On SAT representations of XOR constraints

Matthew Gwynne[1] and Oliver Kullmann[2]

[1] Computer Science Department, Swansea University
http://cs.swan.ac.uk/~csmg/
[2] Computer Science Department, Swansea University
http://cs.swan.ac.uk/~csoliver

Abstract. We consider the problem of finding good representations, via boolean conjunctive normal forms F (clause-sets), of systems S of XOR-constraints $x_1 \oplus \cdots \oplus x_n = \varepsilon$, $\varepsilon \in \{0,1\}$ (also called parity constraints), i.e., systems of linear equations over the two-element field. These representations are to be used as parts of SAT problems $F^* \supset F$, such that F^* has "good" properties for SAT solving. The basic quality criterion is "arc consistency", that is, for every partial assignment φ to the variables of S, all assignments $x_i = \varepsilon$ forced by φ are determined by unit-clause propagation on the result $\varphi * F$ of the application. We show there is no arc-consistent representation of polynomial size for arbitrary S. The proof combines the basic method by Bessiere et al. 2009 ([2]) on the relation between monotone circuits and "consistency checkers", adapted and simplified in the underlying report Gwynne et al. [10], with the lower bound on monotone circuits for monotone span programs in Babai et al. 1999 [1]. On the other side, our basic positive result is that computing an arc-consistent representation is fixed-parameter tractable in the number m of equations of S. To obtain stronger representations, instead of mere arc-consistency we consider the class $\mathcal{PC}$ of propagation-complete clause-sets, as introduced in Bordeaux et al. 2012 [4]. The stronger criterion is now $F \in \mathcal{PC}$, which requires for *all* partial assignments, possibly involving also the auxiliary (new) variables in F, that forced assignments can be determined by unit-clause propagation. We analyse the basic translation, which for $m = 1$ lies in $\mathcal{PC}$, but fails badly so already for $m = 2$, and we show how to repair this.

Keywords: arc consistency, parity constraints, monotone circuits, monotone span programs, unit-propagation complete, acyclic incidence graph

1 Introduction

Recall that the two-element field $\mathbb{Z}_2$ has elements $0, 1$, where addition is XOR, which we write as $\oplus$, while multiplication is AND, written $\cdot$. A linear system S of equations over $\mathbb{Z}_2$, in matrix form $A \cdot x = b$, where A is an $m \times n$ matrix over $\{0,1\}$, with m the number of equations, n the number of variables, while $b \in \{0,1\}^m$, yields a boolean function f_S, which assigns 1 to a total assignment of the n variables of S iff that assignment is a solution of S. The task of finding

"good" representations of f_S by conjunctive normal forms F (clause-sets, to be precise), for the purpose of SAT solving, shows up in many applications, for example cryptanalysing the Data Encryption Standard and the MD5 hashing algorithm in [5], translating Pseudo-Boolean constraints to SAT in [6], and in roughly 1 in 6 benchmarks from SAT 2005 to 2011 according to [19].

The basic criterion for a good F is "arc-consistency". See Chapter 3 of [24] for an overview of "arc-consistency" at the constraint level, see [7] for discussion of the support encoding, a SAT translation of explicitly-given constraints which maintains arc-consistency, and see [6] for an overview of maintaining arc-consistency when translating Pseudo-boolean constraints to SAT. To define arc-consistency for SAT, we use r_1 for unit-clause propagation, and we write $\varphi * F$ for the application of a partial assignment φ to a clause-set F.[3)] For a boolean function f, a CNF-representation F of f is arc-consistent iff for every partial assignment φ to the variables of f the reduced instantiation $F' := \mathrm{r}_1(\varphi * F)$ has no forced assignments anymore, that is, for every remaining variable v and $\varepsilon \in \{0,1\}$ the result $\langle v \to \varepsilon \rangle * F'$ of assigning ε to v in F' is satisfiable.

We show that there is no polynomial-size arc-consistent representation of arbitrary S (Theorem 7). The proof combines the translation of arc-consistent CNF-representations of f into monotone circuits computing a monotonisation $\widehat{f}$, motivated by [2] and proven in the underlying report [10], with the lower bound on monotone circuit sizes for monotone span programs (msp's) from [1]. Besides this fundamental negative result, we provide various forms of good representations of systems S with bounded number of equations. Theorem 12 shows that there is an arc-consistent representation with $O(n \cdot 2^m)$ many clauses. The remaining results use a stronger criterion for a "good" representation, namely they demand that $F \in \mathcal{PC}$, where $\mathcal{PC}$ is the class of "unit-propagation complete clause-sets" as introduced in [4] — while for arc-consistency only partial assignments to the variables of f are considered, now partial assignments for all variables in F (which contains the variables of f, and possibly further auxiliary variables) are to be considered. For $m = 1$ the obvious translation X_1, by subdividing the constraints into small constraints, is in $\mathcal{PC}$ (Lemma 10). For $m = 2$ we have an intelligent representation X_2 in $\mathcal{PC}$ (Theorem 13), while the use of X_1 (piecewise) is still feasible for full (dag-)resolution, but not for tree-resolution. We conjecture (Conjecture 14) that Theorem 12 and Theorem 13 can be combined, which would yield a fixed-parameter tractable algorithm for computing a representation $F \in \mathcal{PC}$ for arbitrary S with the parameter m.

