Artificial Intelligence 203 (2013) 19–34

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Complexity issues related to propagation completeness

Martin Babka, Tomáš Balyo, Ondřej Čepek, Štefan Gurský, Petr Kučera *, Václav Vlček

Department of Theoretical Computer Science and Mathematical Logic, Faculty of Mathematics and Physics, Charles University in Prague, Malostranské nám. 25, 118 00 Praha 1, Czech Republic

ARTICLE INFO

Article history:
Received 10 July 2012
Received in revised form 14 June 2013
Accepted 30 July 2013
Available online 7 August 2013

Keywords:
Boolean functions
Satisfiability
Knowledge compilation
Empowering implicates
Unit propagation
Propagation completeness

ABSTRACT

Knowledge compilation is a process of adding more information to a knowledge base in order to make it easier to deduce facts from the compiled base than from the original one. One type of knowledge compilation occurs when the knowledge in question is represented by a Boolean formula in conjunctive normal form (CNF). The goal of knowledge compilation in this case is to add clauses to the input CNF until a logically equivalent propagation complete CNF is obtained. A CNF is called propagation complete if after any partial substitution of truth values all logically entailed literals can be inferred from the resulting CNF formula by unit propagation. The key to this type of knowledge compilation is the ability to generate so-called empowering clauses. A clause is empowering for a CNF if it is an implicate and for some partial substitution of truth values it enlarges the set of entailed literals inferable by unit propagation.

In this paper we study several complexity issues related to empowering implicates, propagation completeness, and its relation to resolution proofs. We show several results: (a) given a CNF and a clause it is co-NP complete to decide whether the clause is an empowering implicate of the CNF, (b) given a CNF it is NP-complete to decide whether there exists an empowering implicate for it and thus it is co-NP complete to decide whether a CNF is propagation complete, and (c) there exist CNFs to which an exponential number of clauses must be added to make them propagation complete.

1. Introduction

One of the most studied problems in computer science, both theoretical and applied, is the satisfiability problem for CNF formulas (SAT). The difficulty of SAT depends on the class of CNF formulas to which the input formula belongs. There are various techniques and algorithms for SAT for different classes of CNF formulas ranging from linear algorithms for Horn, quadratic (2-CNF) and SLUR formulas [1,2] to the very complex variants of the exponential DPLL [3,4] and CDCL [5–8] procedures implemented in general purpose SAT solvers. Even the most complicated SAT solvers usually perform a task called unit propagation [3]. The goal of unit propagation is to infer as many logically entailed literals as possible from a partial truth assignment and the input formula. Although in general unit propagation is not a complete method (it does not infer all logically entailed literals), it is complete for the class of *propagation complete* (*PC*) CNF formulas [9].

PC formulas play an important role also in constraint programming, or more specifically, in CNF encodings of global constraints. There is a strong connection between propagation completeness of the CNF encoding and domain consistency

* Corresponding author. Tel.: +420 221 914 138; fax: +420 221 914 323.
E-mail addresses: babkys@gmail.com (M. Babka), biotomas@gmail.com (T. Balyo), ondrej.cepek@mff.cuni.cz (O. Čepek), stevko@mail.ru (Š. Gurský), kucerap@ktiml.mff.cuni.cz (P. Kučera), vlcek@ktiml.mff.cuni.cz (V. Vlček).

0004-3702/$ – see front matter © 2013 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.artint.2013.07.006

of the encoded constraint [10,11]. It has been studied for several concrete global constraints such as the AllDifferent constraint [12], the Sequence constraint [13], Regular, Among, and Generalized Sequence [10], or the Grammar constraints [14].

Some SAT solvers try to avoid searching in the state subspaces with no solution by learning from conflicts, i.e. by performing *conflict driven clause learning* (*CDCL*) [5–8], the name CDCL is also used for the complete algorithm solving SAT problem. It is useful to learn clauses (called *empowering implicates* [9,15]) that allow unit propagation to infer more logically entailed literals after such a clause is added to the CNF formula than it was possible to infer before the addition. Therefore, to speed up the CDCL SAT solver search for a satisfying assignment, it is often very useful to learn (generate) empowering implicates and add them to the input CNF formula. Let us mention that today's most successful SAT solvers for real-world applications are the ones using CDCL procedure.

This process of adding empowering implicates to a CNF formula can be viewed as a special type of knowledge compilation where both the input and the output representation of the knowledge is a CNF formula. In general, knowledge compilation is a process of adding more information to a given knowledge representation in order to make it computationally easier to infer facts from the compiled representation [16,17], or a process of transforming a given knowledge representation into another knowledge representation which is more tractable with respect to fact deduction, such as transforming a CNF into a BDD [18]. Nevertheless, in this paper we are interested only in the very limited case of knowledge compilation that rests in adding empowering implicates to a CNF.

It has been shown in [9], along with other properties of PC formulas, that a formula φ is PC if and only if there is no empowering implicate for φ. However, several complexity issues directly connected to propagation completeness and empowering implicates are left open in [9]. A short list of such questions is the following:

1. Given a CNF formula φ and a clause C, what is the complexity of deciding whether C is an empowering implicate for φ?
2. Given a CNF formula φ that is not PC, how difficult is it to generate an empowering implicate for φ by resolution, where the "level of difficulty" is measured by the length of the resolution proof?
3. Given a CNF formula φ, what is the complexity of deciding whether there exists an empowering implicate for φ?
4. Given a CNF formula φ that is not PC, how many empowering implicates is it necessary to add to φ in order to make it PC?

In this paper we tackle all of the above listed problems. After reviewing basic definitions and notation in Section 2, we derive several simple properties of empowering implicates in Section 3. We address the following four questions as follows:

1. In Section 3 we show that the first problem is co-NP complete. This is not a very difficult result, however, to the best of our knowledge, it was not stated in the related literature yet.
2. In Section 4 we tackle the second problem. We prove that for a non-PC CNF formula with s occurrences of literals there always exists a resolution proof of length $O(s)$ of some empowering implicate. On the other hand, we construct examples of CNF formulas where a resolution proof of length $\Omega(s)$ is needed for any empowering implicate, which means that $\Theta(s)$ is an asymptotically tight bound for this problem. It is important to note that the upper bound result does not require the derived empowering implicate to be prime. We show (by a simple modification of results concerning refutation proofs [19,20]) that there exist CNF formulas such that in order to derive any prime empowering implicate of such CNF a resolution proof of an exponential length is needed.
3. Section 5 contains the main results of this paper which are connected to the third problem. It was proved in [9] that deciding about an existence of an empowering implicate is in Σ_2^p. Using the results from Section 4 we strengthen this result by showing that the problem belongs to $\Sigma_1^p = \mathrm{NP}$. Given the equivalence between propagation completeness and non-existence of empowering implicates proved in [9], this immediately implies that testing propagation completeness belongs to co-NP. Then we proceed with the hardness proof for this problem. We present a reduction from a well-known NP-complete 3-dimensional matching problem which proves that deciding for a CNF formula whether there exists an empowering implicate for it is NP-hard (and thus testing propagation completeness is coNP-hard).
4. The fourth question is answered in Section 5 as well by showing that there exist CNF formulas where an exponential number (both with respect to the number of variables and the number of clauses) of empowering implicates must be added in order to arrive at a PC formula. This strengthens the superpolynomial bound which follows from a combination of results in [9] and [21] using a superpolynomial lower bound for certain monotone circuits from [22]. The connection is discussed in detail in Section 2.5.

We close the paper by giving few concluding remarks in Section 6.

2. Definitions

2.1. Basic definitions

A *Boolean function of n variables* is a mapping $f : \{0,1\}^n \to \{0,1\}$. We say that a Boolean function f is *satisfiable* if there is a vector $\vec{x} \in \{0,1\}^n$ such that $f(\vec{x}) = 1$. A *literal* is either a variable (x, called *positive literal*) or its negation ($\neg x$ or $\bar{x}$, 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. A clause which contains just one literal is called a *unit clause*. Formula φ is in *conjunctive normal form* (*CNF*) if it is a conjunction 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. It is a well-known fact that every Boolean function can be represented by a CNF formula (see e.g. [23]). A CNF formula φ is called a *Horn formula* if every clause in φ has at most one positive literal. A *quadratic CNF formula* (also called 2-CNF formula) is a CNF formula where each clause contains at most two literals. 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 $\varphi_1 \equiv \varphi_2$.

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 use notation $f \models C$ to denote that C is an implicate of f. We also say that C is *logically entailed* by f. Since every CNF formula φ represents a function, we shall often use $\varphi \models C$ to denote the fact that C is an implicate of the function represented by φ. We shall also say that C is an implicate of φ or that it is logically entailed by φ. A clause is a special case of a CNF formula and thus for two clauses C_1 and C_2 we can also use $C_1 \models C_2$. This is possible only if literals contained in C_1 are also contained in C_2. In this case we say that *clause* C_1 *subsumes clause* 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. We say that CNF formula φ is *prime* if it contains only prime implicates. A special case of prime CNF formula is the *canonical CNF formula of* f (also called a canonical representation of f), which consists of all the prime implicates of f.

Given a CNF formula φ and a clause C we define IsImplicate(φ, C) as the problem of deciding whether C is an implicate of φ. It is well known that this problem is co-NP complete (co-SAT is a special case of IsImplicate(φ, C) in which C is an empty clause).

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 $C_2 = (\widetilde{C_2} \vee \bar{x})$ 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. A resolution in which one of the parent clauses is a unit clause is called a *unit resolution*. A resolution in which the parent clauses have no common literal is called a *non-merge resolution*, otherwise it is a *merge resolution*.

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, $1 \leqslant i \leqslant k$, 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. Resolution proof of a contradiction (i.e. an empty clause $\perp$) from formula φ is also called a *resolution refutation*. A resolution proof in which every resolution is unit is called a *unit resolution proof*, a unit resolution proof of a contradiction $\perp$ from φ is called a *unit refutation* and if such a proof exists for φ, then this formula is called *unit refutable*. This fact is denoted by $\varphi \vdash_1 \perp$. If a unit clause (literal) x can be derived by unit resolutions then we write $\varphi \vdash_1 x$. 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 (see e.g. [24]). Another well known fact is that every prime implicate of f can be derived from φ by a series of resolutions (see e.g. [24]).

2.3. Unit propagation and refutation

Unit propagation is an iterative procedure which in each step selects a unit clause (a literal), removes each clause containing this literal, and removes the complementary literal from the remaining clauses (i.e. satisfies the selected literal). This process iterates until an empty clause (a contradiction) is derived or there is no unit clause in the formula. Unit propagation can be performed in linear time [25]. It is a well-known fact that if $\varphi \vdash_1 x$ (where x is an arbitrary literal), then unit propagation on φ satisfies x or derives contradiction. On the other hand, if x is satisfied during unit propagation, then $\varphi \vdash_1 x$.

