Complexity of SAT Problems, Clone Theory and the Exponential Time Hypothesis

Peter Jonsonn, Victor Lagerkvist, Gustav Nordh, Bruno Zanuttini

Complexity of SAT Problems, Clone Theory and the Exponential Time Hypothesis

Peter Jonsson[1], Victor Lagerkvist[2], Gustav Nordh[1], and Bruno Zanuttini[3]

[1] Department of Computer and Information Science, Linköpings Universitet, Sweden
{peter.jonsson, gustav.nordh}@liu.se
[2] Department of Computer and Information Science, Linköpings Universitet, Sweden, (Corresponding author)
vicla605@student.liu.se
[3] GREYC, Université de Caen Basse-Normandie, France
bruno.zanuttini@unicaen.fr

Abstract. The construction of exact exponential-time algorithms for NP-complete problems has for some time been a very active research area. Unfortunately, there is a lack of general methods for studying and comparing the time complexity of algorithms for such problems. We propose such a method based on *clone theory* and demonstrate it on the SAT problem. Schaefer has completely classified the complexity of SAT with respect to the set of allowed relations and proved that this parameterized problem exhibits a dichotomy: it is either in P or NP-complete. We show that there is a certain partial order on the NP-complete SAT problems with a close connection to their worst-case time complexities; if a problem SAT(S) is below a problem SAT(S') in this partial order, then SAT(S') cannot be solved strictly faster than SAT(S). By using this order, we identify a relation R such that SAT(R) is the *computationally easiest* NP-complete SAT problem. This result may be interesting when investigating the borderline between P and NP since one appealing way of studying this borderline is to identify problems that, in some sense, are situated close to it (such as a 'very hard' problem in P or a 'very easy' NP-complete problem). We strengthen the result by showing that SAT(R)-2 (i.e. SAT(R) restricted to instances where no variable appears in more than two clauses) is NP-complete, too. This is in contrast to, for example, 1-in-3-SAT which is in P under the same restriction. We then relate SAT(R)-2 to the exponential-time hypothesis (ETH) and show that ETH holds if and only if SAT(R)-2 is not sub-exponential. This constitutes a strong connection between ETH and the SAT problem under both severe relational and severe structural restrictions, and it may thus serve as a tool for studying the borderline between sub-exponential and exponential problems. In the process, we also prove a stronger version of Impagliazzo et. al's sparsification lemma for k-SAT; namely that all finite, NP-complete Boolean languages can be sparsified into each other. This should be compared with Santhanam and Srinivasan's recent negative result which states that the same does not hold for all infinite Boolean languages.

1 Introduction

This paper is concerned with the SAT(S) class of problems: given a finite set of Boolean relations S, decide whether a conjunction of constraints (where only relations from S are used) is satisfiable or not. This class of problems is very rich and contains many problems that are highly relevant both theoretically and in practice. Since Schaefer's seminal dichotomy result [27], the computational complexity of SAT(S) is completely known: we know for which S that SAT(S) is polynomial-time solvable and for which it is NP-complete, and these are the only possible cases. On the other hand, judging from the running times of the many algorithms that have been proposed for different NP-complete SAT(S) problems, it seems that the computational complexity varies greatly for different S. As an example, 3-SAT (S consists of all clauses of length at most 3) is only known to be solvable in time $O(1.3334^n)$ [22] (where n is the number of variables), and so it seems to be a much harder problem than, for instance, positive 1-in-3-SAT (S consists only of the relation $\{(0,0,1),(0,1,0),(1,0,0)\}$), which can be solved in time $O(1.0984^n)$ [32]. It is fair to say that we have a very vague understanding of the time complexity of NP-complete problems, and this fact is clearly expressed in [6].

> What the field of exponential-time algorithms sorely lacks is a complexity-theoretic framework for showing running time lower bounds.

In this paper, we initiate a systematic study of the relationships between the worst-case complexity of different SAT(S) problems, where we measure the complexity as a function of the number of variables. Ultimately, one would like to have a 'table' that for each NP-complete SAT(S) problem contains a number c such that SAT(S) can be solved in $\Theta(c^n)$ time. It seems that we are very far from this goal, unfortunately. Let us imagine a weaker qualitative approach: construct a table that for every two problems SAT(S) and SAT(S') tells us whether SAT(S) and SAT(S') can be solved equally fast, whether SAT(S) can be solved strictly faster than SAT(S'), or vice versa. That is, we have access to the underlying total order on running times but we cannot say anything about the exact figures. Not surprisingly, we are far from this goal, too. However, this table can, in a sense, be approximated: there are non-trivial lattices that satisfy this property whenever S and S' are comparable to each other in the lattice. To obtain such lattices, we exploit *clone theory* [18,30]. This theory has proven to be very powerful when studying the complexity of SAT(S) and its multi-valued generalization known as *constraint satisfaction problems (CSP)* [5]. However, it is not clear how this theory can be used for studying the worst-case running times for algorithms. We show how to use it for this purpose in Section 3, and our basic observation is that the lattice of *partial clones* [2,3] has the required properties. We would like to emphasize that this approach can be generalized in different ways; it is not restricted to Boolean problems and it is applicable to other computational problems such as counting and enumeration.