It is well-known that translating each XOR to its prime implicates can result in hard (unsatisfiable) instances for resolution. This goes back to the "Tseitin formulas" introduced in [26], which were proven hard for full resolution in [27], and generalised to (empirically) hard satisfiable instances in [12]. Thus, to tackle XOR-constraints, some solvers integrate XOR reasoning. `EqSatz` ([23]) extracts XOR clauses from its input and applies DP-resolution plus incomplete XOR

[3)] r_1 has been generalised to r_k for $k \in \mathbb{N}_0$ in [14,15]. In the underlying report [10] we discuss this form of generalised unit-clause propagation, where for example r_2 is failed-literal elimination, but in this paper we concentrate on r_1.

reasoning rules. `CryptoMiniSAT` ([25]) integrates Gaußian elimination during search, allowing both explicitly specified XOR clauses and also XOR clauses extracted from CNF input, however in the newest version 3.3 the XOR handling during search is removed, since it is deemed too expensive. [17] integrates XOR reasoning into `MiniSat` in a similar manner to SMT, while [18] expands on this by reasoning about equivalence classes of literals created by binary XORs. [20] learns conflicts in terms of "parity (XOR) explanations". [21] extends the reasoning from "Gaußian elimination" to "Gauß-Jordan elimination", which also detects forced literals, not just inconsistencies. Still, for leading current SAT solvers usage of SAT translations is important. Considering such translations of XORs to CNF, [19] identifies the subsets of "tree-like" systems of XOR constraints, where one obtains an arc-consistent CNF representation; our results on acyclic systems strengthens this. Additionally they consider equivalence reasoning, where for "cycle-partitionable" systems of XOR constraints this reasoning is sufficient to derive all conclusions. They also show how to eliminate the need for such special reasoning by another arc-consistent CNF representation. In general, the idea is to only use Gaußian elimination for such parts of XOR systems which the SAT solver is otherwise incapable of propagating on. Existing propagation mechanisms, especially unit-clause propagation, and to a lesser degree equivalence reasoning, are very fast, while Gaußian elimination is much slower. Experimental evaluation on SAT 2005 benchmarks instances showed that such CNF translations can outperform dedicated XOR reasoning modules.

Viewing a linear system S as a constraint on $\mathrm{var}(S)$, one can encode evaluation via Tseitin's translation, obtaining a CNF-representation F with the property that for every *total* assignment φ, i.e., $\mathrm{var}(\varphi) = \mathrm{var}(S)$, we have that $\mathrm{r}_1(\varphi * F)$ either contains the empty clause or is empty.[4] However, as Theorem 7 shows, there is no polysize representation which treats all *partial* assignments. Gaußian elimination handles all partial assignments in polynomial time (detects unsatisfiability of $\varphi * F$ for all partial assignments φ), but this can not be integrated into the CNF formalism (by using auxiliary variables and clauses), since algorithms always need total assignments, and so partial assignments φ would need to be encoded — the information "variable v not assigned" (i.e., $v \notin \mathrm{var}(\varphi)$) needs to be represented by *setting* some auxiliary variable, and this must happen by a mechanism outside of the CNF formalism. It is an essential strength of the CNF formalism to allow partial instantiation; if we want these partial instantiations also to be easily understandable by a SAT solver, then the results of [2] and our results show that there are restrictions. Yet there is little understanding of these restrictions. There are many examples where arc-consistent and stronger representations are possible, while the current non-representability results, one in [2], one in this article and a variation on [2] in [16], rely on non-trivial lower bounds on monotone circuit complexity; in fact, as we show in [10], there is a polysize arc-consistent representation of a boolean function f iff the monotonisation $\widehat{f}$, encoding partial assignments to f, has polysize monotone circuits.

[4] In Subsection 9.4.1 of [11] this class of representations is called $\exists\mathcal{UP}$; up to linear-time transformation it is the same as representations by boolean circuits.