We say that a literal x is logically entailed from φ by a partial assignment $l_1, \ldots, l_k$ (by setting literals $l_1, \ldots, l_k$ to true) if any assignment that extends $l_1, \ldots, l_k$ and satisfies φ sets x to true. Note that this is equivalent to $\varphi \wedge l_1 \wedge \cdots \wedge l_k \models x$. Clearly, literals that form unit clauses in φ are logically entailed from φ. Also note that given clause $C = l_1 \vee l_2 \vee \cdots \vee l_k$, formula $\varphi \wedge \bigwedge_{j=1}^{k} \neg l_j$ is by De Morgan's laws equivalent to $\varphi \wedge \neg C$. In the subsequent text we shall use these notations interchangeably.

A clause C is called 1-*provable* with respect to a CNF formula φ, if $\varphi \wedge \neg C \vdash_1 \perp$. A clause C is thus 1-provable if it is an implicate of C and the fact that C is an implicate can be proved using unit propagation. The notion of 1-provability was introduced in [15]. A formula φ is called *unit refutation complete* if every implicate C of φ is 1-provable with respect to φ. The notion of unit refutation completeness was introduced in [26] and later inspired the definition of propagation completeness in [9], which we give in the following subsection.

2.4. Propagation completeness and empowerement

Definition 2.1 *(Propagation Completeness (PC) [9]).* We call a CNF formula φ propagation complete (PC) if for any partial assignment $l_1, \ldots, l_k$ and any literal d the following holds. If d is logically entailed from φ by $l_1, \ldots, l_k$, then d can be derived from φ by unit propagation after fixing values of $l_1, \ldots, l_k$, i.e., if $\varphi \wedge l_1 \wedge \cdots \wedge l_k \models d$, then $\varphi \wedge l_1 \wedge \cdots \wedge l_k \vdash_1 d$. The class PC is the class of all PC formulas.

Definition 2.2 *(Empowering implicate [15,9], Absorption [27,9]).* A clause $C = l_1 \vee l_2 \vee l_3 \vee \cdots \vee l_k$ is called an *empowering implicate* for a formula φ if it contains a literal l_i called an *empowered literal* such that $\varphi \wedge \bigwedge_{j \in 1 \ldots k,\ j \neq i} \neg l_j \nvdash_1 \bot$ and $\varphi \wedge \bigwedge_{j \in 1 \ldots k,\ j \neq i} \neg l_j \models l_i$ but $\varphi \wedge \bigwedge_{j \in 1 \ldots k,\ j \neq i} \neg l_j \nvdash_1 l_i$. An implicate C is called *absorbed* by φ if it has no empowered literal.

It is known that a formula is propagation complete if and only if it has no empowering implicates [9]. Thus, any formula can be extended to a propagation complete one by repeatedly adding empowering implicates.

The notion of empowering clauses is closely related to CDCL SAT solvers, see [15] and [26]. It is important to note that each asserting clause is empowering as observed in [15]. Since most CDCL SAT solvers learn only asserting clauses the notion of propagation completeness is important also in the context of CDCL SAT solvers.

Example 2.3. Consider the following formula

$$\varphi = (a \vee b \vee x) \wedge (a \vee c \vee \neg x).$$

Formula φ is not propagation complete, because $\varphi \wedge \neg b \wedge \neg c \models a$ but $\varphi \wedge \neg b \wedge \neg c \nvdash_1 a$. Indeed, by falsifying b and c we get $(a \vee x) \wedge (a \vee \neg x)$ and thus literal a can be obtained by resolution, but not by unit propagation.

It follows that $(a \vee b \vee c)$ is an empowering implicate for φ with a being its empowered literal. By adding this implicate to φ we get a propagation complete formula

$$\varphi' = (a \vee b \vee x) \wedge (a \vee c \vee \neg x) \wedge (a \vee b \vee c).$$

The fact that φ' is propagation complete follows from the fact that it consists of all prime implicates of φ. Consider for example a non-prime implicate $C = (a \vee b \vee c \vee x)$, this implicate is absorbed by φ' because by falsifying any three literals from C we either get an unsatisfiable formula (such as if a, b, and x are falsified) or we obtain the remaining literal by unit propagation (such as if a, c, and x are falsified, b is obtained from the first clause of φ'). We can observe that in fact, C is already absorbed by φ.

2.5. Relation of propagation completeness to constraint propagation

Propagation complete formulas play an important role in constraint propagation (for general reading on constraint propagation see Chapter 3 of [28]), where the notion appeared implicitly in earlier literature [10–14,29] concentrated on CNF encodings of constraints both in general [10,11], and for particular constraints (the sequence constraint [13], the grammar constraint [14], the Regular, the Among, and the Generalized sequence constraints [11], or the AllDifferent constraint [12]). A general idea which appeared in all these papers is to take a constraint C, encode it using a CNF and then use unit propagation to maintain some kind of domain consistency (e.g. generalized arc consistency (GAC) in [10], or (relational) (i, j)-consistency in [29]). In general, these encodings are polynomial in table representation of a constraint, however in the case of a particular global constraint we can sometimes use its special properties to get a CNF representation of polynomial size with respect to the arity of a constraint with suitable properties, such as in [11] in case of the Regular constraint, the Among constraint, or the Generalized sequence constraint. On the other hand in some cases this is not possible as it is shown in [12] for the AllDifferent constraint.

The idea of using unit propagation to maintain domain consistencies was formalized in [12], let us recall some of the results and the notation from [12] here. A *constraint* C is defined over a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, each of which has a finite domain $D(X_i)$. An *assignment* to a variable X_i is a mapping of X_i to a value $j \in D(X_i)$, called a literal and written $X_i = j$. $\mathbf{D}(\mathbf{X})$ denotes the set of literals, i.e. $\mathbf{D}(\mathbf{X}) = \{X_i = j \mid X_i \in \mathbf{X} \wedge j \in D(X_i)\}$. $\mathcal{P}(\mathbf{D}(\mathbf{X}))$ then denotes the set of all possible sets of literals. A constraint C is defined over a set of variables denoted as $scope(C) \subseteq \mathbf{X}$ (*scope* of C) and it allows a subset of the possible assignments to the variables in $scope(C)$.

Following [12,30,28] a propagator for a constraint C is an algorithm which takes as input the domains of variables in $scope(C)$ and returns restrictions of these domains. Formally a propagator for a constraint C can be defined as follows (definition is taken from [12], where they followed [30], see also Chapter 3 of [28]):

Definition 2.4. (See Definition 1 in [12].) A *propagator* f for a constraint C is a polynomial time computable function $f : \mathcal{P}(\mathbf{D}(\mathbf{X})) \mapsto \mathcal{P}(\mathbf{D}(\mathbf{X}))$, such that f is *monotone*, i.e. $\mathbf{D}'(\mathbf{X}) \subseteq \mathbf{D}(\mathbf{X}) \Rightarrow f(\mathbf{D}'(\mathbf{X})) \subseteq f(\mathbf{D}(\mathbf{X}))$, *contracting*, i.e. $f(\mathbf{D}(\mathbf{X})) \subseteq \mathbf{D}(\mathbf{X})$, and *idempotent*, i.e. $f(\mathbf{D}(\mathbf{X})) = f(f(\mathbf{D}(\mathbf{X})))$. If f detects that C has no solutions under $\mathbf{D}(\mathbf{X})$ then $f(\mathbf{D}(\mathbf{X})) = \emptyset$.

We say that a propagator *detects dis-entailment* if $f(\mathbf{D}(\mathbf{X})) = \emptyset$ whenever C has no solution. A propagator enforces *domain consistency* (*DC*) when $X_i = j \in f(\mathbf{D}(\mathbf{X}))$ implies that there exists a solution of C that contains $X_i = j$. Other consistencies (such as GAC [10] which is equivalent to DC, or (i, j)-consistency [29]) can be considered as well, we mention domain consistency here because later on we shall recall the results of [12] on domain consistency propagators.

Let us recall a definition of a CNF decomposition of a propagator as it appeared in [12]. Before that let us recall a general way how a constraint satisfaction program (CSP) variables with multiple valued domains are usually encoded within a CNF. Given a variable $X_i \in \mathbf{X}$ with domain $D(X_i)$ we encode it with a set of boolean variables $x_{i,j}$, $X_i \in \mathbf{X}$, $j \in D(X_i)$ such that $X_i \neq j \Leftrightarrow \bar{x}_{i,j}$. The property that a CSP variable X_i has at most one value is enforced by the set of *AMO* (i.e. *at most one*) clauses $(\bar{x}_{i,j} \vee \bar{x}_{i,k})$ for all $j, k \in D(X_i)$, $k \neq j$, and the property that it has at least one value is enforced by the *ALO* (i.e. *at least one*) clause $\bigvee_{j \in D(X_i)} x_{i,j}$. Following [12] we call the above described propositional representation of $\mathbf{D}((X))$ (i.e. set of all AMO and ALO clauses over all CSP variables) a *direct encoding* and denote it as $\mathbf{D}^{sat}(\mathbf{X})$.

Definition 2.5. (See Definition 4 in [12].) A CNF decomposition of a propagation algorithm (propagator) f_P is a formula in CNF φ_P over variables $\mathbf{x} \cup \mathbf{y}$ such that

- The *input variables* $\mathbf{x}$ are the propositional representation $\mathbf{D}^{sat}(\mathbf{X})$ of $\mathbf{D}(\mathbf{X})$ and $\mathbf{y}$ is a set of *auxiliary variables* whose size is polynomial in $|x|$.
- $x_{i,j}$ is set to 0 by a unit propagation if and only if $X_i = j \notin f_P(\mathbf{D}(\mathbf{X}))$.
- Unit propagation on φ_P produces the empty clause when $f_P(\mathbf{D}(\mathbf{X})) = \emptyset$.

Before further discussion, let us look at an example from [12].

Example 2.6. (See Example 1 in [12].) Consider a Table constraint over the variables X_1, X_2 with $D(X_1) = D(X_2) = \{a, b\}$ and the satisfying assignments: $\{\langle a, a\rangle, \langle b, b\rangle, \langle a, b\rangle\}$. Using encoding introduced in [10] we can decompose a Table constraint into the following CNF φ_T:

$$\begin{aligned}\varphi_T = {} & (\bar{x}_{1a} \vee y_1 \vee y_3) \wedge (\bar{x}_{2a} \vee y_1) \wedge (\bar{y}_1 \vee x_{1a}) \wedge (\bar{y}_1 \vee x_{2a}) \\ & \wedge (\bar{x}_{1b} \vee y_2) \wedge (\bar{x}_{2b} \vee y_2 \vee y_3) \wedge (\bar{y}_2 \vee x_{1b}) \wedge (\bar{y}_2 \vee x_{2b}) \\ & \wedge (\bar{y}_3 \vee x_{1a}) \wedge (\bar{y}_3 \vee x_{2b}) \wedge (y_1 \vee y_2 \vee y_3).\end{aligned}$$

Here $\mathbf{y} = \{y_1, y_2, y_3\}$ consists of auxiliary variables corresponding to the three possible solutions to the Table constraint (y_1 corresponds to $\langle a, a\rangle$, y_2 to $\langle b, b\rangle$, and y_3 to $\langle a, b\rangle$). Suppose the value a is removed from the domain of X_1. The assignment $x_{1a} = 0$ forces the variable y_1 to 0, which in turn causes the variable x_{2a} to 0, removing the value a from the domain of X_2 as well.