As a concrete application of this method, we identify the computationally easiest NP-complete SAT(S) problem in Section 4; by 'computationally easiest', we mean that if any NP-complete SAT(S) problem can be solved in $O(c^n)$ time, then the easiest problem can be solved in $O(c^n)$ time, too. This easiest NP-complete SAT(S) problem is surprisingly simple: S consists of a single 6-ary relation $R_{1/3}^{\neq\neq\neq}$ which contains the three tuples $(1,0,0,0,1,1)$, $(0,1,0,1,0,1)$, and $(0,0,1,1,1,0)$. This result is obtained by making use of Schnoor and Schnoor's [28] machinery for constructing *weak bases*. We note that there has been an interest in identifying extremely easy NP-complete problems before. For instance, van Rooij et. al have shown that the PARTITION INTO TRIANGLES problem restricted to graphs of maximum degree four can be solved in $O(1.02445^n)$ time [31]. They argue that practical algorithms may arise from this kind of studies, and the very same observation has been made by, for instance, Woeginger [33]. It is important to note that our results give much more information than just the mere fact that SAT($R_{1/3}^{\neq\neq\neq}$) is easy to solve; they also tell us how this problem is related to all other problems within the large and diverse class of finite SAT problems. This is one of the major advantages in using the clone-theoretical approach when studying these kind of questions. Another reason to study such problems is that they, in some sense, are 'close' to the borderline between problems in NP that are not complete and NP-complete problems (here we tacitly assume that P $\neq$ NP). The structure of this borderline has been studied with many different aims and many different methods; two well-known examples are the articles by Ladner [17] and Schöning [29].

We continue by studying the complexity of SAT($R_{1/3}^{\neq\neq\neq}$) and general SAT problems in greater detail by relating them to the EXPONENTIAL TIME HYPOTHESIS (ETH) [13], i.e. the hypothesis that k-SAT cannot be solved in sub-exponential time for $k \geq 3$. The ETH has recently gained popularity when studying the computational complexity of combinatorial problems, cf. the survey by Lokshtanov et al. [19].

We first note (in Section 5) that SAT($R_{1/3}^{\neq\neq\neq}$) restricted to instances where no variable appears more than twice (the SAT($R_{1/3}^{\neq\neq\neq}$)-2 problem) is still NP-complete (in contrast to, for instance, positive 1-in-3-SAT which is in P under the same restriction). We prove this by using results by Dalmau and Ford [8] combined with the fact that $R_{1/3}^{\neq\neq\neq}$ is not a Δ-matroid relation. We then show (in Section 6) that the exponential-time hypothesis holds if and only if SAT($R_{1/3}^{\neq\neq\neq}$)-2 cannot be solved in sub-exponential time, i.e. SAT($R_{1/3}^{\neq\neq\neq}$)-2 is ETH-hard. By using this result, we show the following consequence: if ETH does not hold, then SAT(S)-k is sub-exponential for every k whenever S is finite. Impagliazzo et al. [13] have proved that many NP-complete problems in SNP (which contains the SAT problems) are ETH-hard. Thus, we strengthen this result when restricted to SAT problems. In the process, we also prove a stronger version of Impagliazzo et. al's [13] sparsification lemma for k-SAT; namely that all finite, NP-complete Boolean languages can be sparsified into each other. This can be contrasted with Santhanam's and Srinivasan's [26] recent negative result which states that the same does not hold for the unrestricted SAT problem and, consequently, not for all infinite Boolean languages.

2 The Boolean SAT problem

We begin by introducing the notation and basic results that will be used in the rest of this paper. The set of all n-tuples over $\{0,1\}$ is denoted by $\{0,1\}^n$. Any subset of $\{0,1\}^n$ is called an n-ary relation on $\{0,1\}$. The set of all finitary relations over $\{0,1\}$ is denoted by BR. A *constraint language* over $\{0,1\}$ is a finite set $S \subseteq BR$.

Definition 1. *The Boolean satisfiability problem over the constraint language $S \subseteq BR$, denoted SAT(S), is defined to be the decision problem with instance (V, C), where V is a set of Boolean variables, and C is a set of constraints $\{C_1, \ldots, C_q\}$, in which each constraint C_i is a pair (s_i, R_i) with s_i a list of variables of length m_i, called the constraint scope, and R_i an m_i-ary relation over the set $\{0,1\}$, belonging to S, called the constraint relation.*

The question is whether there exists a solution to (V, C), that is, a function from V to $\{0,1\}$ such that, for each constraint in C, the image of the constraint scope is a member of the constraint relation.