2 Preliminaries

We follow the general notations and definitions as outlined in [13]; for full details see the underlying report [10] (which contains additional and generalised results). We use $\mathbb{N} = \{1, 2, \ldots\}$ and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Let $\mathcal{VA}$ be the infinite set of variables, and let $\mathcal{LIT} = \mathcal{VA} \cup \{\overline{v} : v \in \mathcal{VA}\}$ be the set of literals, the disjoint union of variables as positive literals and complemented variables as negative literals. We use $\overline{L} := \{\overline{x} : x \in L\}$ to complement a set L of literals. A clause is a finite subset $C \subset \mathcal{LIT}$ which is complement-free, i.e., $C \cap \overline{C} = \emptyset$; the set of all clauses is denoted by $\mathcal{CL}$. A clause-set is a finite set of clauses, the set of all clause-sets is $\mathcal{CLS}$. By $\mathrm{var}(x) \in \mathcal{VA}$ we denote the underlying variable of a literal $x \in \mathcal{LIT}$, and we extend this via $\mathrm{var}(C) := \{\mathrm{var}(x) : x \in C\} \subset \mathcal{VA}$ for clauses C, and via $\mathrm{var}(F) := \bigcup_{C \in F} \mathrm{var}(C)$ for clause-sets F. The possible literals in a clause-set F are denoted by $\mathrm{lit}(F) := \mathrm{var}(F) \cup \overline{\mathrm{var}(F)}$. Measuring clause-sets happens by $n(F) := |\mathrm{var}(F)|$ for the number of variables, $c(F) := |F|$ for the number of clauses, and $\ell(F) := \sum_{C \in F} |C|$ for the number of literal occurrences. A special clause-set is $\top := \emptyset \in \mathcal{CLS}$, the empty clause-set, and a special clause is $\bot := \emptyset \in \mathcal{CL}$, the empty clause. A partial assignment is a map $\varphi : V \to \{0, 1\}$ for some finite $V \subset \mathcal{VA}$, where we set $\mathrm{var}(\varphi) := V$, and where the set of all partial assignments is $\mathcal{PASS}$. For $v \in \mathrm{var}(\varphi)$ let $\varphi(\overline{v}) := \overline{\varphi(v)}$ (with $\overline{0} = 1$ and $\overline{1} = 0$). We construct partial assignments by terms $\langle x_1 \to \varepsilon_1, \ldots, x_n \to \varepsilon_n \rangle \in \mathcal{PASS}$ for literals $x_1, \ldots, x_n$ with different underlying variables and $\varepsilon_i \in \{0, 1\}$. For $\varphi \in \mathcal{PASS}$ and $F \in \mathcal{CLS}$ we denote the result of applying φ to F by $\varphi * F$, removing clauses $C \in F$ containing $x \in C$ with $\varphi(x) = 1$, and removing literals x with $\varphi(x) = 0$ from the remaining clauses. By $\mathcal{SAT} := \{F \in \mathcal{CLS} \mid \exists\, \varphi \in \mathcal{PASS} : \varphi * F = \top\}$ the set of satisfiable clause-sets is denoted, and by $\mathcal{USAT} := \mathcal{CLS} \setminus \mathcal{SAT}$ the set of unsatisfiable clause-sets. By $\mathbf{r_1} : \mathcal{CLS} \to \mathcal{CLS}$ we denote unit-clause propagation, that is $\mathrm{r}_1(F) := \{\bot\}$ if $\bot \in F$, $\mathrm{r}_1(F) := F$ if F contains only clauses of length at least 2, while otherwise a unit-clause $\{x\} \in F$ is chosen, and recursively we define $\mathrm{r}_1(F) := \mathrm{r}_1(\langle x \to 1 \rangle * F)$; it is easy to see that the final result $\mathrm{r}_1(F)$ does not depend on the choice of unit-clauses. Reduction by r_1 applies certain **forced assignments** to the (current) F, which are assignments $\langle x \to 1 \rangle$ such that the opposite assignment yields an unsatisfiable clause-set, that is, where $\langle x \to 0 \rangle * F \in \mathcal{USAT}$; the literal x here is also called a **forced literal**. Two clauses $C, D \in \mathcal{CL}$ are resolvable iff they clash in exactly one literal x, that is, $C \cap \overline{D} = \{x\}$, in which case their resolvent is $(C \cup D) \setminus \{x, \overline{x}\}$ (with resolution literal x). A resolution tree is a full binary tree formed by the resolution operation. We write $T : F \vdash C$ if T is a resolution tree with axioms (the clauses at the leaves) all in F and with derived clause (at the root) C. A *prime implicate* of $F \in \mathcal{CLS}$ is a clause C such that a resolution tree T with $T : F \vdash C$ exists, but no T' exists for some $C' \subset C$ with $T' : F \vdash C'$; the set of all prime implicates of F is denoted by $\mathbf{prc_0}(F) \in \mathcal{CLS}$. Two clause-sets $F, F' \in \mathcal{CLS}$ are equivalent iff $\mathrm{prc}_0(F) = \mathrm{prc}_0(F')$. A clause-set F is unsatisfiable iff $\mathrm{prc}_0(F) = \{\bot\}$. If F is unsatisfiable, then every literal $x \in \mathcal{LIT}$ is a forced literal for F, while otherwise x is forced for F iff $\{x\} \in \mathrm{prc}_0(F)$.

3 Propagation-hardness and $\mathcal{PC}$

A clause-set F is a "CNF-representation" of a boolean function f, if the satisfying assignments of F projected to the variables of f are precisely the satisfying assignments of f. Stronger, F is an "arc-consistent" representation of f, if F is a CNF-representation of f and $\mathrm{phd}^{\mathrm{var}(f)}(F) \leq 1$ holds, which is defined as follows.

Definition 1. *For $F \in \mathcal{CLS}$ and $V \subseteq \mathcal{VA}$ the relation $\mathbf{phd}^V(F) \leq 1$ holds (F has p(ropagation)-hardness at most 1 relative to V) if for all partial assignments $\varphi \in \mathcal{PASS}$ with $\mathrm{var}(\varphi) \subseteq V$ the clause-set $F' := \mathrm{r}_1(\varphi * F)$ has no forced literals $x \in \mathrm{lit}(F')$, that is, for all $x \in \mathrm{lit}(F')$ the clause-set $\langle x \to 0\rangle * F'$ is satisfiable. We write "$\mathrm{phd}(F)$" for "$\mathrm{phd}^{\mathrm{var}(F)}(F)$". The class $\mathcal{PC} \subset \mathcal{CLS}$ is the set of all F with $\mathrm{phd}(F) \leq 1$ (the class of **unit-propagation-complete clause-sets**).*

See [8,9] and the underlying report [10] for the general picture, where the measure $\mathrm{phd}^V(F) \in \mathbb{N}_0$ is defined in general. We now present the basic graph-theoretic criterion for $\bigcup_{i\in I} F_i \in \mathcal{PC}$ for clause-sets $F_i \in \mathcal{PC}$.

Definition 2. *For a finite family $(F_i)_{i\in I}$ of clause-sets $F_i \in \mathcal{CLS}$ the **incidence graph** $B((F_i)_{i\in I})$ is the bipartite graph, where the two parts are given by $\bigcup_{i\in I} \mathrm{var}(F_i)$ and I, while there is an edge between v and i if $v \in \mathrm{var}(F_i)$. We say that $(F_i)_{i\in I}$ **is acyclic** if $B((F_i)_{i\in I})$ is acyclic (i.e., has no cycle as an (undirected) graph, or, equivalently, is a forest). A single clause-set $F \in \mathcal{CLS}$ **is acyclic** if $(\{C\})_{C\in F}$ is acyclic.*

The following central lemma is kind of folklore in the CSP literature; for a complete proof see the underlying report [10].