Now let us generalize the ideas presented in the above example. Let us consider a constraint C, its consistency propagator f_P, and a CNF decomposition of f_P via a CNF φ_P.

- By definition, if f_P detects dis-entailment, then $f_P(\mathbf{D}(\mathbf{X})) = \emptyset$ whenever C restricted to $\mathbf{D}(\mathbf{X})$ admits no solution. Passing $\mathbf{D}(\mathbf{X})$ to f_P corresponds to setting all values $x_{i,j} = 0$ for $X_i = j \notin \mathbf{D}(\mathbf{X})$. The third condition of Definition 2.5 thus requires that after this partial assignment φ_P is not only unsatisfiable but this fact can be detected by unit propagation. In particular we require that for any $\mathbf{D}(\mathbf{X})$:

$$\varphi_P \wedge \bigwedge_{i,j:\ X_i = j \notin \mathbf{D}(\mathbf{X})} \neg x_{i,j} \models \bot \quad \Leftrightarrow \quad \varphi_P \wedge \bigwedge_{i,j:\ X_i = j \notin \mathbf{D}(\mathbf{X})} \neg x_{i,j} \vdash_1 \bot.$$

  What we in fact require here is that φ_P is unit refutation complete with respect to the partial assignments to the input variables $\mathbf{x}$. Although we admit here only assignments to 0, the direct encoding clauses $\mathbf{D}^{sat}(\mathbf{X})$ allow us to use 1 as well (assigning $x_{i,j} = 1$ forces $x_{i,k} = 0$ for any $k \neq j$, $k \in D(X_i)$ by unit propagation on $\mathbf{D}^{sat}(\mathbf{X})$).
- If f_P is a domain consistency propagator then there is no solution to C containing $X_i = j$ if and only if $X_i = j \notin f(\mathbf{D}(\mathbf{X}))$. Using the second condition from Definition 2.5 it corresponds to the fact that $\bar{x}_{i,j}$ is a unit implicate of φ_P under partial assignment given by $\mathbf{D}(\mathbf{X})$ if and only if $x_{i,j}$ is forced to 0 by unit propagation on φ_P under this partial assignment. In particular we require that for any $\mathbf{D}(\mathbf{X})$ and any $x_{i,j}$:

$$\varphi_P \wedge \bigwedge_{i',j':\ X_{i'} = j' \notin \mathbf{D}(\mathbf{X})} \neg x_{i',j'} \models \neg x_{i,j} \quad \Leftrightarrow \quad \varphi_P \wedge \bigwedge_{i',j':\ X_{i'} = j' \notin \mathbf{D}(\mathbf{X})} \neg x_{i',j'} \vdash_1 \neg x_{i,j}.$$

  What we in fact require here is the fact that φ_P is propagation complete with respect to partial assignments and literals on the input variables in $\mathbf{x}$. In the above equation it is enough to consider only negative literals $x_{i,j}$ due to presence of clauses of direct encoding $\mathbf{D}^{sat}(\mathbf{X})$.

The author of [10] suggests that a canonical CNF decomposition (i.e. CNF consisting of all prime implicates) is sufficient to encode both dis-entailment detecting propagator and domain consistency propagator. The above discussion shows that a shorter CNF is sufficient, in particular unit refutation completeness is sufficient for detecting dis-entailment and propagation completeness is sufficient for domain consistency propagator. This can be naturally generalized to other consistencies such as (i, j)-consistency [29].

It was shown in [21] that a polynomial sized decomposition of a consistency propagator f_P exists if and only if it can by computed by a monotone circuit of polynomial size. This result was used to derive the following corollary:

Corollary 2.7. *(See Corollary 4 in [21].) There is no polynomial sized CNF decomposition of any* AllDifferent *domain consistency propagator.*

On the other hand we can get a polynomial sized CNF encoding of the AllDifferent constraint. Using such an encoding and Corollary 2.7 it follows that no propagation complete encoding of the AllDifferent constraint can have polynomial size. There are thus formulas such that any equivalent propagation complete formula has superpolynomial size. Let us have a more detailed look at the proof of Corollary 2.7 in [21]. It was shown in [31] that AllDifferent constraint has a solution if and only if the corresponding bipartite value graph has a perfect matching. From the aformentioned connection to monotone circuits proved in [21] it follows that based on an AllDifferent domain consistency propagator we can construct a monotone circuit that computes whether a bipartite graph has a perfect matching and such a circuit has a polynomial size with respect to the domain consistency propagator we start with. The proof is finished using an older result of Razborov [22] according to which the size of monotone circuit computing whether there is a perfect matching in a bipartite graph G on n vertices has size at least $n^{\Omega(\log n)}$, which is a superpolynomial (in some literature called quasi-polynomial) but not an exponential bound. In Section 5 we strengthen the result of Corollary 2.7 for propagation complete formulas by showing that there are in fact formulas to which an exponential number of implicates have to be added in order to make them propagation complete.

3. Properties of empowering implicates

Let us start this section with a discussion on connection between unit refutation completeness introduced in [26] and propagation completeness introduced in [9]. First let us recall that a formula φ is unit refutation complete if for every implicate C of φ we have that $\varphi \wedge \neg C \vdash_1 \bot$. This means that the fact that C is an implicate of φ can be proved using just unit resolution. It is not hard to see that if a formula φ is propagation complete, then it is unit refutation complete as well. Indeed, if $C = (l_1 \vee \cdots \vee l_k)$ is an implicate of φ, then either $\varphi \wedge \bigwedge_{i=1}^{k-1} \neg l_i \vdash_1 \bot$, or $\varphi \wedge \bigwedge_{i=1}^{k-1} \neg l_i \vdash_1 l_k$ which implies $\varphi \wedge \bigwedge_{i=1}^{k} \neg l_i \vdash_1 \bot$. On the other hand, it is not true that every unit refutation complete formula is also propagation complete. Consider the following formula φ:

$$\varphi = (\bar{x} \vee a \vee b \vee c) \wedge (x \vee a \vee b \vee d).$$

This is a prime CNF which has only one more prime implicate $C = (a \vee b \vee c \vee d)$ produced by resolving the two clauses in φ. We can observe that φ is unit refutation complete. If we add negation of C to φ, then unit resolution of $\bar{x}$ and x gives us the empty clause. However φ is not propagation complete and C is an empowering implicate of φ with empowered literal a (or b). Indeed, if we set b, c, and d to false in φ, we get CNF $(\bar{x} \vee a) \wedge (x \vee a)$ from which it is not possible to derive a just using unit resolution.

It follows from the above discussion that the set of propagation complete CNFs is a proper subset of unit refutation complete CNFs. The aim of this section is to recall several results about the class of unit refutation complete CNFs from [26] and observe that some of these results are in fact true already for the set of propagation complete formulas.

We shall start with showing that adding a literal to a non-empowering implicate cannot make it empowering with respect to any of the original literals.

Lemma 3.1. *Let φ be a CNF formula and let $C = l_1 \vee \cdots \vee l_k$ be an implicate of φ which is not empowering. Let A be a clause and let $l_i \in C$ be an arbitrary literal. Then $C \vee A$ is not empowering implicate of φ with empowered literal l_i.*

Proof. Let us assume without loss of generality that $i = k$ (if not, then we can achieve this by renaming the variables). By definition, the fact that C is not empowering with empowered literal l_k implies that either $\varphi \wedge \bigwedge_{j=1}^{k-1} \neg l_j \vdash_1 \bot$, or $\varphi \wedge \bigwedge_{j=1}^{k-1} \neg l_j \vdash_1 l_k$ because C is an implicate of φ and thus $\varphi \wedge \bigwedge_{j=1}^{k-1} \neg l_j \models l_k$. Since $\varphi \wedge \bigwedge_{j=1}^{k-1} \neg l_j$ is a subformula of $\varphi \wedge \bigwedge_{j=1}^{k-1} \neg l_j \wedge \bigwedge_{a \in A} \neg a$, what we can derive by unit propagation from the former formula, can be derived from latter one as well. Thus we have that $\varphi \wedge \bigwedge_{j=1}^{k-1} \neg l_j \wedge \bigwedge_{a \in A} \neg a \vdash_1 \bot$ or $\varphi \wedge \bigwedge_{j=1}^{k-1} \neg l_j \wedge \bigwedge_{a \in A} \neg a \vdash_1 l_k$. This means that by definition, $C \vee A$ is not empowering implicate with empowered literal l_k. □

Note that in the previous proposition we cannot argue that $A \vee C$ is not empowering because it could be empowering with an empowered literal from A, e.g. if A would itself be an empowering clause. That is why we consider only literals

in C. Using Lemma 3.1, we can easily show that among empowering implicates the prime implicates are the only ones we need to consider.

Lemma 3.2. *Let φ be a nonempty satisfiable CNF formula (i.e. it has at least one nonempty clause) and let C be an empowering implicate for φ. Then any implicate C' of φ subsuming C is empowering for φ (this in particular includes the case when C' is prime).*

Proof. Let C' be an arbitrary implicate subsuming C and let us assume that $C' \neq C$. If C' is not empowering, then $\varphi \wedge \neg C' \vdash_1 \bot$ and thus C cannot be empowering with respect to a literal a, which is not in C'. On the other hand, C cannot be empowering with respect to a literal in C' as well due to Lemma 3.1. □

As an easy corollary we now get that a canonical CNF formula is always propagation complete, although this can be easily deduced from properties of canonical CNFs as well.

Now let us consider the problem of generating an empowering implicate for a given CNF formula. A natural method to consider is to generate an empowering implicate by the resolution procedure. In [26] it is shown that non-merge resolution cannot produce an empowering implicate with respect to unit refutation completeness. In case of propagation completeness the same is true. It follows from discussion at the beginning of Section 5 in [9]. We shall formulate this proposition as a lemma to be able to reference to it later.

Lemma 3.3. *Let φ be a CNF formula and let C be produced from φ by a series of non-merge resolutions. Then C is not an empowering implicate of φ.*

Keeping in mind that a canonical formula is propagation complete, it follows that a CNF formula φ satisfying that every prime implicate of φ is either present in it or it can be derived from φ by a series of non-merge resolutions is always propagation complete. This property was already shown in form of Theorem 1 in [26] for unit refutation completeness and it is an easy corollary of arguments about non-merge resolution in [9] (here stated as Lemma 3.3) that it is true also for PC formulas.

Since PC formulas allow easy inference, it is interesting to investigate classes which are contained in the class of PC CNF formulas. One such example is given by the following theorem which shows the desired property for the class of prime quadratic CNF formulas.

Theorem 3.4. *If φ is a prime quadratic CNF formula, then it is propagation complete.*

Proof. If φ is not satisfiable, then the proposition of the theorem is trivial. Let us assume that φ is a satisfiable prime quadratic CNF formula.