Example 2. Let R_{NAE} be the following ternary relation on $\{0,1\}$: $R_{\mathrm{NAE}} = \{0,1\}^3 \backslash \{(0,0,0),(1,1,1)\}$. It is easy to see that the well known NP-complete problem NOT-ALL-EQUAL 3-SAT can be expressed as SAT($\{R_{\mathrm{NAE}}\}$).

Using negation needs some extra care: let the *sign pattern* of a constraint $\gamma(x_1, \ldots, x_k)$ be the tuple $(s_1, \ldots, s_k)$, where $s_i = +$ if x_i is unnegated, and $s_i = -$ if x_i is negated. For each sign pattern we can then associate a relation that captures the satisfying assignments of the constraint. For example, the sign pattern of $R_{\mathrm{NAE}}(x, \neg y, \neg z)$ is the tuple $(+,-,-)$, and its associated relation is $R_{\mathrm{NAE}}^{(+,-,-)} = \{0,1\}^3 \setminus \{(1,0,0),(0,1,1)\}$. More generally, we use Γ_{NAE}^k to denote the corresponding language of not-all-equal relations (with all possible sign patterns) of length k. If ϕ is a SAT(Γ_{NAE}^k) instance we use $\gamma_{\mathrm{NAE}}^k(x_1, \ldots, x_k)$ to denote a constraint in ϕ, where each x_i is unnegated or negated. In the same manner we use Γ_{SAT}^k to denote the language consisting of all k-SAT relations of length k.

When explicitly defining relations, we often use the standard matrix representation where the rows of the matrix are the tuples in the relation. For example,

$$R_{\mathrm{NAE}} = \begin{pmatrix} 0\,0\,1 \\ 0\,1\,0 \\ 1\,0\,0 \\ 0\,1\,1 \\ 1\,0\,1 \\ 1\,1\,0 \end{pmatrix}$$

Note that the order of the rows in the matrix representation does not matter since this only correspond to a different order of the variables in a constraint.

3 Partial clones and the complexity of SAT

We will now show that the time complexity of SAT(S) is determined by the so-called *partial polymorphisms* of S. For a more in-depth background on SAT and algebraic techniques, we refer the reader to

[4] and [18], respectively. Note that most of the results in this section hold for arbitrary finite domains, but we present everything in the Boolean setting. We first note that any k-ary operation f on $\{0,1\}$ can be extended in a standard way to an operation on tuples over $\{0,1\}$, as follows: for any collection of k tuples, $t_1, t_2, \ldots, t_k \in R$, the n-tuple $f(t_1, t_2, \ldots, t_k)$ is defined as follows:

$$f(t_1, t_2, \ldots, t_k) = (f(t_1[1], t_2[1], \ldots, t_k[1]), f(t_1[2], t_2[2], \ldots, t_k[2]), \ldots, f(t_1[n], t_2[n], \ldots, t_k[n])),$$

where $t_j[i]$ is the i-th component in tuple t_j. We are now ready to define the concept of polymorphisms.

Definition 3. *Let S be a Boolean language and R an arbitrary relation from S. If f is an operation such that for all $t_1, t_2, \ldots, t_k \in R$ it holds that $f(t_1, t_2, \ldots, t_k) \in R$, then R is* closed *(or* invariant*) under f. If all relations in S are closed under f then S is closed under f. An operation f such that S is closed under f is called a* polymorphism *of S. The set of all polymorphisms of S is denoted by $Pol(S)$. Given a set of operations F, the set of all relations that are closed under all the operations in F is denoted by $Inv(F)$.*

Sets of operations of the form $Pol(S)$ are referred to as *clones*. The lattice (under set inclusion) of all clones over the Boolean domain was completely determined by Post [24] and it is usually referred to as *Post's lattice*. It is visualized in Figure 1 (see Appendix C). The following result forms the basis of the *algebraic approach* for analyzing the complexity of SAT, and, more generally, of constraint satisfaction problems. It states that the complexity of SAT(S) is determined, up to polynomial-time reductions, by the polymorphisms of S.

Theorem 4. *[15] Let S_1 and S_2 be finite non-empty sets of Boolean relations. If $Pol(S_2) \subseteq Pol(S_1)$, then SAT($S_1$) is polynomial-time reducible to SAT(S_2).*

Schaefer's classification of SAT(S) follows more or less directly from this result together with Post's lattice of clones. It is worth noting that out of the countably infinite number of Boolean clones, there are just two that corresponds to NP-complete SAT(S) problems. These are the clone I_2 consisting of all projections (i.e. the operations of the form $f_i^k(x_1, \ldots, x_k) = x_i$), and the clone N_2 consisting of all projections together with the unary negation function $n(0) = 1$, $n(1) = 0$. It is easy to realize that $Inv(I_2)$ is the set of all Boolean relations (i.e., BR) and we denote $Inv(N_2)$ by IN_2.