Lemma 3. *Consider an acyclic family $(F_i)_{i\in I}$ of clause-sets. If no F_i has forced assignments, then also $\bigcup_{i\in I} F_i$ has no forced assignments.*

We obtain a sufficient criterion for the union of unit-propagation complete clause-sets to be itself unit-propagation complete:

Theorem 4. *Consider an acyclic family $(F_i)_{i\in I}$ of clause-sets. If for all $i \in I$ we have $F_i \in \mathcal{PC}$, then also $\bigcup_{i\in I} F_i \in \mathcal{PC}$.*

Proof. Let $F := \bigcup_{i\in I} F_i$, and consider a partial assignment φ with $F' \neq \{\bot\}$ for $F' := \mathrm{r}_1(\varphi * F)$. We have to show that F' has no forced assignments. For all $i \in I$ we have $\mathrm{r}_1(\varphi * F_i) \neq \{\bot\}$, and thus $\mathrm{r}_1(\varphi * F_i)$ has no forced assignments (since $F_i \in \mathcal{PC}$). So $\bigcup_{i\in I} \mathrm{r}_1(\varphi * F_i)$ has no forced assignments by Lemma 3. Using that for $A, B \in \mathcal{CLS}$ holds $\mathrm{r}_1(A \cup \mathrm{r}_1(B)) = \mathrm{r}_1(A \cup B)$, we get $F' = \mathrm{r}_1(\bigcup_{i\in I} \varphi * F_i) = \mathrm{r}_1(\bigcup_{i\in I} \mathrm{r}_1(\varphi * F_i)) = \bigcup_{i\in I} \mathrm{r}_1(\varphi * F_i)$, whence F' has no forced assignments. □

Two special cases of acyclic $(F_i)_{i\in I}$ are of special importance to us, and are spelled out in the following corollary (see [10] for full details).

Corollary 5. *Consider a family $(F_i)_{i\in I}$ of clause-sets with $F_i \in \mathcal{PC}$ for all $i \in I$. Then each of the following conditions implies $\bigcup_{i\in I} F_i \in \mathcal{PC}$:*

1. *Any two different clause-sets have at most one variable in common, and the variable-interaction graph is acyclic. (The variable-interaction graph has vertex-set I, while there is an edge between $i, j \in I$ with $i \neq j$ if* $\mathrm{var}(F_i) \cap \mathrm{var}(F_j) \neq \emptyset$.*)*
2. *There is a variable v with* $\mathrm{var}(F_i) \cap \mathrm{var}(F_j) \subseteq \{v\}$ *for all $i, j \in I$, $i \neq j$.*

4 XOR-clause-sets

As usual, an **XOR-constraint** (also known as "parity constraint") is a (boolean) constraint of the form $x_1 \oplus \cdots \oplus x_n = \varepsilon$ for literals $x_1, \ldots, x_n$ and $\varepsilon \in \{0, 1\}$, where $\oplus$ is the addition in the 2-element field $\mathbb{Z}_2 = \{0, 1\}$. Note that $x_1 \oplus \cdots \oplus x_n = y$ is equivalent to $x_1 \oplus \cdots \oplus x_n \oplus y = 0$, while $x \oplus x = 0$ and $x \oplus \overline{x} = 1$, and $0 \oplus x = x$ and $1 \oplus x = \overline{x}$. Two XOR-constraints are *equivalent*, if they have exactly the same set of solutions. We represent XOR-constraints by **XOR-clauses**, which are just ordinary clauses $C \in \mathcal{CL}$, but now under a different interpretation, namely implicitly interpreting C as the XOR-constraints $\oplus_{x \in C} x = 0$. And instead of systems of XOR-constraints we just handle **XOR-clause-sets** F, which are sets of XOR-clauses, that is, ordinary clause-sets $F \in \mathcal{CLS}$ with a different interpretation. So two XOR-clauses C, D are equivalent iff $\mathrm{var}(C) = \mathrm{var}(D)$ and the number of complements in C has the same parity as the number of complements in D. That clauses are sets is justified by the commutativity of XOR, while repetition of literals is not needed due to $x \oplus x = 0$. Clashing literal pairs can be removed by $x \oplus \overline{x} = 1$ and $1 \oplus y = \overline{y}$, as long as there is still a literal left. So every XOR-constraint can be represented by an XOR-clause except of inconsistent XOR-constraints, where the simplest form is $0 = 1$; we can represent this by two XOR-clauses $\{v\}, \{\overline{v}\}$. In our theoretical study we might even assume that the case of an inconsistent XOR-clause-set is filtered out by preprocessing.

The appropriate theoretical background for XOR-constraints is the theory of systems of linear equations over a field (here the two-element field). To an XOR-clause-set F corresponds a system $A(F) \cdot \boldsymbol{v} = b(F)$, using ordinary matrix notation. To make this correspondence explicit we use $n := n(F)$, $m := c(F)$, $\mathrm{var}(F) = \{v_1, \ldots, v_n\}$, and $F = \{C_1, \ldots, C_m\}$. Now F yields an $m \times n$ matrix $A(F)$ over $\mathbb{Z}_2$ together with a vector $b(F) \in \{0, 1\}^m$, where the rows $A(F)_{i,-}$ of $A(F)$ correspond to the clauses $C_i \in F$, where a coefficient $A(F)_{i,j}$ of v_j is 0 iff $v_j \notin \mathrm{var}(C_i)$, and $b_i = 0$ iff the number of complementations in C_i is even.

A partial assignment $\varphi \in \mathcal{PASS}$ satisfies an XOR-clause-set F iff $\mathrm{var}(\varphi) \supseteq \mathrm{var}(F)$ and for every $C \in F$ the number of $x \in C$ with $\varphi(x) = 1$ is even. An XOR-clause-set F implies an XOR-clause C if every satisfying partial assignment φ for F is also a satisfying assignment for $\{C\}$. The satisfying total assignments for an XOR-clause-set F correspond one-to-one to the solutions of $A(F) \cdot \boldsymbol{v} = b$ (as elements of $\{0, 1\}^n$), while implication of XOR-clauses C by F correspond to single equations $c \cdot \boldsymbol{v} = d$, which follow from the system, where c is an $1 \times n$-matrix over $\mathbb{Z}_2$, and $d \in \mathbb{Z}_2$. A **CNF-representation** of an XOR-clause-set $F \in \mathcal{CLS}$ is a clause-set $F' \in \mathcal{CLS}$ with $\mathrm{var}(F) \subseteq \mathrm{var}(F')$, such that the projections of the satisfying total assignments for F' (as CNF-clause-set) to $\mathrm{var}(F)$ are precisely the satisfying (total) assignments for F (as XOR-clause-set).