By Lemma 3.2 it is enough to consider prime implicates as candidates for empowering implicates. Because φ is a prime CNF formula, it must contain all the unit prime implicates. Hence, if any other prime implicate should be added to φ to make it propagation complete, it must be a quadratic clause which is produced by resolving two other quadratic clauses. It is a simple observation that these resolutions have to be non-merge. Thus by Lemma 3.3 we have that φ must already be propagation complete. □

The following lemma shows that the primeness assumption in the statement of Theorem 3.4 is necessary.

Lemma 3.5. *There is a (non-prime) quadratic CNF formula which is not PC.*

Proof. Let us consider the following CNF formula:

$$\varphi = (\bar{a} \vee b) \wedge (\bar{a} \vee \bar{b}).$$

φ is clearly a quadratic CNF formula. On the other hand, φ is not PC because $\varphi \models \bar{a}$, but $\varphi \nvdash_1 \bar{a}$. □

Now let us turn our attention to the complexity of testing if a given clause C is an empowering implicate of a CNF formula φ. We shall denote this problem as IsEmpowering(φ, C). Note that co-NP completeness of IsEmpowering(φ, C) comes as no surprise since it is in essence very similar to another co-NP complete problem IsImplicate(φ, C). The hard part of checking whether given clause C is an empowering implicate of φ is in fact checking whether C is an implicate of φ at all. Thus the co-NP completeness of IsEmpowering(φ, C) is a direct consequence of co-NP completeness of IsImplicate(φ, C). The proof of the following theorem only formalizes this idea.

Theorem 3.6. *The problem* IsEmpowering(φ, C) *is co-NP complete.*

Proof. To show that a problem is in co-NP it suffices to have for every negative instance of the problem a polynomially verifiable certificate which allows to verify that the answer is no. We can distinguish two cases when a pair (φ, C) forms a negative instance of IsEmpowering. In the first case C is not even an implicate of φ and the desired certificate is then an assignment of truth values to the variables which satisfies φ and falsifies C. The second case is when C is an implicate of φ but not an empowering one. In this case an empty certificate is good enough because one needs no additional information to be able to check in polynomial time that no literal in C is empowered. This can be done by running unit propagation and checking for every literal ℓ in C that $\varphi \wedge \bigwedge_{\ell' \in C,\, \ell' \neq \ell} \neg\ell' \vdash_1 \bot$ or $\varphi \wedge \bigwedge_{\ell' \in C,\, \ell' \neq \ell} \neg\ell' \vdash_1 \ell$.

For the co-NP hardness we reduce the co-NP complete problem IsImplicate to IsEmpowering. Let (φ, C) be an arbitrary instance of IsImplicate. We start by a simple preprocessing step in which we run unit propagation to test whether $\varphi \wedge \bigwedge_{\ell \in C} \neg\ell \vdash_1 \bot$. If yes, then C is an implicate of φ (in fact a 1-provable implicate), i.e. (φ, C) is a positive instance of IsImplicate, and the reduction algorithm can terminate by answering yes. Note that this case includes the situation when C is a clause in φ. If no, i.e. if $\varphi \wedge \bigwedge_{\ell \in C} \neg\ell \nvdash_1 \bot$, we define an instance (φ', C') of IsEmpowering by $\varphi' = \varphi \wedge (x \vee y) \wedge (x \vee \bar{y})$ and $C' = C \vee \bar{x}$, where x and y are two new variables not appearing in (φ, C). We shall show that C is an implicate of φ if and only if C' is an empowering implicate of φ' with $\bar{x}$ as the empowered literal.

Let C be an implicate of φ. C is clearly an implicate of φ', and hence also C' is an implicate of φ'. To see that C' is empowering with $\bar{x}$ as the empowered literal recall that $\varphi \wedge \bigwedge_{\ell \in C} \neg\ell \nvdash_1 \bot$ and thus $\varphi' \wedge \bigwedge_{\ell \in C} \neg\ell \nvdash_1 \bar{x}$ because unit propagation does nothing on $(x \vee y) \wedge (x \vee \bar{y})$ and neither x nor y appear in φ and C.

Let C' be an empowering implicate of φ' with $\bar{x}$ as the empowered literal. The fact that C' is an implicate of φ' means that any assignment that falsifies C' must also falsify φ'. However, any assignment which falsifies C' sets x to 1, falsifies C and satisfies $(x \vee y) \wedge (x \vee \bar{y})$ regardless of the value of y. Thus, to falsify φ' it must falsify φ. Therefore any assignment which falsifies C must also falsify φ, which means that C is an implicate of φ. □

Let us recall the notion of a tied chain in a CNF formula used in [26] in case of unit refutation completeness.

Definition 3.7. (See [26], introduced in [32].) A *tied chain* in a CNF formula φ is a sequence of triples $(x_1, C_1, y_1), (x_2, C_2, y_2), \ldots, (x_n, C_n, y_n)$ such that:

- For $1 \leqslant i \leqslant n$, C_i is a clause in φ and x_i, y_i are two different literals in C_i (i.e. $x_i \neq y_i$).
- For $1 \leqslant i \leqslant n-1$, we have that y_i and x_{i+1} are complementary literal (called *link literals* of the chain).
- $x_1 = y_n$ is called the *tied literal* of the chain.

For example CNF formula $\varphi = (p \vee q \vee r) \wedge (\bar{r} \vee s) \wedge (\bar{s} \vee p)$ contains a tied chain with p as tied literal. In [32] it is shown that the absence of tied chains is a sufficient condition for unit refutation completeness. The following lemma was shown in [26] as Lemma 6.

Lemma 3.8. *(See Lemma 6 of [26].) Let C be an implicate of a CNF formula φ which is produced from φ by a resolution proof D in which the last resolution made is a merge resolution. Let us assume that the parent clauses of C in D are C_1 and C_2, where M denotes the set of common literals in C_1 and C_2. $M \neq \emptyset$ since the last step in D is a merge resolution and every literal of M is contained in C. Then φ contains, for each literal $\ell \in M$, a tied chain T_ℓ with ℓ as its tied literal. Furthermore, each link literal in T_ℓ has a clause in D which is produced from its parent clauses by resolution upon ℓ.*

It is argued in [26] that if there are no tied chains in a CNF formula φ, there can be no merge resolutions and thus the formula φ has to be unit refutation complete. The same argument can be used for propagation completeness. We can argue in the same way using Lemma 3.3 that absence of tied chains in a CNF formula φ implies that φ is propagation complete. We shall use this property in proofs in the text and thus we shall formulate it as a lemma to be able to reference to it.

Lemma 3.9. *If a CNF formula φ does not contain tied chains, then it is propagation complete.*

4. Resolution derivations of empowering implicates

We have seen in Theorem 3.6 that it is hard to check whether a given clause C is an empowering implicate of a given formula φ. The hard part of this test is to check whether C is in the set S of all implicates of φ. The core of the proof of Theorem 3.6 shows that considering a smaller set $S' \subseteq S$ of all empowering implicates of φ does not make this test easier.

We shall show now that the hard part of the test can be in some sense avoided by considering a suitable enlargement of the tested clause. In particular, we shall show that if a clause C is an empowering implicate of a CNF formula φ, we can always extend C by adding suitable literals to obtain clause C' which is still empowering and moreover we can check that C' is an implicate of φ simply by unit resolution. Let us recall that such a clause is called 1-provable (which is a notion introduced in [15]).

Proof of Theorem 4.2 is significantly based on Proposition 2. The proposition is given in [15] and may be restated as follows.

Proposition 4.1. *(See Proposition 2 of [15].) Let ψ be an unsatisfiable CNF such that $\psi \nvdash_1 \bot$ and Π be its resolution refutation. Then there exists a clause $C_\psi \in \Pi$ which is both empowering and 1-provable.*

Let C be an empowering implicate of a formula φ which is not 1-provable. This means that if we falsify all the literals in C and add them to φ, thus producing $\psi = \varphi \wedge \neg C$, then we get a contradiction which is not provable by unit resolution. By using the previous proposition we show that we may add the clause C to the obtained clause C_ψ to obtain a clause which is both 1-provable and empowering.

The following Theorem 4.2 is a consequence of Proposition 4.1. For readers familiar with CDCL SAT solvers, the idea remains the same as in the previous paragraph. By falsifying the literals in C any CDCL SAT solver must derive a contradiction. We add the solver's decisions to the input clause C, i.e. we add $\neg C$ and the conjunction of literals corresponding to each assigned variable to the input formula. After such an addition, we obtain the desired 1-provable and empowering clause. As we shall show later in Proposition 4.3 we can even derive such a clause C using only linear number of resolution steps with respect to the number of literal occurrences in φ.

Theorem 4.2. *Assume that C is an empowering implicate of a formula φ which is not 1-provable. Then there is an implicate C' of φ such that $C \subset C'$ and C is both 1-provable and empowering.*

Proof. Let $\psi = \varphi \wedge \neg C$ and Π be a resolution refutation of ψ. Because C is not 1-provable for φ it holds that $\psi \nvdash_1 \bot$. Then according to the proposition there is a clause $C_\psi \in \Pi$ which is both empowering and 1-provable with respect to ψ. We consider a clause $C_\varphi = C_\psi \vee C$.

First, C_φ is 1-provable for φ because C_ψ is 1-provable for ψ and all the possibly required literals were added to C_φ. It follows from the following obvious chain of equivalence:

$$C_\psi \text{ is 1-provable for } \psi \quad \Leftrightarrow \quad \psi \wedge \neg C_\psi \vdash_1 \bot \quad \Leftrightarrow \quad \varphi \wedge \neg C \wedge \neg C_\psi \vdash_1 \bot$$
$$\Leftrightarrow \quad \varphi \wedge \neg C_\varphi \vdash_1 \bot \quad \Leftrightarrow \quad C_\varphi \text{ is 1-provable for } \varphi. \tag{1}$$

Let ℓ be an empowering literal of C_ψ for ψ. Then ℓ is trivially also an empowered literal of C_φ for φ. The required properties for unit resolution and entailment come from the definitions of C_φ and ψ using similar chain of equivalences as in 1.

Thus C_φ is the desired clause since it is both empowering and 1-provable with respect to φ. □

The following proposition shows that not only can we find a 1-provable and empowering implicate as in Theorem 4.2 but we can also derive some empowering implicate by a resolution derivation of linear length with respect to the number of literals occurring in given formula. The proof is based on ideas presented in [15] and [33–35] in the context of CDCL SAT solvers. More detailed discussion about this connection is presented just after the proof of the proposition.

Proposition 4.3. *Let φ be a formula on n variables which is not propagation complete and s be the size of the CNF representation of φ (i.e. s is the total number of occurrences of literals in φ). Then there is an empowering implicate C of φ which can be derived by a series of resolutions of length at most s from φ.*

Proof. From Theorem 4.2 it follows that there is an empowering 1-provable implicate C_φ of φ with an empowered literal ℓ. Let φ' denote the formula which originates from φ after adding unit clauses formed by negated literals from C_φ, i.e. $\varphi' \equiv \varphi \wedge \neg C_\varphi$. Since C_φ is 1-provable it follows that $\varphi' \vdash_1 \bot$, i.e. we can derive contradiction from φ' by using only unit resolution.