Theorem 4 is not very useful for studying the complexity of SAT problems in terms of their worst-case complexity as a function of the number of variables. The reason is that the reductions do not preserve the size of the instances and may introduce large numbers of new variables. It also seems that the lattice of clones is not fine grained enough for this purpose. For example, 1-in-3-SAT and k-SAT (for $k \geq 3$) both correspond to the same clone I_2.

One way to get a more refined framework is to consider partial operations in Definition 3. That is, we say that R is closed under the (partial) operation f if f applied componentwise to the tuples of R always results in a tuple from R or an undefined result (i.e., f is undefined on at least one of the components). The set of all (partial) operations preserving the relations in S, i.e., the *partial polymorphisms* of S is denoted $pPol(S)$ and forms a *partial clone*. Unlike the lattice of Boolean clones, the lattice of partial Boolean clones consists of an uncountable infinite number of partial clones, and despite being a well-studied mathematical object [18], its structure is far from being completely understood.

Before we show that the lattice of partial clones is fine-grained enough to capture the complexity of SAT(S) problems (in terms of the worst-case complexity as a function of the number of variables) we need to present a Galois connection between sets of relations and sets of (partial) functions.

Definition 5. *For any set $S \subseteq BR$, the set $\langle S \rangle$ consists of all relations that can be expressed (or implemented) using relations from $S \cup \{=\}$ (where $=$ denotes the equality relation on $\{0,1\}$), conjunction, and existential quantification. We call such implementations* primitive positive (p.p.) implementations. *Similarly, for any set $S \subseteq BR$ the set $\langle S \rangle_{\sharp}$ consists of all relations that can be expressed using relations from $S \cup \{=\}$ and conjunction. We call such implementations* quantifier-free primitive positive (q.p.p.) implementations.

Sets of relations of the form $\langle S \rangle$ and $\langle S \rangle_{\sharp}$ are referred to as relational clones (or co-clones) and partial relational clones, respectively. The lattice of Boolean co-clones is visualized in Figure 2 (see Appendix C). There is a Galois connection between (partial) clones and (partial) relational clones given by the following result.

Theorem 6. *[2,3,9,25] Let S_1 and S_2 be constraint languages. Then $S_1 \subseteq \langle S_2 \rangle$ if and only if $Pol(S_2) \subseteq Pol(S_1)$, and $S_1 \subseteq \langle S_2 \rangle_{\sharp}$ if and only if $pPol(S_2) \subseteq pPol(S_1)$.*

Finally, we show that the complexity of SAT(S), in terms of the worst-case complexity as a function of the number of variables, is determined by the lattice of partial clones.

Theorem 7. *Let S_1 and S_2 be finite non-empty sets of Boolean relations. If $pPol(S_2) \subseteq pPol(S_1)$ and SAT(S_2) is solvable in time $O(c^n)$, then SAT(S_1) is solvable in time $O(c^n)$.*

Proof. Given an instance I of SAT(S_1) on n variables we transform I into an equivalent instance I' of SAT(S_2) on at most n variables. Since S_1 is fixed and finite we can assume that the quantifier-free primitive positive implementations of every relation in S_1 by relations in S_2 has been precomputed and stored in a table (of fixed constant size). Every constraint $R(x_1, \ldots, x_n)$ in I can be represented as

$$R_1(x_{11}, \ldots, x_{1n_1}) \wedge \cdots \wedge R_k(x_{k1}, \ldots, x_{kn_k})$$

where $R_1, \ldots, R_k \in S_2 \cup \{=\}$ and $x_{11}, \ldots, x_{kn_k} \in \{x_1, x_2, \ldots, x_n\}$. Replace the constraint $R(x_1, \ldots, x_n)$ with the constraints $R_1, \ldots, R_k$. If we repeat the same reduction for every constraint in I it results in an equivalent instance of SAT($S_2 \cup \{=\}$) having at most n variables. For each equality constraint $x_i = x_j$ we replace all occurrences of x_i with x_j and remove the equality constraint. The resulting instance I' is an instance of SAT(S_2) having at most n variables. Also, note that since S_1 is finite, there cannot be more than O(n^p) constraints in total, where p is the highest arity of a relation in S_1. Thus, if SAT(S_2) is solvable in time $O(c^n)$, then SAT(S_1) is solvable in time $O(c^n)$. □

4 The easiest NP-complete SAT problem

In this section we will use the theory and results presented in the previous section to determine the easiest NP-complete SAT(S) problem. Recall that by easiest we mean that if any NP-complete SAT(S) problem can be solved in $O(c^n)$ time, then the easiest problem can be solved in $O(c^n)$ time, too. Following this lead we say that SAT(S) is easier than SAT(S) if SAT(S) is solvable in $O(c^n)$ time whenever SAT(S') is solvable in $O(c^n)$ time. A crucial step for doing this is the explicit construction of Schnoor and Schnoor [28] that, for each relational clone X, gives a relation R such that $X = \langle \{R\} \rangle$ and R has a q.p.p. implementation in every constraint language S such that $\langle S \rangle = X$. Essentially, this construction gives the bottom element of the interval of partial relational clones contained in each relational clone.