The resolution operation for CNF-clauses is the basic semantic operation, and analogically for XOR-clauses we have the addition of clauses, which corresponds to set-union, that is, from two XOR-clauses C, D follows $C \cup D$. Since we do not allow clashing literals, some rule is supposed here to translate $C \cup D$ into an equivalent $E \in \mathcal{CL}$ in case the two clauses are not inconsistent together. More generally, for an arbitrary XOR-clause-set F we can consider the *sum*, written as $\oplus F \in \mathcal{CL}$, which is defined as the reduction of $\bigcup F$ to some clause $\oplus F := E \in \mathcal{CL}$, assuming that the reduction does not end up in the situation $\{v, \overline{v}\}$ for some variable v — in this case we say that $\oplus F$ is inconsistent (which is only possible for $c(F) \geq 2$). The following fundamental lemma translates witnessing of unsatisfiable systems of linear equations and derivation of implied equations into the language of XOR-clause-sets; it is basically a result of linear algebra, and the proof is provided in the underlying report [10].

Lemma 6. *Consider an XOR-clause-set $F \in \mathcal{CLS}$.*

1. *F is unsatisfiable if and only if there is $F' \subseteq F$ such that $\oplus F'$ is inconsistent.*
2. *Assume that F is satisfiable. Then for all $F' \subseteq F$ the sum $\oplus F'$ is defined, and the set of all these clauses is modulo equivalence precisely the set of all XOR-clauses which follow from F.*

5 No arc-consistent representations in general

We now show, if there were polynomial size arc-consistent representations of all XOR-clause-sets, then all "monotone span programs" (msp's) could be computed by monotone boolean circuits, which is not possible by [1]. The first step here is to translate msp's into linear systems S. An msp computes a boolean function $f(x_1, \ldots, x_n) \in \{0, 1\}$ (with $x_i \in \{0, 1\}$), by using auxiliary boolean variables $y_1, \ldots, y_m$, and for each $i \in \{1, \ldots, n\}$ a linear system $A_i \cdot y = b_i$, where A_i is an $m_i \times m$ matrix over $\mathbb{Z}_2$. For the computation of $f(x_1, \ldots, x_n)$, a value $x_i = 0$ means the system $A_i \cdot y = b_i$ is active, while otherwise it is inactive; the value of f is 0 if all the active systems together are unsatisfiable, and 1 otherwise. Obviously f is monotonically increasing. The task is now to put that machinery into a single system S of equations. The main idea is to "dope" each equation of every $A_i \cdot y = b_i$ with a dedicated new boolean variable added to the equation, making that equation trivially satisfiable, independently of everything else; all these new variables together are called $z_1, \ldots, z_N$, where $N = \sum_{i=1}^{n} m_i$ is the number of equations in S. If all the doping variable used for a system $A_i \cdot y = b_i$ are set to 0, then they disappear and the system is active, while if they are not set, then the system is trivially satisfiable, and thus is deactivated. Now consider an arc-consistent representation F of S. Note that the x_i are not part of F, but the variables of F are $y_1, \ldots, y_m$ together with $z_1, \ldots, z_N$, where the latter represent in a sense the $x_1, \ldots, x_n$. Using F we can compute f by setting the z_j accordingly (if $x_i = 0$, then all z_j belonging to $A_i \cdot y = b_i$ are set to 0, if $x_i = 1$, then these variables stay unassigned), running r_1 on the system, and output 0 iff the empty clause was produced by r_1. By Theorem 6.1 in [10], based

on [2], finally from F we obtain a monotone circuit $\mathcal{C}$ computing f, whose size is polynomial in $\ell(F)$, where by [1] the size of $\mathcal{C}$ is $N^{O(\log N)}$.

Theorem 7. *There is no polynomial p s.t. for all XOR-clause-sets $F \in \mathcal{CLS}$ there is a representation $F' \in \mathcal{CLS}$ with $\ell(F') \le p(\ell(F))$ and $\mathrm{phd}^{\mathrm{var}(F)}(F') \le 1$.*

Proof. We consider representations of monotone boolean functions $f : \{0,1\}^n \to \{0,1\}$ (that is, $\boldsymbol{x} \le \boldsymbol{y} \Rightarrow f(\boldsymbol{x}) \le f(\boldsymbol{y})$) by monotone span programs (msp's). The input variables are given by $x_1, \ldots, x_n$. Additionally $m \in \mathbb{N}_0$ boolean variables $y_1, \ldots, y_m$ can be used, where m is the dimension, which we can also be taken as the size of the span program. For each $i \in \{1, \ldots, n\}$ there is a linear system $A_i \cdot y = b_i$ over $\mathbb{Z}_2$, where A_i is an $m_i \times m$ matrix with $m_i \le m$. For a total assignment φ, i.e., $\varphi \in \mathcal{PASS}$ with $\mathrm{var}(\varphi) = \{x_1, \ldots, x_n\}$, the value $f(\varphi)$ is 0 if and only if the linear systems given by $\varphi(x_i) = 0$ together are unsatisfiable, that is, $\{y \in \{0,1\}^m \mid \forall\, i \in \{1, \ldots, n\} : \varphi(x_i) = 0 \Rightarrow A_i \cdot y = b_i\} = \emptyset$. W.l.o.g. we assume that each system $A_i \cdot y = b_i$ is satisfiable.