Due to the nature of unit resolutions, we can assume that the unit refutation proceeds in two phases.

1. In the first phase we take all unit clauses from $\neg C_\varphi$ and perform unit resolutions only over the variables from C_φ.
2. In the second phase we continue with refutation proof without using unit clauses from $\neg C_\varphi$ and we do not resolve over variables from C_φ at all.

In case of unit resolutions if we already have some unit clauses it does not matter in which order we make unit resolutions over them. Thus we may assume without loss of generality that unit clauses from $\neg C_\varphi$ are used first. The first phase thus corresponds to performing partial assignment to variables of φ which falsifies literals in C_φ.

Let us now assume that $D'_1, \ldots, D'_m = \bot$ is a unit refutation proof which proceeds in the above two phases. Since it is a unit refutation proof, it can be observed that $m \leqslant s$. Let us assume that the first phase is formed by clauses $D'_1, \ldots, D'_{k'-1}$ and the second phase is formed by clauses $D'_{k'}, \ldots, D'_m$. Each clause among $D'_1, \ldots, D'_{k'-1}$ is therefore either a clause in φ' or it originates from two preceding clauses by unit resolution over variable from C_φ. Similarly each clause among $D'_{k'}, \ldots, D'_m$ is either an original clause from φ or it originates from two preceding clauses by unit resolution over a variable which is not in C_φ. Observe that if the resolution proof is irredundant, i.e. no clause can be dropped from it, no clause among

$D'_{k'}, \ldots, D'_m$ contains a variable from C_φ. This is because in the end we arrive at an empty clause and there is no way to remove a variable from C_φ in the second phase.

Let us now consider the situation in which we do not proceed with the first phase, in particular if we replace the first phase only with a list of corresponding clauses from φ. In this case the second phase can proceed as before (except that unit resolution steps are replaced by general resolution steps) only now the input clauses of phase 2 may contain some literals from C_φ as these were not removed in missing phase 1. These literals propagate down to the end of the proof and $\bot$ now becomes a subclause of C_φ. In this way we shall obtain a resolution proof of a clause $C' \subseteq C_\varphi$. Note that ℓ will be present in C' because otherwise the original unit refutation would actually prove that $\varphi \wedge \neg(C_\varphi \setminus \{\ell\}) \vdash_1 \bot$ which would be in contradiction with the fact that C_φ is an empowering implicate with empowered literal ℓ. Thus C' will be empowering by Lemma 3.1.

Let us now formalize the above idea. If D'_j is a clause among the clauses $D'_1, \ldots, D'_{k'-1}$, then it is either a unit literal from $\neg C_\varphi$ or there is a clause $D_j \in \varphi$, such that D'_j originates from D_j by falsifying some literals from C_φ. If D'_j is later used in the second phase of the unit refutation proof, the latter is the case and then D'_j originates from some clause $D_j \in \varphi$ by falsifying all literals of C_φ which appear in D_j.

If D'_j is among $D'_{k'}, \ldots, D'_m$ then we shall define clause D_j as follows. If D'_j is an original clause from φ, then $D_j = D'_j$, if $D'_j = R(D'_a, D'_b)$ where $1 \leqslant a, b < j$, then we define $D_j = R(D_a, D_b)$. Note that if D'_a and D'_b were two resolvable clauses, then the same is true about D_a and D_b. This is because if D_a contains more literals than D'_a these literals are from C_φ and the same is true for D_b and D'_b.

In the end we get a resolution derivation of length at most $m \leqslant s$ of clause $C = D_m$ which is a subclause of C_φ. As we have already mentioned ℓ must be present in C. Otherwise we would get that it was not necessary to use ℓ to derive $\bot$ from φ' which would be in contradiction with the fact that C_φ is an empowering implicate. According to Lemma 3.1 C is an empowering implicate. In particular, if C would not be empowering, then by Lemma 3.1 C_φ would not be empowering with empowered literal ℓ. □

It is also possible to prove Proposition 4.3 by analyzing a run of a CDCL SAT solver. Let us describe the sketch of such proof, the precise definitions of the below mentioned properties can be found in [33–35]. Consider the situation during the run of a CDCL SAT solver solving φ when the partial assignment satisfies $\neg C_\varphi$. Clearly then unit propagation is able to derive a conflict. By Proposition 3 of [34] each conflict clause can be derived by a trivial resolution derivation of length at most s. From Proposition 2 of [33] it follows that each asserting clause is also empowering. Since for each conflict there is at least one asserting clause, e.g. 1-UIP [35], then we have found an empowering clause with a short resolution proof from φ.

As the example in the following proposition shows, the linear upper bound on the length of resolution derivation of an empowering implicate is tight up to a multiplicative constant.

Theorem 4.4. *For each n there is a formula on $2n+1$ variables with size $O(n)$ such that it is not propagation complete, but resolution derivation of length n is needed to find an empowering implicate.*

Proof. This is actually a very simple corollary to Lemma 3.9. It is enough to construct a tied chain of length n. The following formula (for given n) contains such a chain:

$$\varphi_n = (z \vee A_1 \vee a_1) \wedge (\overline{a_1} \vee A_2 \vee a_2) \wedge \cdots \wedge (\overline{a_{n-3}} \vee A_{n-2} \vee a_{n-2}) \\ \wedge (\overline{a_{n-2}} \vee A_{n-1} \vee a_{n-1}) \wedge (\overline{a_{n-1}} \vee A_n \vee z).$$

Clearly both the number of variables and the number of clauses are linear in n. First, let us observe that $(A_1 \vee A_2 \vee \cdots \vee A_n \vee z)$ is an empowering implicate with empowered literal z. This is because by falsifying all literals $A_1, \ldots, A_n$ we get a quadratic formula which has z as a unit implicate, but unit resolution cannot derive this fact. Since the only tied chain in φ_n is composed by all the n clauses in φ_n, by Lemma 3.9 we need all these clauses in order to derive an empowering implicate. □

Note also that formula φ_n is anti-Horn (i.e. every clause contains at most one negative literal) and thus Theorem 4.4 holds also when restricted to anti-Horn or Horn formulas as we observe in the following corollary.

Corollary 4.5. *For each n there is a Horn formula on $2n+1$ variables with n clauses that it is not propagation complete but resolution derivation of length n is needed to find an empowering implicate.*

Proof. The formula φ_n in the previous theorem had at most one negative literal in each clause. Therefore, switching all literals to their complement in φ_n creates a Horn CNF formula with the desired property. □

The first idea which comes to mind when trying to find an empowering implicate of a CNF formula φ is to run resolution until one such implicate is generated. Of course, if φ already is propagation complete, it might be necessary to find all prime

implicates before we can claim φ is propagation complete. On the other hand, Proposition 4.3 suggests that in some cases, we can find an empowering implicate relatively quickly in this way. However, the obtained empowering implicate is not guaranteed to be prime. So it is natural to ask what is the necessary length of resolution derivations of prime empowering implicates. As the following observation shows, when seeking to find a prime empowering implicate all the hardness results about resolution refutations apply in this case.

Lemma 4.6. *Let φ be a CNF formula and let x be a new variable not appearing in φ. Then $\varphi \models \bot$ if and only if $\varphi \vdash_1 \bot$ or x is the only prime empowering implicate of $(\varphi \vee x)$. Moreover, if $\varphi \nvdash_1 \bot$, then there is a one-to-one correspondence between the resolution refutations of φ and resolution derivations of x from $(\varphi \vee x)$.*

Proof. Let us denote φ' the CNF formula equivalent to $\varphi \vee x$. Note that φ' can be obtained from φ by adding literal x to every clause.

Let us at first suppose that $\varphi \models \bot$. In this case $\varphi' \equiv x$ and thus x is the only prime implicate of φ'. Let us assume that x is not an empowering implicate of φ', thus $\varphi' \vdash_1 x$. Let $D'_1, \ldots, D'_k$ be a unit resolution derivation of x from φ'. Let us now denote by D_i the clause D'_i with literal x removed. It is clear that $D_1, \ldots, D_k$ is now a unit resolution refutation of φ and thus $\varphi \vdash_1 \bot$.

Now let us assume that $\varphi \vdash_1 \bot$ or x is the only prime empowering implicate of φ'. In the former case trivially $\varphi \models \bot$. In the latter case observe that φ is equivalent to φ' with x assigned to 0. Since x is an implicate of φ' it implies that $\varphi \models \bot$. Moreover, the resolution proof of x from φ' immediately gives resolution proof of $\bot$ from φ.

The one-to-one correspondence is immediately seen from the above arguments. □

Lemma 4.6 is an easy observation which shows that all results about complexity of resolution refutations of CNF formulas can be repeated for resolution derivations of prime empowering implicates as well. There are many results that can be used in this context, let us mention at least some of them. In [19] (and in many papers and books that followed) it was shown that pigeon hole principle formulas on $n(n+1)$ variables and $O(n^3)$ clauses (PHP_n) have minimal resolution refutation of size c^n for some $c > 0$. If a formula PHP_n is used with Lemma 4.6, we get immediately that every resolution derivation of the single prime empowering implicate x from $(PHP_n \vee x)$ must have superpolynomial length as well. This is in contrast with Proposition 4.3 in which we showed that for a formula which is not propagation complete there always exists some empowering implicate that can be generated using only number of resolutions that is linear in the length of the formula i.e. $O(n^4)$ for $(PHP_n \vee x)$. Of course, in case of $(PHP_n \vee x)$ such an implicate would not be prime.

Note that in [9] the formula $(PHP_n \vee x)$ was used in Section 4.2 as an example of a formula in which we can generate superpolynomially many empowering implicates while the only meaningful empowering implicate is x. In this sense Lemma 4.6 can be viewed as a simple generalization of their example, where by meaningful implicates we now consider prime implicates.

The lower bound on the length of a resolution refutation of a PHP_n formula is only superpolynomial with respect to the length of the formula. Examples of formulas on $\Theta(n)$ variables consisting of $\Theta(n)$ clauses were given in [20] for which the lower bound on the length of a minimal resolution refutation is truly exponential.

Although we used Lemma 4.6 for general resolution in our example, it is more general than that, it shows that in fact any hardness result about resolution refutations can be used for similar results about resolution derivations of a prime empowering implicate. Thus we can consider formulas which require exponential tree resolution refutations though shortest general resolution refutations have only polynomial number of steps [36]. There are other resolution refinements inbetween tree and general resolution we can take into account as well [37]. Similarly, we can use results of [38] which show formulas with resolution refutations requiring almost linear depth, it can be observed that these formulas are even Horn. There are many other results about resolution refutations which we have omitted but all of them could be used for claims about resolution derivations of prime empowering implicates as well.

5. Hardness of generating an empowering implicate

In this section we prove that testing whether a given CNF formula has an empowering implicate is an NP-complete problem. We start by showing that it is in NP.