Let $\{0,1\}$-$COLS_n$ denote the Boolean relation with arity 2^n that contains all Boolean tuples from 0 to $2^n - 1$ as columns. Given two relations $R = \{r_1, \ldots, r_n\}$ and $Q = \{q_1, \ldots, q_n\}$ (of the same cardinality), then $R \circ Q$ denotes the relation that for each $r_i = (\alpha_1, \ldots, \alpha_k)$ and $q_i = (\beta_1, \ldots, \beta_l)$ contains the tuple $(\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_l)$. Assume that a set of Boolean functions $\mathcal{F}$ and a relation $R \subseteq \{0,1\}^n$ are given. The $\mathcal{F}$-closure of R, denoted $\mathcal{F}(R)$, is the relation $\bigcap_{R' \in Inv(\mathcal{F}), R \subseteq R'} R'$. Define the *extension* of a Boolean relation R, $R^{[ext]}$, to be the relation $\mathcal{F}(R \circ \{0,1\} - COLS_n)$. Finally, define the *irreduntant core* R^{irr} of $R^{[ext]}$ to be the relation obtained when identical columns are removed. In order to unambiguously define R^{irr}, we assume that the first occurrence of each column in $R^{[ext]}$ is kept in R^{irr}.

Theorem 8. *[28] Let R be a Boolean relation. Then $R^{[ext]}$ and R^{irr} have q.p.p. implementations in R. Furthermore, any language S such that $\langle S \rangle = \langle \{R\} \rangle$ implements $R^{[ext]}$ and R^{irr} via q.p.p. implementations.*

We are now in the position to define the easiest NP-complete SAT(S) problem. In the sequel, we use $b_1 \ldots b_k$ as a shorthand for the tuple $(b_1, \ldots, b_k)$. Define the relation $R_{1/3} = \{001, 010, 100\}$ and note that SAT($\{R_{1/3}\}$) corresponds to the 1-in-3-SAT problem. The relation $R_{1/3}^{\neq\neq\neq} = \{001110, 010101, 100011\}$ is formed by taking $R_{1/3}$ and adding the negation of each of the columns, i.e., $R_{1/3}^{\neq\neq\neq}$ can be defined as $R_{1/3}(x_1, x_2, x_3) \wedge (x_1 \neq x_4) \wedge (x_2 \neq x_5) \wedge (x_3 \neq x_6)$.

Lemma 9. *Let S be a language such that $\langle S \rangle = BR$. Then $R_{1/3}^{irr}$ has a q.p.p. implementation in S.*

Proof. We first construct the extension of $R_{1/3}$. Since the arity of $R_{1/3}$ is 3 we must augment the matrix representation of $R_{1/3}$ with the binary numbers from 0 to 7 as columns and close the resulting relation under every polymorphism of BR. However, the projection functions are the only polymorphisms of BR so the relation is left unchanged. Hence,

$$R_{1/3}^{[ext]} = \begin{pmatrix} 0\,0\,1\,0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,0\,0\,0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,0\,0\,1\,0\,1\,0\,1\,0\,1 \end{pmatrix}$$

The irredundant core is then obtained by removing identical columns.

$$R_{1/3}^{irr} = \begin{pmatrix} 0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,1\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,0\,1\,0\,1 \end{pmatrix}$$

□

The relation $R_{1/3}^{irr}$ is nothing else than $R_{1/3}^{\neq\neq\neq}$ with the two constant columns adjoined, and there is a trivial reduction from SAT($R_{1/3}^{\neq\neq\neq}$) to SAT($R_{1/3}^{irr}$) introducing only 2 extra variables (one for each constant). Combining this with Theorem 7 and 9 we immediately obtain the following result:

Lemma 10. *Let S be a language such that $\langle S\rangle = BR$. Then SAT($R_{1/3}^{\neq\neq\neq}$) is not harder than SAT(S).*

We are left with the relational clone IN_2 and need to make sure that the bottom partial relational clone in IN_2 is not (strictly) easier than $R_{1/3}^{\neq\neq\neq}$. We proceed in an analogous manner to the derivation of $R_{1/3}^{\neq\neq\neq}$, and define a maximal extended relation which is then pruned of superfluous columns. Let R_{NAE} be defined as in Section 2. Analogously to the relation $R_{1/3}^{\neq\neq\neq}$ in BR, we consider the relation $R_{2/4}^{\neq\neq\neq\neq} = \{00111100,01011010,10010110,\ 11000011,10100101,\ 01101001\}$ in IN_2. The proof can be found in Appendix A.