Consider for each $i \in \{1, \ldots, n\}$ an XOR-clause-set $A'_i \in \mathcal{CLS}$ representing $A_i \cdot y = b_i$ (so $\mathrm{var}(A'_i) \supseteq \{y_1, \ldots, y_m\}$), where, as always, new variables for different A'_i are used, that is, for $i \ne j$ we have $(\mathrm{var}(A'_i) \cap \mathrm{var}(A'_j)) \setminus \{y_1, \ldots, y_m\} = \emptyset$. We use the process $\mathrm{D} : \mathcal{CLS} \to \mathcal{CLS}$ of "doping", as introduced in [11], where $\mathrm{D}(F)$ is obtained from F by adding to each clause a new variable. Let $A''_i := \mathrm{D}(A'_i)$, where the doping variables for different i do not clash; we denote them (altogether) by $z_1, \ldots, z_N$. Let $F := \bigcup_{i=1}^n A''_i$. Consider a CNF-representation F' of the XOR-clause-set F.

We have $f(\varphi) = 0$ iff $\varphi' * F' \in \mathcal{USAT}$, where φ' is a partial assignment with φ' assigning only doping variables z_j, namely if $\varphi(x_i) = 0$, then all the doping variables used in $\mathrm{D}(A'_i)$ are set to 0, while if $\varphi(x_i) = 1$, then nothing is assigned here. The reason is that by setting the doping variables to 0 we obtain the original system $A_i \cdot y = b_i$, while by leaving them in, this system becomes satisfiable whatever the assignments to the y-variables are.

Now assume that we have $\mathrm{phd}^{\{z_1, \ldots, z_N\}}(F') \le 1$. By Theorem 6.1 in [10] we obtain from F' a monotone circuit $\mathcal{C}$ (using only and's and or's) of size polynomial in $\ell(F')$ with input variables $z'_1, z''_1, \ldots, z'_N, z''_N$, where

- $z'_j = z''_j = 1$ means that z_j has not been assigned,
- $z'_j = 1$, $z''_j = 0$ means $z_j = 0$,
- $z'_j = 0$, $z''_j = 1$ means $z_j = 1$,
- while $z'_j = 0$, $z''_j = 0$ means "contradiction" (forcing the output of $\mathcal{C}$ to 0).

The value of $\mathcal{C}$ is 0 iff the corresponding partial assignment applied to F' yields an unsatisfiable clause-set. In $\mathcal{C}$ we now replace the inputs z_j by inputs x_i, which in case of $x_i = 0$ sets $z'_j = 1$, $z''_j = 0$ for all related j, while in case of $x_i = 1$ all related z'_j, z''_j are set to 1.[5)] This is now a monotone circuit computing f. By [1], Theorem 1.1, thus it is not possible that F' is of polynomial size in F. □

[5)] In other words, for the j related to i always all z'_j are set to 1, while $z''_j = x_i$.

6 The translations X_0, X_1

After having shown that there is no "small" arc-consistent representation of arbitrary XOR-clause-sets F, the task is to find "good" CNF-representations for special F. First we consider $c(F) = 1$, that is, a single XOR-clause C, to which we often refer as "$x_1 \oplus \cdots \oplus x_n = 0$". There is precisely one equivalent clause-set, i.e., there is exactly one representation without new variables, namely $\boldsymbol{X_0(C)} := \mathrm{prc}_0(x_1 \oplus \cdots \oplus x_n = 0)$, the set of prime implicates of the underlying boolean function, which is unique since these prime implicates are not resolvable. $X_0(C)$ has 2^{n-1} clauses for $n \geq 1$ (while for $n = 0$ we have $X_0(C) = \top$), namely precisely those full clauses (containing all variables) over $\{\mathrm{var}(x_1), \ldots, \mathrm{var}(x_n)\}$ where the parity of the number of complementations is different from the parity of the number of complementations in C. Note that for two XOR-clauses C, D we have $X_0(C) = X_0(D)$ iff C, D are equivalent. More generally, we define $X_0 : \mathcal{CLS} \to \mathcal{CLS}$, where the input is interpreted as XOR-clause-set and the output as CNF-clause-set, by $\boldsymbol{X_0(F)} := \bigcup_{C \in F} X_0(C)$. By Theorem 4:

Lemma 8. *If $F \in \mathcal{CLS}$ is acyclic, then $X_0(F) \in \mathcal{PC}$.*

An early and influential example of unsatisfiable clause-sets are the "Tseitin formulas" introduced in [26], which are obtained as applications of X_0 to XOR-clause-sets derived from graphs; see the underlying report [10] for various discussions. In [26] an exponential lower bound for regular resolution refutations of (special) Tseitin clause-sets was shown, and thus unsatisfiable Tseitin clause-sets in general have high hardness. This was extended in [27] to full resolution, and thus unsatisfiable Tseitin clause-sets in general have also high "w-hardness" (see [10]; as hardness captures tree-resolution, w-hardness captures dag-resolution).

In the following we refine $X_0 : \mathcal{CLS} \to \mathcal{CLS}$ in various ways, by first transforming an XOR-clause-set F into another XOR-clause-set F' representing F, and then using $X_0(F')$. If the XOR-clause-set F contains large clauses, then $X_0(F)$ is not feasible, and the XOR-clauses of F have to be broken up into short clauses, which we consider now. As we have defined how a CNF-clause-set can represent an XOR-clause-set, we can define that an XOR-clause-set F' represents an XOR-clause-set F, namely if the satisfying assignments of F' projected to the variables of F are precisely the satisfying assignments of F.

Definition 9. *Consider an XOR-clause $C = \{x_1, \ldots, x_n\} \in \mathcal{CL}$. The **natural splitting** of C is the XOR-clause-set F' obtained as follows, using $n := |C|$:*

- *If $n \leq 2$, then $F' := \{C\}$.*
- *Otherwise choose pairwise different new variables $y_2, \ldots, y_{n-1}$, and let $F' := \{x_1 \oplus x_2 = y_2\} \cup \{y_{i-1} \oplus x_i = y_i\}_{i \in \{3,\ldots,n-1\}} \cup \{y_{n-1} \oplus x_n = 0\}$, (i.e., $F' = \{\{x_1, x_2, y_2\}\} \cup \{\{y_{i-1}, x_i, y_i\}\}_{i \in \{3,\ldots,n-1\}} \cup \{\{y_{n-1}, x_n\}\}$).*

Then F' is, as XOR-clause-set, a representation of $\{C\}$. Let $\boldsymbol{X_1(C)} := X_0(F')$.