Lemma 5.1. *The problem of testing whether a given CNF formula φ has an empowering implicate is in NP.*

Proof. It follows from Theorem 4.2 that φ has an empowering implicate if and only if it has an empowering and 1-provable implicate. Thus the certificate for φ having an empowering implicate is a clause C which is both empowering and 1-provable. These properties can be checked in polynomial time using unit propagation. It follows that the problem is in NP. □

Now we shall show that the problem is NP-hard by a reduction from 3D Matching (3DM), which is a well-known NP-complete problem [39,40]. In 3DM we are given three pairwise disjoint sets X, Y, Z of the same size $|X| = |Y| = |Z| = q$ and a set of triples $W \subseteq X \times Y \times Z$. The question we seek to answer is whether W contains a matching of size q, i.e.

whether there is a subset $M \subseteq W$ of size $|M| = q$ such that each element of X, Y, and Z is contained in exactly one triple in M (i.e. the triples in M are pairwise disjoint).

Next, we present a reduction of a 3DM problem into the problem of testing the existence of an empowering implicate. The reduction is a slight modification of the proof of coNP-hardness of recognizing whether a given CNF formula is an SLUR formula [41]. Unfortunately, the reduction from [41] cannot be used directly and we have to modify it. This is because the SLUR class coincides with the class of unit refutation complete formulas (see [42]), and the class of propagation complete formulas forms a strict subclass of unit refutation formulas as we have argued at the beginning of Section 3. In [41] we have associated a formula φ_W to an instance of 3DM for which it was true that it was SLUR (or unit refutation complete) if and only if W contained a perfect matching. In case W does not contain a perfect matching φ_W is not unit refutation complete and thus it is not propagation complete as well. Unfortunately, the opposite implication was not true for formula φ_W constructed in [41], i.e. if W contains a perfect matching, then φ_W is unit refutation complete, but it still not propagation complete. Thus we have to modify the original reduction in order to get the opposite implication as well.

Definition 5.2. With every instance X, Y, Z, W of 3DM we associate a CNF formula φ_W as follows. We assume that $X = \{x_1, \ldots, x_q\}$, $Y = \{y_1, \ldots, y_q\}$, $Z = \{z_1, \ldots, z_q\}$, and $W = \{E_1, \ldots, E_w\}$ where $w = |W|$. We also assume that $E_j = (x_{f(j)}, y_{g(j)}, z_{h(j)})$ where f, g, and h are functions determining which elements of X, Y, and Z belong to E_j (i.e. given j with $E_j = [x_{i_1}, y_{i_2}, z_{i_3}]$, function f returns the index of the x member of triple E_j, thus $f(j) = i_1$, similarly $g(j) = i_2$, and $h(j) = i_3$).

- For every $i \in \{1, \ldots, q-1\}$ let us denote $A_i = (a_i \vee \overline{a_{i+1}})$ where $a_1, \ldots, a_q$ are new variables, and let $A_q = (a_q \vee a_1)$.
- For every $i \in \{1, \ldots, q\}$ and $j \in \{1, \ldots, w\}$ let us denote $B_i^j = (\overline{b_i^1} \vee \cdots \vee \overline{b_i^{j-1}} \vee b_i^j \vee \overline{b_i^{j+1}} \vee \cdots \vee \overline{b_i^w})$, i.e. B_i^j denotes a clause on variables $b_i^1, \ldots, b_i^w$ in which every literal is negative except b_i^j.
- For every $i \in \{1, \ldots, q\}$ and $j \in \{1, \ldots, w\}$ let us denote $C_i^j = (\overline{c_i^1} \vee \cdots \vee \overline{c_i^{j-1}} \vee c_i^j \vee \overline{c_i^{j+1}} \vee \cdots \vee \overline{c_i^w})$, i.e. C_i^j denotes a clause on variables $c_i^1, \ldots, c_i^w$ in which every literal is negative except c_i^j.
- Given a triple $E_j \in W$, let $D_j = (A_{f(j)} \vee B_{g(j)}^j \vee C_{h(j)}^j)$.
- Finally, let $\varphi_W = \bigwedge_{j=1}^w D_j$.

Example 5.3. Let $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3\}$, $Z = \{z_1, z_2, z_3\}$, and $W = \{[x_1, y_1, z_1], [x_2, y_3, z_2], [x_3, y_2, z_3], [x_1, y_2, z_3], [x_3, y_1, z_1]\}$. Then there are two possible matchings $M_1 = \{[x_1, y_1, z_1], [x_2, y_3, z_2], [x_3, y_2, z_3]\}$ and $M_2 = \{[x_1, y_2, z_3], [x_2, y_3, z_2], [x_3, y_1, z_1]\}$.

The formula φ_W for this 3DM instance would be

$$\begin{aligned}
\varphi_W = {} & \big(a_1 \vee \overline{a_2} \vee b_1^1 \vee \overline{b_1^2} \vee \overline{b_1^3} \vee \overline{b_1^4} \vee \overline{b_1^5} \vee c_1^1 \vee \overline{c_1^2} \vee \overline{c_1^3} \vee \overline{c_1^4} \vee \overline{c_1^5}\big) \wedge \\
& \big(a_2 \vee \overline{a_3} \vee \overline{b_3^1} \vee b_3^2 \vee \overline{b_3^3} \vee \overline{b_3^4} \vee \overline{b_3^5} \vee \overline{c_2^1} \vee c_2^2 \vee \overline{c_2^3} \vee \overline{c_2^4} \vee \overline{c_2^5}\big) \wedge \\
& \big(a_3 \vee a_1 \vee \overline{b_2^1} \vee \overline{b_2^2} \vee b_2^3 \vee \overline{b_2^4} \vee \overline{b_2^5} \vee \overline{c_3^1} \vee \overline{c_3^2} \vee c_3^3 \vee \overline{c_3^4} \vee \overline{c_3^5}\big) \wedge \\
& \big(a_1 \vee \overline{a_2} \vee \overline{b_2^1} \vee \overline{b_2^2} \vee \overline{b_2^3} \vee b_2^4 \vee \overline{b_2^5} \vee \overline{c_3^1} \vee \overline{c_3^2} \vee \overline{c_3^3} \vee c_3^4 \vee \overline{c_3^5}\big) \wedge \\
& \big(a_3 \vee a_1 \vee \overline{b_1^1} \vee \overline{b_1^2} \vee \overline{b_1^3} \vee \overline{b_1^4} \vee b_1^5 \vee \overline{c_1^1} \vee \overline{c_1^2} \vee \overline{c_1^3} \vee \overline{c_1^4} \vee c_1^5\big).
\end{aligned}$$

In the following, we shall show that the empowering implicates of φ_W correspond to perfect matchings of W. In particular, given φ_W in this example we shall have two possible empowering implicates corresponding to matchings M_1 and M_2 respectively.

$$\begin{aligned}
H_1 = {} & \big(a_1 \vee b_1^1 \vee \overline{b_1^2} \vee \overline{b_1^3} \vee \overline{b_1^4} \vee \overline{b_1^5} \vee c_1^1 \vee \overline{c_1^2} \vee \overline{c_1^3} \vee \overline{c_1^4} \vee \overline{c_1^5} \vee \\
& \overline{b_3^1} \vee b_3^2 \vee \overline{b_3^3} \vee \overline{b_3^4} \vee \overline{b_3^5} \vee \overline{c_2^1} \vee c_2^2 \vee \overline{c_2^3} \vee \overline{c_2^4} \vee \overline{c_2^5} \vee \\
& \overline{b_2^1} \vee \overline{b_2^2} \vee b_2^3 \vee \overline{b_2^4} \vee \overline{b_2^5} \vee \overline{c_3^1} \vee \overline{c_3^2} \vee c_3^3 \vee \overline{c_3^4} \vee \overline{c_3^5}\big)
\end{aligned}$$

and

$$\begin{aligned}
H_2 = {} & \big(a_1 \vee \overline{b_2^1} \vee \overline{b_2^2} \vee \overline{b_2^3} \vee b_2^4 \vee \overline{b_2^5} \vee \overline{c_3^1} \vee \overline{c_3^2} \vee \overline{c_3^3} \vee c_3^4 \vee \overline{c_3^5} \vee \\
& \overline{b_3^1} \vee b_3^2 \vee \overline{b_3^3} \vee \overline{b_3^4} \vee \overline{b_3^5} \vee \overline{c_2^1} \vee c_2^2 \vee \overline{c_2^3} \vee \overline{c_2^4} \vee \overline{c_2^5} \vee \\
& \overline{b_1^1} \vee \overline{b_1^2} \vee \overline{b_1^3} \vee \overline{b_1^4} \vee b_1^5 \vee \overline{c_1^1} \vee \overline{c_1^2} \vee \overline{c_1^3} \vee \overline{c_1^4} \vee c_1^5\big).
\end{aligned}$$

We claim that φ_W admits an empowering implicate if and only if the input 3DM instance has a perfect matching. The proof of this claim is split into the following two lemmas.

Lemma 5.4. *Using the notation from Definition 5.2 let $M \subseteq \{1, \ldots, w\}$ be a perfect matching and let $G = \bigvee_{j \in M} B^j_{g(j)} \vee C^j_{h(j)} \vee a_1$ be a clause. Then G is an empowering implicate of φ_W with a_1 being the empowered literal.*

Proof. Let us renumber the triples in W so that $M = \{1, \ldots, q\}$ and that $f(i) = i$, i.e. $x_i \in E_i$, for $i = 1, \ldots, q$. Now, let us consider the following chain of resolutions:

$$G_1 = D_1, G_2 = R(G_1, D_2), \ldots, G_i = R(G_{i-1}, D_i), \ldots, G_q = R(G_{q-1}, D_q).$$

One can check that $G_q = G$ and thus G is an implicate of φ_W. All clauses in this chain are resolvable because the triples in the matching are disjoint.