Lemma 11. *Let S be a language such that $\langle S\rangle = IN_2$. Then S implements $R_{2/4}^{\neq\neq\neq\neq}$ via a q.p.p. implementation.*

Both $R_{1/3}^{\neq\neq\neq}$ and $R_{2/4}^{\neq\neq\neq\neq}$ can be viewed as candidates for the easiest NP-complete languages. In order to prove that $R_{1/3}^{\neq\neq\neq}$ is not harder than $R_{2/4}^{\neq\neq\neq\neq}$ we must give a size-preserving reduction from SAT($R_{1/3}^{\neq\neq\neq}$) to SAT($R_{2/4}^{\neq\neq\neq\neq}$).

Lemma 12. *SAT($R_{1/3}^{\neq\neq\neq}$) is not harder than SAT($R_{2/4}^{\neq\neq\neq\neq}$).*

Proof. Let ϕ be an instance of SAT($R_{1/3}^{\neq\neq\neq}$) and $C = R_{1/3}^{\neq\neq\neq}(x_1, x_2, x_3, x_4, x_5, x_6)$ be an arbitrary constraint in ϕ. Let Y_1 and Y_2 be two global variables. Then the constraint $C' = R_{2/4}^{\neq\neq\neq\neq}(x_1, x_2, x_3, Y_1, x_4, x_5, x_6, Y_2)$ is satisfiable if and only if C is satisfiable, with $Y_1 = 1$ and $Y_2 = 0$ (we may assume that $Y_1 = 1$ since the complement of a satisfiable assignment is also a satisfiable assignment for languages in IN_2). If we repeat this reduction for every constraint in ϕ we get a SAT($R_{2/4}^{\neq\neq\neq\neq}$) instance which is satisfiable if and only if ϕ is satisfiable. Since the reduction only introduces two new variables, it follows that an $O(c^n)$ algorithm for SAT($R_{2/4}^{\neq\neq\neq\neq}$) can be used to solve SAT($R_{1/3}^{\neq\neq\neq}$) in $O(c^n)$ time, too. □

Since S is NP-complete if and only if $\langle S\rangle = BR$ or $\langle S\rangle = IN_2$, Lemma 10 together with Lemma 12 gives that SAT($R_{1/3}^{\neq\neq\neq}$) is the easiest NP-complete SAT problem.

Theorem 13. *Let S be an NP-complete Boolean language. Then SAT($R_{1/3}^{\neq\neq\neq}$) is not harder than SAT(S).*

5 SAT problems with bounded degree

In this section, we investigate the SAT problem where restrictions are placed on the number of occurrences per variable. If x occurs in k constraints then we say that the *degree* of x is k. If S is a language, then SAT(S)-k is the SAT(S) problem where the degree of each variable is at most k. This restriction is of particular interest since, for all languages S such that SAT(S) is NP-complete, SAT(S)-k is NP-complete for some k.

Theorem 14. *[16] For any fixed S such that CSP(S) is NP-complete, there is an integer k such that CSP(S)-k is NP-complete.*

The most interesting case is when k is the smallest k such that SAT(S)-k is NP-complete. These values are already known for 1-in-3-SAT: for $k = 2$ it can be reduced to the problem of finding a perfect matching in a graph [14], but for $k = 3$ it is NP-complete even for planar instances [21]. It might be expected that the same holds for SAT($R_{1/3}^{\neq\neq\neq}$) since it is easier than SAT($R_{1/3}$). This is however not the case: SAT($R_{1/3}^{\neq\neq\neq}$)-k is NP-complete even for $k = 2$. To prove this we first note that $R_{1/3}^{\neq\neq\neq}$ is not a Δ-matroid relation.

Definition 15. ***(Δ-matroid relation)*** *Let R be a Boolean relation and x, y, x' be Boolean tuples of the same arity. Let $d(x, y)$ be the binary difference function between x and y. Then x' is a step from x to y if $d(x, x') = 1$ and $d(x, y) = d(x', y) + d(x, x')$. R is a* Δ-matroid *relation if it satisfies the following axiom: $\forall x, y \in R\, \forall x'.(x'$ is a step from x to $y) \rightarrow (x' \in R \vee \exists x'' \in R$ which is a step from x' to $y)$.*

Lemma 16. *$R_{1/3}^{\neq\neq\neq}$ is not a Δ-matroid relation.*

Proof. Let $x = 001110$ and $y = 010101$. These are both elements in $R_{1/3}^{\neq\neq\neq}$. Let $x' = 000110$. Then $d(x, x') = 1$, $d(x, y) = 4 = d(x, x') + d(x', y) = 1 + 3 = 4$, whence x' is a step from x to y. For $R_{1/3}^{\neq\neq\neq}$ to be a Δ-matroid relation either $x' \in R_{1/3}^{\neq\neq\neq}$, or there exists a x'' which is a step from x' to y. Since neither of the disjuncts are true, it follows that $R_{1/3}^{\neq\neq\neq}$ is not a Δ-matroid relation. □

The hardness result then follows from Theorem 3 in Dalmau and Ford [8], which states that SAT(S)-2 is NP-complete if S contains a relation that is not Δ-matroid.