We have for $F := X_1(C)$: If $n \leq 2$, then $n(F) = c(F) = n$, and $\ell(F) = 2^{n-1} \cdot n$. Otherwise $n(F) = 2n - 2$, $c(F) = 4n - 6$ and $\ell(F) = 12n - 20$. Corollary 5, Part 2, applies to F' from Definition 9, and thus:

Lemma 10. *For $C \in \mathcal{CL}$ we have $X_1(C) \in \mathcal{PC}$.*

We define $X_1 : \mathcal{CLS} \to \mathcal{CLS}$, where the input is interpreted as XOR-clause-set and the output as CNF-clause-set, by $X_1(F) := \bigcup_{C \in F} X_1(C)$ for $F \in \mathcal{CLS}$, where some choice for the new variables is used, so that the new variables for different XOR-clauses do not overlap. By Theorem 4 and Lemma 10:

Theorem 11. *If $F \in \mathcal{CLS}$ is acyclic, then $X_1(F) \in \mathcal{PC}$.*

A precursor to Theorem 11 is found in Theorem 1 of [19], where it is stated that tree-like XOR clause-sets are "UP-deducible", which is precisely the assertion that for acyclic $F \in \mathcal{CLS}$ the representation $X_1(F)$ is arc-consistent. We now show that the problem of computing an arc-consistent CNF-representation for an XOR-clause-set F is fixed-parameter tractable in the parameter $c(F)$.

Theorem 12. *Consider a satisfiable XOR-clause-set $F \in \mathcal{CLS}$. Let $F^* := \{\oplus F' : F' \subseteq F\} \in \mathcal{CLS}$ (recall Lemma 6); F^* is computable in time $O(\ell(F) \cdot 2^{c(F)})$. Then $X_1(F^*)$ is a CNF-representation of F with $\mathrm{phd}^{\mathrm{var}(F)}(X_1(F^*)) \leq 1$.*

Proof. Consider some partial assignment φ with $\mathrm{var}(\varphi) \subseteq \mathrm{var}(F)$, let $F' := \mathrm{r}_1(\varphi * F^*)$, and assume there is a forced literal $x \in \mathrm{lit}(F')$ for F'. Then the XOR-clause $C := \{y \in \mathcal{LIT} : \varphi(y) = 0\} \cup \{\overline{x}\}$ follows from F. By Lemma 6 there is $F' \subseteq F$ with $\oplus F' = C$ modulo equivalence of XOR-clauses. So we have (modulo equivalence) $X_1(C) \subseteq F^*$, where due to $X_1(C) \in \mathcal{PC}$ (Lemma 10) the forced literal x for $\varphi * X_1(C)$ is set by r_1, contradicting the assumption. □

Theorem 4 in [22] yields the weaker bound $O(4^{n(F)})$ for the number of clauses in an arc-consistent representation of F (w.l.o.g. $c(F) \leq n(F)$). In Conjecture 14 we state our belief that we can strengthen Theorem 12 by establishing $\mathrm{phd}(F') \leq 1$ for an appropriate, more intelligent representation F' of F. We now turn to the problem of understanding and refining the basic translation X_1 for two clauses.

7 Translating two XOR-clauses

The analysis of the translation $X_1(\{C, D\})$ for two XOR-clauses C, D in the underlying report [10] shows, that this representation is very hard for tree-resolution, but easy for full and for width-restricted resolution. So it might be usable for (conflict-driven) SAT solvers. But indeed we can provide a representation in $\mathcal{PC}$ as follows; note that an XOR-clause-set $\{C, D\}$ is unsatisfiable iff $|C \cap \overline{D}|$ is odd and $\mathrm{var}(C) = \mathrm{var}(D)$.

Theorem 13. *Consider two XOR-clauses $C, D \in \mathcal{CL}$. To simplify the presentation, using $V := \mathrm{var}(C) \cap \mathrm{var}(D)$, we assume $|V| \geq 2$, and $|C| > |V|$ as well as $|D| > |V|$. Thus w.l.o.g. $|C \cap D| = |V|$. Let $I := C \cap D$. Choose $s \in \mathcal{VA} \setminus \mathrm{var}(\{C, D\})$, and let $I' := I \cup \{s\}$. Let $C' := (C \setminus I) \cup \{s\}$ and $D' := (D \setminus I) \cup \{s\}$. Now $\{I', C', D'\}$ is an XOR-clause-set which represents the XOR-clause-set $\{C, D\}$. Let $\boldsymbol{X_2(C, D)} := X_1(\{I', C', D'\})$. Then $X_2(C, D) \in \mathcal{PC}$ is a CNF-representation of the XOR-clause-set $\{C, D\}$.*

Proof. That $\{I', C', D'\}$ represents $\{C, D\}$ is obvious, since s is the sum of the common part. $\{I', C', D'\}$ is acyclic (besides the common variable s the three variable-sets are disjoint), and thus by Theorem 11 we get $X_2(C, D) \in \mathcal{PC}$. □

Conjecture 14. We can combine a generalisation of Theorem 13 with Theorem 12 and obtain $X_* : \mathcal{CLS} \to \mathcal{PC}$, which computes for an XOR-clause-set $F \in \mathcal{CLS}$ a CNF-representation $X_*(F)$ such that $\ell(X_*(F)) = 2^{O(c(F))} \cdot \ell(F)^{O(1)}$.

8 Open problems and future research directions

Regarding lower bounds, the main question for Theorem 7 is to obtain sharp bounds on the size of shortest representations F' with $\mathrm{phd}^{\mathrm{var}(F)}(F') \leq 1$. Turning to upper bounds, in Lemma 8, Theorem 11, and Theorem 13 we have established methods to obtain representations in $\mathcal{PC}$, while Conjecture 14 says, that computing a representation in $\mathcal{PC}$ should be fixed-parameter tractable in the number of XOR-clauses. See [10] for more open problems and conjectures.