It remains to prove that G is empowering. Let φ' denote the formula originating from φ_W by falsifying (substituting the value *false* for) all literals in G except a_1. We are going to show that $\varphi' \equiv A_1 \wedge \cdots \wedge A_q$. To this end, let $D_j = (A_{f(j)} \vee B^j_{g(j)} \vee C^j_{h(j)})$ be an arbitrary clause in φ_W and let us have a look on what happens with D_j after falsifying the aforementioned literals. If $j \in M$, then we have falsified all literals in D_j except $A_{f(j)}$. If $j \notin M$, then let $j' \in M$ be an index for which $g(j') = g(j)$. Such an index exists because M as a matching covers element $y_{g(j)}$. It follows that $B^{j'}_{g(j')}$ has two conflict variables with $B^j_{g(j)}$ (note that $j \neq j'$) and thus by falsifying all literals in $B^{j'}_{g(j')}$ we necessarily satisfy the clause D_j. By these considerations only clauses $A_1, \ldots, A_q$ remain in φ'. Note that none of these clauses is missing since M is a matching. The CNF formula $\varphi' = A_1 \wedge \cdots \wedge A_q$ has no unit clauses and thus unit propagation will not derive anything from it. It follows that G is an empowering implicate. □

Example 5.5. Let us consider implicate H_1 from Example 5.3 and let us show how it can be derived by resolutions from formula φ_W. Implicate H_1 corresponds to matching $M_1 = \{[x_1, y_1, z_1], [x_2, y_3, z_2], [x_3, y_2, z_3]\}$. The clauses of φ_W corresponding to triples in M_1 are:

$$D_1 = \big(a_1 \vee \overline{a_2} \vee b^1_1 \vee \overline{b^2_1} \vee \overline{b^3_1} \vee \overline{b^4_1} \vee \overline{b^5_1} \vee c^1_1 \vee \overline{c^2_1} \vee \overline{c^3_1} \vee \overline{c^4_1} \vee \overline{c^5_1}\big),$$
$$D_2 = \big(a_2 \vee \overline{a_3} \vee \overline{b^1_3} \vee b^2_3 \vee \overline{b^3_3} \vee \overline{b^4_3} \vee \overline{b^5_3} \vee \overline{c^1_2} \vee c^2_2 \vee \overline{c^3_2} \vee \overline{c^4_2} \vee \overline{c^5_2}\big),$$
$$D_3 = \big(a_3 \vee a_1 \vee \overline{b^1_2} \vee \overline{b^2_2} \vee b^3_2 \vee \overline{b^4_2} \vee \overline{b^5_2} \vee \overline{c^1_3} \vee \overline{c^2_3} \vee c^3_3 \vee \overline{c^4_3} \vee \overline{c^5_3}\big).$$

By resolving D_1 with D_2 we get clause

$$G_1 = \big(a_1 \vee \overline{a_3} \vee b^1_1 \vee \overline{b^2_1} \vee \overline{b^3_1} \vee \overline{b^4_1} \vee \overline{b^5_1} \vee c^1_1 \vee \overline{c^2_1} \vee \overline{c^3_1} \vee \overline{c^4_1} \vee \overline{c^5_1} \vee$$
$$\overline{b^1_3} \vee b^2_3 \vee \overline{b^3_3} \vee \overline{b^4_3} \vee \overline{b^5_3} \vee \overline{c^1_2} \vee c^2_2 \vee \overline{c^3_2} \vee \overline{c^4_2} \vee \overline{c^5_2}\big).$$

By further resolving G_1 with D_3 we get desired implicate H_1. Observe that G_1 was produced by non-merge resolution and thus by Lemma 3.3 it is not an empowering implicate. On the other hand $H_1 = R(G_1, D3)$ is produced using a merge resolution and it is hence a good candidate for an empowering implicate of φ_W. If we falsify all literals in H_1 except a_1, we get clauses $D'_1 = (a_1 \vee \overline{a_2})$, $D'_2 = (a_2 \vee \overline{a_3})$, and $D'_3 = (a_3 \vee a_1)$ from clauses D_1, D_2, and D_3 respectively. The remaining two clauses in φ_W are satisfied by this assignment. Thus H_1 is indeed an empowering implicate of φ_W.

Lemma 5.6. *Let $G = G' \vee u$ be a prime empowering implicate of φ_W with u being the empowered literal. Then u must be a_1 and G' determines perfect matching in the same way as G in Lemma 5.4.*

Proof. The proof heavily relies on Lemma 3.8 and Lemma 3.9. Let us consider a resolution derivation of $G' \vee u$. By Lemma 3.8 and Lemma 3.9 this derivation contains a tied chain in which every link literal is resolved upon. Consider how such a tied chain in φ_W may look like. The only possible tied literal is a_1, and the link literals are among $a_2, \ldots, a_q$. This is because, as we can observe, the c and b variables cannot be resolved upon (if two clauses have a conflict in a c or b variable, they are not resolvable as they have at least two conflicts). Thus a tied chain in φ_W has to look as follows: $D_{i_1}, \ldots, D_{i_q}$, where $f(i_j) = j$, i.e. D_{i_1} contains $(a_1 \vee \overline{a_2})$, D_{i_q} contains $(a_q \vee a_1)$ or vice versa if we look at the tied chain in the opposite direction. We shall assume without loss generality that the former is the case (i.e. D_{i_1} contains $(a_1 \vee \overline{a_2})$ and D_{i_q} contains $(a_q \vee a_1)$). For $1 < j < q$ clause D_{i_j} contains $(a_j \vee \overline{a_{j+1}})$. The chain has to have the length q as this is necessary to get from a_1 back to a_1 through link literals among $a_2, \ldots, a_q$. Now, if we look at how clauses $D_{i_1}, \ldots, D_{i_q}$ can be resolved upon, we get that triples $E_{i_1}, \ldots, E_{i_q}$ must be disjoint to have each consecutive pair in the sequence $D_{i_1}, \ldots, D_{i_q}$ resolvable. This is because if D_{i_j}, $D_{i_{j+1}}$ are resolvable, then they cannot have a conflict in any b or c variable, which implies that E_{i_j} and $E_{i_{j+1}}$ are disjoint. We can also observe that the b and c variables cannot be cancelled out by resolution and they are therefore all present in the resolvent. Thus, the only possibility is that G' itself contains all the B and C parts from $D_{i_1}, \ldots, D_{i_q}$ and thus G' itself determines the matching. □

Theorem 5.7. *The problem of testing whether a given CNF formula has an empowering implicate is an NP-complete problem.*

Proof. The NP membership is proved in Lemma 5.1. The NP-hardness follows from the reduction defined by Definition 5.2 using Lemma 5.4 and Lemma 5.6. Note that by Lemma 3.2 it is enough to consider prime empowering implicates and thus the assumption that G is a prime implicate in Lemma 5.6 is not restrictive. □

Using the reduction from Definition 5.2, we can also show that when trying to make a CNF formula propagation complete, we can in general observe an exponential blow up of the number of clauses. In particular we can show that there is a uniform and size increasing family of CNF formulas in which an exponential number of implicates has to be added to a member of this family in order to make it propagation complete. Here by "uniform" we mean that we have a uniform construction which constructs a formula on n variables from this family based only on a parameter n.

Theorem 5.8. *There is a uniform and size increasing family of CNF formulas parameterized with the number of variables n and where the number of clauses is $O(n)$, such that the number of implicates that needs to be added to a CNF formula φ on n variables from this family to make it propagation complete is exponential in n (and thus in the size of the formula, too).*

Proof. Let us consider an instance of 3DM where $|X| = |Y| = |Z| = q$ and $W = X \times Y \times Z$. We claim that at least $(q!)^2$ implicates have to be added to φ_W to make it propagation complete. First, let us observe that we can find $(q!)^2$ pairwise different perfect matchings in W. This is because each perfect matching in W can be viewed as a pair of perfect matchings in a complete bipartite graph with partities X and Y and a perfect matching in a complete bipartite graph with partities Y and Z. We have $q!$ perfect matchings in each of these bipartite graphs and thus we have $(q!)^2$ possible pairs of them. Note, on the other hand, that $w = |W| = q^3$.

Let $M = \{m_{r_1}, m_{r_2}, \ldots, m_{r_q}\}$, $M \subseteq W$ be a perfect matching. Let us denote the clause

$$H_M = a_1 \vee \bigvee_{i=1}^{q} \left(B^{r_i}_{g(r_i)} \vee C^{r_i}_{h(r_i)}\right).$$

From Lemma 5.4 it follows that H_M is an empowering implicate. We claim that if we add H_M to φ_W, then $H_{M'}$ remains empowering provided M' is a different matching than M. Since M and M' are different and $|M| = |M'|$, we have that $M \setminus M' \neq \emptyset$ and $M' \setminus M \neq \emptyset$. Let r_ℓ be an index of a triple such that $m_{r_\ell} \in M \setminus M'$. Thus $B^{r_\ell}_{g(r_\ell)}$ forms a subclause of H_M. Since m_{r_ℓ} does not belong to M', we have that $g(r_\ell)$ is covered by a different triple r'_ℓ in M'. Thus $B^{r'_\ell}_{g(r'_\ell)}$, where $g(r'_\ell) = g(r_\ell)$, forms a subclause of $H_{M'}$. If we falsify all literals in $H_{M'}$ except a_1, we get that H_M is satisfied, because $B^{r_\ell}_{g(r_\ell)}$ and $B^{r'_\ell}_{g(r'_\ell)}$ have a conflict variable. Hence H_M plays no role in unit propagation used to possibly derive a_1. Whether a_1 can or cannot be derived by unit resolution from $\varphi_W \wedge H_M$ after falsifying the literals in $H_{M'}$ except a_1 is thus equivalent to whether a_1 can be derived by unit resolution from φ_W after falsifying the literals in $H_{M'}$ except a_1. Note that the above observation can be generalized to the case when we add more than one of these matching clauses to φ_W and thus it is necessary to add all the clauses corresponding to perfect matchings to φ_W to make it propagation complete.

It follows that the number of implicates needed to be added to φ_W to make it propagation complete is at least $(q!)^2$ which is exponential in the size of formula φ_W consisting of q^3 clauses build on $q + qw + qw = \Theta(q^4)$ variables. Family of the formulas defined in this, with $n = \Theta(q^4)$, satisfies the proposition of the theorem. □

As we have already discussed in Section 2.5, the result contained in Theorem 5.8 is in tight connection to the results in [21] where the authors show in Corollary 4 that there is no polynomial sized CNF decomposition of any AllDifferent domain consistency propagator. As we have mentioned in Section 2.5, the quasi-polynomial shown in [12] is based on a quasi-polynomial lower bound of Razborov [22] on the size of a monotone circuit computing whether there is a perfect matching in a given graph. Quite interestingly, to the best of our knowledge, there is no stronger lower bound on the size of monotone circuit computing whether a bipartite graph contains perfect matching than the quasi-polynomial shown in [22]. It is thus an interesting question whether a stronger lower bound could not be shown using the connection between domain consistency propagators and monotone circuits shown in [21]. Note that in the proof of Theorem 5.8 we could use a bipartite matching instead of 3DM for constructing the family of CNF formulas with required properties (we used 3DM mainly to take advantage of Lemma 5.4). Thus, it might be possible to construct a domain consistency propagator for the AllDifferent constraint which contains such a modified family of CNF formulas. On the other hand, it would still be only one example of a domain consistency propagator which is not enough to argue about any domain consistency propagator for the AllDifferent constraint.

6. Conclusions

We derived several properties of propagation complete formulas and empowering implicates. Let us now recollect the answers to the four questions we posed in the introduction.