Theorem 17. *SAT($R_{1/3}^{\neq\neq\neq}$)-2 is NP-complete.*

6 The exponential-time hypothesis

Even though $R_{1/3}^{\neq\neq\neq}$ is the easiest NP-complete language, we cannot hope to prove or disprove that SAT($R_{1/3}^{\neq\neq\neq}$) or SAT($R_{1/3}^{\neq\neq\neq}$)-2 is solvable in polynomial time since this would settle the P = NP question. A more assailable question is if the problem can be solved in *sub-exponential* time. If yes, then we are none the wiser; but if no, then P ≠ NP. As a tool for studying sub-exponential problems, Impagliazzo et. al [13] proved a *sparsification* lemma for k-SAT. Intuitively, the process of sparsification means that a SAT instance with a large number of constraints can be expressed as a disjunction of instances with a comparably small number of constraints. We prove that sparsification is possible not only for k-SAT, but between all NP-complete, finite languages, and use this to prove that SAT($R_{1/3}^{\neq\neq\neq}$)-2 is sub-exponential if and only if the exponential-time hypothesis is false. Due to sparsification we can also prove that *all* NP-complete languages are sub-exponential if and only if *one* NP-complete language is sub-exponential (and that this holds also in the degree-bounded case), which is a significant refinement of Impagliazzo et. al's result when restricted to finite Boolean languages.

6.1 Preliminaries

There has been a stride in constructing faster exponential algorithms for NP-complete problems. A natural question to ask is whether there exists a constant c such that a problem is solvable in $O(c^n)$, but not for any c' smaller than c. Problems without such a sharp limit are said to be sub-exponential.

Definition 18. *A language S is sub-exponential if SAT(S) is solvable in $O(2^{\epsilon n})$ for all $\epsilon > 0$.*

We now need a class of reductions that relates languages based on their sub-exponential complexity. Reductions based on q.p.p. definitions are however to precise to fully encompass this since they preserve exact complexity — a reduction should be able to introduce new variables as long as the resulting instance can be solved in sub-exponential time. We introduce *linear variable reductions*, which should be compared to the more complex but general class of SERF-reductions from Impagliazzo et. al [13].

Definition 19. *Let S and S' be two finite languages and ϕ a SAT(S) instance with n variables. A total function f from SAT(S) to SAT(S') is a* many-one linear variable reduction, or an LV-reduction, if:

1. ϕ is satisfiable if and only if $f(\phi)$ is satisfiable,
2. the number of variables in $f(\phi)$, n', is only increased by a linear amount, i.e. there exists a fixed constant C such that $n' \leq Cn$, and
3. $f(\phi)$ can be computed in $O(poly(n))$ time.

The point of the definition is that an LV-reduction between two languages preserves sub-exponentiality. Hence, if SAT(S) is sub-exponential and we know that SAT(S') is LV-reducible to SAT(S), then SAT(S') is sub-exponential as well. The proof is straightforward and is included in Appendix B.

Lemma 20. *Let S and S' be two finite languages such that SAT(S) is sub-exponential. If there exists an LV-reduction from SAT(S') to SAT(S), then SAT(S') is sub-exponential.*

Let S and S' be two finite languages such that $S \subseteq \langle S' \rangle$. We can then reduce SAT(S) to SAT(S') by replacing each constraint from S by its equivalent implementation in S'. Such a reduction would need $C \cdot m$ new variables, where C is a constant that only depends on S', and m the number of constraints in the instance. If m is large compared to the number of variables, n, this would however require more than a linear amount of new variables. We can therefore only prove that LV-reductions exist for classes of problems where m is linearly bounded by the number of variables. The proof is in Appendix B.

Lemma 21. *Let S and S' be two finite languages such that SAT(S) and SAT(S')-k are NP-complete for some k. If $S' \subseteq \langle S \rangle$, then SAT($S'$)-$k$ is LV-reducible to SAT(S).*

This does not imply that there exists LV-reductions between SAT(S) and SAT(S') for all NP-complete languages S and S' since these problems are not degree-bounded, but it is a useful tool in the sparsification process.