References

1. Babai, L., Gál, A., Wigderson, A.: Superpolynomial lower bounds for monotone span programs. Combinatorica 19(3), 301–319 (March 1999)
2. Bessiere, C., Katsirelos, G., Narodytska, N., Walsh, T.: Circuit complexity and decompositions of global constraints. In: Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09). pp. 412–418 (2009)
3. Biere, A., Heule, M.J., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (February 2009)
4. Bordeaux, L., Marques-Silva, J.: Knowledge compilation with empowerment. In: Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turán, G. (eds.) SOFSEM 2012: Theory and Practice of Computer Science. Lecture Notes in Computer Science, vol. 7147, pp. 612–624. Springer (2012)
5. Courtois, N.T., Bard, G.V.: Algebraic cryptanalysis of the Data Encryption Standard. In: Galbraith, S.D. (ed.) 11th IMA International Conference on Cryptography and Coding. Lecture Notes in Computer Science, vol. 4887, pp. 152–169. Springer (2007)
6. Eén, N., Sörensson, N.: Translating pseudo-boolean constraints into SAT. Journal on Satisfiability, Boolean Modeling and Computation 2, 1–26 (March 2006)
7. Gent, I.P.: Arc consistency in SAT. In: van Harmelen, F. (ed.) 15th European Conference on Artificial Intelligence (ECAI 2002). pp. 121–125. IOS Press (2002)
8. Gwynne, M., Kullmann, O.: Generalising and unifying SLUR and unit-refutation completeness. In: van Emde Boas, P., Groen, F.C.A., Italiano, G.F., Nawrocki, J., Sack, H. (eds.) SOFSEM 2013: Theory and Practice of Computer Science. Lecture Notes in Computer Science (LNCS), vol. 7741, pp. 220–232. Springer (2013)
9. Gwynne, M., Kullmann, O.: Generalising unit-refutation completeness and SLUR via nested input resolution. Journal of Automated Reasoning (2013), to appear; published online 09 March 2013 `http://link.springer.com/article/10.1007/s10817-013-9275-8`

10. Gwynne, M., Kullmann, O.: On SAT representations of XOR constraints. Tech. Rep. arXiv:1309.3060v4 [cs.CC], arXiv (December 2013)
11. Gwynne, M., Kullmann, O.: Trading inference effort versus size in CNF knowledge compilation. Tech. Rep. arXiv:1310.5746v2 [cs.CC], arXiv (November 2013)
12. Haanpää, H., Järvisalo, M., Kaski, P., Niemelä, I.: Hard satisfiable clause sets for benchmarking equivalence reasoning techniques. Journal of Satisfiability, Boolean Modeling and Computation 2, 27–46 (March 2006)
13. Kleine Büning, H., Kullmann, O.: Minimal unsatisfiability and autarkies. In: Biere et al. [3], chap. 11, pp. 339–401
14. Kullmann, O.: Investigating a general hierarchy of polynomially decidable classes of CNF's based on short tree-like resolution proofs. Tech. Rep. TR99-041, Electronic Colloquium on Computational Complexity (ECCC) (October 1999)
15. Kullmann, O.: 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)
16. Kullmann, O.: Hardness measures and resolution lower bounds. Tech. Rep. arXiv:1310.7627v1 [cs.CC], arXiv (October 2013)
17. Laitinen, T., Junttila, T., Niemelä, I.: Extending clause learning DPLL with parity reasoning. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) ECAI 2010 – 19th European Conference on Artificial Intelligence. pp. 21–26. IOS Press (2010)
18. Laitinen, T., Junttila, T., Niemelä, I.: Equivalence class based parity reasoning with DPLL(XOR). In: ICTAI 2011 – 23rd International Conference on Tools with Artificial Intelligence. pp. 649–658 (2011)
19. Laitinen, T., Junttila, T., Niemelä, I.: Classifying and propagating parity constraints. In: Milano, M. (ed.) Principles and Practice of Constraint Programming – CP 2012. Lecture Notes in Computer Science, vol. 7514, pp. 357–372. Springer (2012)
20. Laitinen, T., Junttila, T., Niemelä, I.: Conflict-driven XOR-clause learning. In: Cimatti, A., Sebastiani, R. (eds.) Theory and Applications of Satisfiability Testing SAT 2012. Lecture Notes in Computer Science, vol. 7317, pp. 383–396. Springer (2012)
21. Laitinen, T., Junttila, T., Niemelä, I.: Extending clause learning SAT solvers with complete parity reasoning. In: ICTAI 2012 – 24th International Conference on Tools with Artificial Intelligence. pp. 65–72 (2012)
22. Laitinen, T., Junttila, T., Niemelä, I.: Simulating parity reasoning. In: Logic for Programming Artificial Intelligence and Reasoning – LPAR 2013. LNCS Advanced Research in Computing and Software Science, Springer (2013)
23. Li, C.M.: Equivalency reasoning to solve a class of hard SAT problems. Information Processing Letters 76(1), 75–81 (2000)
24. Rossi, F., van Beek, P., Walsh, T. (eds.): Handbook of Constraint Programming. Foundations of Artificial Intelligence, Elsevier (2006)
25. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: Kullmann, O. (ed.) Theory and Applications of Satisfiability Testing - SAT 2009. Lecture Notes in Computer Science, vol. 5584, pp. 244–257. Springer (2009)
26. Tseitin, G.: On the complexity of derivation in propositional calculus. In: Seminars in Mathematics, vol. 8. V.A. Steklov Mathematical Institute, Leningrad (1968), english translation: Studies in mathematics and mathematical logic, Part II (A.O. Slisenko, editor), 1970, pages 115-125
27. Urquhart, A.: Hard examples for resolution. Journal of the ACM 34, 209–219 (1987)