1. We showed that given a clause C, the problem of deciding whether C is an empowering implicate of a CNF formula φ is co-NP complete. This result comes as no big surprise as the hard part of this decision is answering the question whether C is even an implicate of φ and the new information brought here is that restricting the attention to the subset of empowering implicates does not make this problem easier.
2. On the other hand, if there is an empowering implicate of φ, there is always a 1-provable empowering implicate of φ as stated in Theorem 4.2 which follows from Proposition 2 of [15]. It means that in this case there is always a clause for which it is easy to show that it is an implicate of φ simply by unit propagation. Extending this reasoning further, we can even find an empowering implicate C, such that it can be derived by a series of resolutions of linear length with respect to the number of literals occurring in a given formula (Proposition 4.3 which relates to Proposition 4 of [15] and other results in [33–35]). We also showed that this bound is tight, in some cases linear length of resolution proof is necessary. It is important that in this case we consider a general implicate C since we also showed that if we want to derive a prime empowering implicate C, an exponential length of resolution proof might be necessary.
3. We showed that the problem in which we ask whether there is an empowering implicate for given CNF formula φ is NP-complete, which means that checking whether given CNF formula φ is propagation complete is co-NP complete. Note that this strenghtens the results of [9] where the authors showed that the decision problem whether φ has an empowering implicate belongs to Σ_2^p. At a first sight we might see a slight contrast between the fact that checking whether given clause C is an empowering implicate of given CNF formula φ is co-NP complete while checking whether there exists an empowering implicate for a given CNF formula φ is NP-complete. The difference is that in the former problem we are given a particular clause C and it is already co-NP complete to decide whether C is an implicate of φ checking the empowering property is then easy. In the latter case we are given only a CNF formula φ and thus we may look for special empowering implicates, in particular the 1-provable ones. Checking whether a given clause C is 1-provable empowering implicate of φ is then polynomial and the hard part is to find one.
4. Finally, we showed that there is a uniform and size increasing family of CNF formulas such that given CNF formula φ from this family we have to add an exponential number of empowering implicates to make it propagation complete. This result strenghtens a quasi-polynomial bound which follows from the results of [21] and the connection between CNF decompositions of domain consistency propagators in CSP.
   It remains an interesting question whether a similar result can be obtained for different representations of propositional formulas, such as ZBDD (zero-suppressed binary decision diagrams [43,44], see e.g. [45] for use in propositional formula representation). In particular it may be interesting to investigate, whether the family of formulas defined in Theorem 5.8 requires an exponentially sized ZBDD to represent its propagation complete counterpart. We leave this question for further research. We can consider other representations as well, see e.g. [17] for a comprehensive list of various representations used in knowledge compilation.

Acknowledgements

The authors thank four anonymous referees for their valuable comments that helped to improve the paper. The first, the second, the fourth, and the sixth author gratefully acknowledge the support of the Charles University Grant Agency (grant Nos. 266111, 265511, and 600112). The third and fifth author thankfully acknowledge a support by the Czech Science Foundation (grant P202/10/1188).

References

[1] J. Franco, A. Van Gelder, A perspective on certain polynomial-time solvable classes of satisfiability, Discrete Appl. Math. 125 (2–3) (2003) 177–214.
[2] J.S. Schlipf, F.S. Annexstein, J.V. Franco, R.P. Swaminathan, On finding solutions for extended horn formulas, Inform. Process. Lett. 54 (3) (1995) 133–137.
[3] M. Davis, G. Logemann, D. Loveland, A machine program for theorem-proving, Commun. ACM 5 (7) (1962) 394–397.
[4] A. Darwiche, K. Pipatsrisawat, Complete algorithms, in: A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, in: Frontiers in Artificial Intelligence and Applications, vol. 185, IOS Press, 2009, pp. 99–130.
[5] J.P. Marques-Silva, K.A. Sakallah, Grasp: A search algorithm for propositional satisfiability, IEEE Trans. Comput. 48 (5) (1999) 506–521.
[6] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, S. Malik, Chaff: Engineering an efficient sat solver, in: Proceedings of the 38th Annual Design Automation Conference, ACM, 2001, pp. 530–535.
[7] J.P. Marques-Silva, A.K. Sakallah, GRASP — a new search algorithm for satisfiability, in: Proceedings of the 1996 IEEE/ACM International Conference on Computer-Aided Design, ICCAD '96, IEEE Computer Society, Washington, DC, USA, 1996, pp. 220–227.
[8] J.P.M. Silva, I. Lynce, S. Malik, Conflict-driven clause learning sat solvers, in: A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, in: Frontiers in Artificial Intelligence and Applications, vol. 185, IOS Press, 2009, pp. 131–153.
[9] L. Bordeaux, J. Marques-Silva, Knowledge compilation with empowerment, in: M. Bieliková, G. Friedrich, G. Gottlob, S. Katzenbeisser, G. Turán (Eds.), SOFSEM 2012: Theory and Practice of Computer Science, in: Lecture Notes in Computer Science, vol. 7147, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 612–624.
[10] F. Bacchus, GAC via unit propagation, in: C. Bessière (Ed.), Principles and Practice of Constraint Programming – CP 2007, in: Lecture Notes in Computer Science, vol. 4741, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 133–147.
[11] J. Huang, Universal booleanization of constraint models, in: P.J. Stuckey (Ed.), Principles and Practice of Constraint Programming, in: Lecture Notes in Computer Science, vol. 5202, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 144–158.
[12] C. Bessiere, G. Katsirelos, N. Narodytska, C.-G. Quimper, T. Walsh, Decompositions of all different, global cardinality and related constraints, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009, pp. 419–424.
[13] S. Brand, N. Narodytska, C.-G. Quimper, P. Stuckey, T. Walsh, Encodings of the sequence constraint, in: Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming, CP'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 210–224.

[14] C.-G. Quimper, T. Walsh, Decompositions of grammar constraints, in: Proceedings of the 23rd National Conference on Artificial Intelligence, vol. 3, AAAI'08, AAAI Press, 2008, pp. 1567–1570.
[15] K. Pipatsrisawat, A. Darwiche, On the power of clause-learning sat solvers as resolution engines, Artificial Intelligence 175 (2) (2011) 512–525.
[16] P. Marquis, Knowledge compilation using theory prime implicates, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol. 1, IJCAI'95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, pp. 837–843.
[17] A. Darwiche, P. Marquis, A knowledge compilation map, J. Artificial Intelligence Res. 17 (1) (2002) 229–264.
[18] P. Marquis, S. Sadaoui, A new algorithm for computing theory prime implicates compilations, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence, vol. 1, AAAI'96, AAAI Press, 1996, pp. 504–509.
[19] A. Haken, The intractability of resolution, Theoret. Comput. Sci. 39 (0) (1985) 297–308.
[20] A. Urquhart, Hard examples for resolution, J. ACM 34 (1) (1987) 209–219.
[21] C. Bessiere, G. Katsirelos, N. Narodytska, T. Walsh, Circuit complexity and decompositions of global constraints, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009, pp. 412–418.
[22] A.A. Razborov, Lower bounds on monotone complexity of some boolean functions, Dokl. Akad. Nauk SSSR 281 (4) (1985) 789–801.
[23] M.R. Genesereth, N.J. Nilsson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.
[24] H.K. Büning, T. Lettmann, Propositional Logic: Deduction and Algorithms, Cambridge University Press, New York, NY, USA, 1999.
[25] M. Dalal, D.W. Etherington, A hierarchy of tractable satisfiability problems, Inform. Process. Lett. 44 (4) (1992) 173–180.
[26] A. del Val, Tractable databases: How to make propositional unit resolution complete through compilation, in: J. Doyle, E. Sandewall, P. Torassi (Eds.), Proceedings of Fourth International Conference on Principles of Knowledge Representation and Reasoning, KR'94, Morgan Kaufmann, 1994, pp. 551–561.
[27] A. Atserias, J.K. Fichte, M. Thurley, Clause-learning algorithms with many restarts and bounded-width resolution, in: Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT'09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 114–127.
[28] F. Rossi, P.v. Beek, T. Walsh, Handbook of Constraint Programming, Foundations of Artificial Intelligence, Elsevier Science Inc., New York, NY, USA, 2006.
[29] C. Bessière, E. Hebrard, T. Walsh, Local consistencies in SAT, in: E. Giunchiglia, A. Tacchella (Eds.), Theory and Applications of Satisfiability Testing, in: Lecture Notes in Computer Science, vol. 2919, Springer, Berlin, Heidelberg, 2004, pp. 299–314, http://dx.doi.org/10.1007/978-3-540-24605-3_23.
[30] C. Schulte, P. Stuckey, Speeding up constraint propagation, in: M. Wallace (Ed.), Principles and Practice of Constraint Programming – CP 2004, in: Lecture Notes in Computer Science, vol. 3258, Springer, Berlin, Heidelberg, 2004, pp. 619–633, http://dx.doi.org/10.1007/978-3-540-30201-8_45.
[31] J.-C. Régin, A filtering algorithm for constraints of difference in CSPs, in: Proceedings of the Twelfth National Conference on Artificial Intelligence, vol. 1, AAAI '94, American Association for Artificial Intelligence, Menlo Park, CA, USA, 1994, pp. 362–367.
[32] K. Eshghi, A tractable class of abduction problems, in: Proceedings of the 13th International Joint Conference on Artificial intelligence, vol. 1, IJCAI'93, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993, pp. 3–8.
[33] K. Pipatsrisawat, A. Darwiche, A new clause learning scheme for efficient unsatisfiability proofs, in: Proceedings of the 23rd National Conference on Artificial Intelligence, vol. 3, AAAI'08, AAAI Press, 2008, pp. 1481–1484.
[34] P. Beanie, H. Kautz, A. Sabharwal, Understanding the power of clause learning, in: Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI'03, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003, pp. 1194–1201.
[35] L. Zhang, C.F. Madigan, M.H. Moskewicz, S. Malik, Efficient conflict driven learning in a boolean satisfiability solver, in: Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design, ICCAD '01, IEEE Press, Piscataway, NJ, USA, 2001, pp. 279–285.
[36] E. Ben-Sasson, R. Impagliazzo, A. Wigderson, Near optimal separation of tree-like and general resolution, Combinatorica 24 (4) (2004) 585–603.
[37] J. Buresh-Oppenheim, T. Pitassi, The complexity of resolution refinements, in: Proceedings of the 18th Annual IEEE Symposium, Logic in Computer Science, 2003, 2003, pp. 138–147, http://dx.doi.org/10.1109/LICS.2003.1210053.
[38] A. Urquhart, The depth of resolution proofs, Stud. Log. 99 (1–3) (2011) 349–364.
[39] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, 1972, pp. 85–103.
[40] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, San Francisco, 1979.
[41] O. Čepek, P. Kučera, V. Vlček, Properties of SLUR formulae, in: M. Bieliková, G. Friedrich, G. Gottlob, S. Katzenbeisser, G. Turán (Eds.), SOFSEM 2012: Theory and Practice of Computer Science, in: Lecture Notes in Computer Science, vol. 7147, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 177–189.
[42] M. Gwynne, O. Kullmann, Generalising unit-refutation completeness and SLUR via nested input resolution, http://arxiv.org/abs/1204.6529.
[43] S. Minato, Zero-suppressed BDDs for set manipulation in combinatorial problems, in: 30th Conference on Design Automation, 1993, 1993, pp. 272–277.
[44] S.-i. Minato, Zero-suppressed BDDs and their applications, Int. J. Softw. Tools Technol. Transf. (STTT) 3 (2) (2001) 156–170.
[45] P. Chatalic, L.S. Zres, The old Davis–Putnam procedure meets ZBDD, in: D. McAllester (Ed.), Automated Deduction – CADE-17, in: Lecture Notes in Computer Science, vol. 1831, Springer, Berlin, Heidelberg, 2000, pp. 449–454, http://dx.doi.org/10.1007/10721959_35.