denotes the time for solving k-SAT on instances over k variables.*

Proof. For each set $U \subset V(C)$ with $|U| \leq k$ in linear time compute the retraction of C over U and, if necessary, remove all unit clauses from it by evaluating it accordingly. Obviously, the resulting formula $C_U \in k$-CNF and $|V(C_U)| \leq |U| \leq k$, hence we can check in time $T(k\text{-SAT}_k)$ whether $C_U \in \mathrm{SAT}$ yielding the decision whether U is autark according to Theorem 4. There are $O(n^k)$ subsets of $V(C)$ of at most k elements yielding the claim. □

The last result amounts to an FPT algorithm [6] for recognizing membership of k-CNF formulas in AUT_r, for a fixed positive integer k.

Corollary 2 *For $C \in k$-CNF and $r \in \mathbb{N}$, it can be tested in time*

$$\begin{cases} O(|C|2^k T(r\text{-SAT}_r); r \leq k \\ O(|C|r^k T(k\text{-SAT}_r); r > k \end{cases}$$

whether $C \in \mathrm{AUT}_r$, where again $T(i\text{-SAT}_j)$ denotes the time of the currently best known algorithm for solving i-SAT for an input instance of j variables.

Theorem 8 2-AUT *is polynomial-time solvable.*

Proof. First in polynomial time construct the formula graph G_C for an instance $C \in 2$-CNF with n variables and m clauses. The number of vertices in G_C is upper bounded by $m = m(n) \in O(n^2)$. Next, in polynomial time compute the connected components G_i, $1 \leq i \leq s$, of G_C, where also $s = s(n) \in O(n^2)$. According to Theorem 4 (a), exactly those subformulas C_i of C determined by G_i correspond to the variable hulls in C, more precisely $\mathfrak{H}(C) = \{V(C_i) : 1 \leq i \leq s\}$. Thus according to Theorem 3 it suffices to check whether $C_i \in 2$-SAT, $1 \leq i \leq s$, which for each i can be done in linear time [1]. In summary 2-AUT is shown to be polynomial-time decidable. □

6 Concluding Remarks and Open Problems

The autarky and co-autarky structure specifically of linear formulas was investigated regarding the classical satisfiability problem. Several tasks remain for future work such as the autarky structure of diagonal fibre transversals or of exact-linearly based formulas, which have an exact linear base hypergraph. An open question is how linear formulas in AUT' can be characterized and recognized. Also the hull structure of formulas needs to be investigated further. Specifically, assume there are hulls U', U such that U is not autark, $Q := U' \setminus U$ is autark in $C' := C \setminus C(U)$ and $Q \cap V(C(U)) \neq \emptyset$, then the question remains whether U' is autark in C according to Theorem 5 (1). Specifically the hull structure of linear formulas is of interest as they are the hardest in most algorithmic approaches. Observe that the method in the proof of Theorem 8 cannot be transfered immediately to 2-AUT_k because there might be autark sets contained in autark hulls thus having a smaller size, so the complexity of this problem remains open. Finally, it would be relevant to transfer the autarky notion also to other branches of satisfiability such as exact satisfiablity or not-all-equal satisfiability. The latter problems which in general are NP-complete [8] have been investigated recently from a complexity theoretical point of view when restricted to various linear formula classes [16]. Intending an algorithmic progress for these variants of SAT the investigation of a related autarky structure might be useful.

References

[1] Aspvall, B., Plass, M.R., Tarjan, R.E., “A linear-time algorithm for testing the truth of certain quantified Boolean formulas,” *Inform. Process. Lett.* pp. 121-123, 8/1979.

[2] Berge, C., *Hypergraphs*, North-Holland, Amsterdam, 1989.

[3] Boros, E., Crama, Y., Hammer, P.L., “Polynomial-time inference of all valid implications for Horn and related formulae,” *Annals of Math. Artif. Intellig.* pp. 21-32, 1/1990.

[4] Clarke, E.M., Grumberg, O., Peled, D.A., *Model Checking*, The MIT Press, 2000.

[5] Cook, S.A., “The Complexity of Theorem Proving Procedures,” *3rd ACM Symposium on Theory of Computing* pp. 151-158, 1971.

[6] Downey, R.G., Fellows, M.R., *Parameterized Complexity*, Springer-Verlag, New York, 1999.

[7] Franco, J., VanGelder, A., “A perspective on certain polynomial-time solvable classes of satisfiability,” *Discrete Appl. Math.* pp. 177-214, 125/2003.

[8] Garey M.R., Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.

[9] Gu, J., Purdom, P.W., Franco, J., Wah, B.W., "Algorithms for the Satisfiability (SAT) Problem: A Survey," in: D. Du, J. Gu, P. M. Pardalos (Eds.), *Satisfiability Problem: Theory and Applications*, DIMACS Workshop, March 11-13, 1996, DIMACS Series, V35, pp. 19-151, American Mathematical Society, Providence, Rhode Island, 1997.

[10] Karp, R.M., "Reducibility Among Combinatorial Problems," in: *Proc. Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y.*, Plenum, New York, pp. 85-103, 1972.

[11] Knuth, D.E., "Nested satisfiability," *Acta Informatica* pp. 1-6, 28/1990.

[12] Lewis, H.R., "Renaming a Set of Clauses as a Horn Set," *J. ACM* pp. 134-135, 25/1978.

[13] Monien, B., Speckenmeyer, E., "Solving satisfiability in less than 2^n steps," *Discrete Appl. Math.* pp. 287-295, 10/1985.

[14] Porschen, S., "A CNF Formula Hierarchy over the Hypercube," Proc. AI 2007, *LNAI* pp. 234-243, 4830/2007.

[15] Porschen, S., "On Problems With Closure Properties," in: S. Ao, O. Castillo, X. Huang (Eds.), *Intelligent Control and Innovative Computing*, Lecture Notes in Electrical Engineering Vol. 110, pp. 325-336, Springer, New York 2012.

[16] Porschen, S., Schmidt, T., Speckenmeyer, E., Wotzlaw, A., "XSAT and NAE-SAT of linear CNF classes," *Discrete Appl. Math.* , pp. 1-14, 167/2014.

[17] Porschen, S., Speckenmeyer, E., Zhao, X., "Linear CNF formulas and satisfiability," *Discrete Appl. Math.*, pp. 1046-1068, 157/2009.

[18] Schlipf, J., Annexstein, F.S., Franco, J., Swaminathan, R.P., "On finding solutions for extended Horn formulas," *Inform. Process. Lett.* pp. 133-137, 54/1995.

[19] Stalmarck, G., Säflund, M., "Modeling and verifying systems and software in propositional logic," in Daniels, B.K. (editor), *Safety of Computer Control Systems* (SAFECOMP 90), Pergamon Press, pp. 31-36, 1990.

[20] Tovey, C.A., "A Simplified NP-Complete Satisfiability Problem", *Discrete Appl. Math.* pp. 85-89, 8/1984.