Definition 22. *Let S and S' be two finite languages. We say that S is sparsifiable into S' if, for all $\epsilon > 0$ and for all SAT(S) instance ϕ (with n variables), ϕ can be expressed by a disjunctive formula $\bigvee_{i=1}^{t} \phi_i$, where:*

1. *ϕ is satisfiable if and only if at least one ϕ_i is satisfiable,*
2. *k is a constant that only depends on ϵ, S and S',*
3. *ϕ_i is a SAT(S')-k instance,*
4. *$t \leq 2^{\epsilon n}$, and*
5. *$\bigvee_{i=1}^{t} \phi_i$ can be computed in $O(poly(n) \cdot 2^{\epsilon n})$ time.*

Note that nothing in the definition says that S and S' cannot be the same language. If so, we simply say that S is *sparsifiable*. Impagliazzo et. al [13] prove the following for k-SAT.

Lemma 23. ***(sparsification lemma for k-SAT)*** *k-SAT is sparsifiable.*

6.2 General sparsification

Recall from Section 2 that we use Γ^k_{SAT} and Γ^k_{NAE} to denote the languages of k-SAT and NAE-k-SAT respectively. In order to prove that sparsification is possible between all NP-complete, finite languages we first prove that Γ^k_{NAE} is sparsifiable, and then that all NP-complete languages in BR and IN_2 can be sparsified by reducing them to either Γ^k_{SAT} or Γ^k_{NAE}.

Lemma 24. ***(sparsification lemma for NAE-k-SAT)*** *Γ^k_{NAE} is sparsifiable.*

Proof. Let ϕ be a SAT(Γ^k_{NAE}) instance with n variables. If $\gamma^k_{\mathrm{NAE}}(x_1, \ldots, x_k)$ is a constraint from ϕ it can be verified that it is satisfiable if and only if $\gamma^k_{\mathrm{SAT}}(x_1, \ldots, x_k) \wedge \gamma^k_{\mathrm{SAT}}(\neg x_1, \ldots, \neg x_k)$ is satisfiable. We can therefore form an equivalent SAT(Γ^k_{SAT}) instance ψ by adding the complement of every γ^k_{NAE}-constraint. By the sparsification lemma for k-SAT, it then follows that ψ can be sparsified into the disjunctive formula $\bigvee_{i=1}^{t} \psi_i$. We must now prove that each ψ_i is reducible to an equivalent SAT(Γ^k_{NAE})-l instance, for some constant l that does not depend on n.

For simplicity, we shall first reduce each disjunct to SAT($\Gamma^{k+1}_{\mathrm{NAE}}$). For each constraint $\gamma^k_{\mathrm{SAT}}(x_1, \ldots, x_k) \in \psi_i$ we let $\gamma^{k+1}_{\mathrm{NAE}}(x_1, \ldots, x_k, X)$ be the corresponding $\gamma^{k+1}_{\mathrm{NAE}}$-constraint, where X is a fresh variable common to all constraints. Let ψ'_i be the resulting SAT($\Gamma^{k+1}_{\mathrm{NAE}}$) instance. Then ψ_i is satisfiable if and only if ψ'_i is satisfiable: if ψ_i is satisfiable, then ψ'_i is satisfiable with $X = 0$; if ψ'_i is satisfiable we may assume

that $X = 0$ since the complement of each valid assignment is also a valid assignment. But then each constraint has at least one literal that is not 0, by which it follows that ψ_i must be satisfiable.

Since ψ was sparsified, the degree of the variables in ψ_i is bounded by some constant C. Hence X cannot occur in more than $C \cdot n$ constraints. We now prove that the degree of X can be reduced to a constant value. Since $\langle \Gamma^{k+1}_{\mathrm{NAE}} \rangle = IN_2$ we can implement an equality relation that has the form $Eq(x, y) \equiv \exists z_1, \ldots, z_T.\theta$, where θ is a conjunction of constraints over $\Gamma^{k+1}_{\mathrm{NAE}}$. Let V denote the highest degree of any variable in θ. We may without loss of generality assume that $2V < C$ since we can otherwise adjust the ϵ-parameter in the sparsification process.

To decrease the degree of X we introduce the fresh variables $X'_1, \ldots, X'_W$ in place of X and the following chain of equality constraints: $Eq(X, X'_1) \wedge Eq(X'_1, X'_2) \wedge \ldots \wedge Eq(X'_{W-1}, X'_W)$. Let the resulting formula be ψ''_i. Then ψ'_i is satisfiable if and only if ψ''_i is satisfiable since $X = X'_1 = \ldots = X'_W$ in all models. Then $W = \frac{C \cdot n}{C - 2V}$ new variables are needed since each X'_i can occur in $C - 2V$ additional constraints. Since each equality constraint requires T variables the whole equality chain requires $\frac{C \cdot n \cdot T}{C - 2V}$ variables which is linear with respect to n since T, C and V are constants.

But since $\langle \Gamma^{k+1}_{\mathrm{NAE}} \rangle = \langle \Gamma^{k}_{\mathrm{NAE}} \rangle = IN_2$ and no variable occurs more than C times we can use Lemma 21 and LV-reduce ψ''_i to an equivalent SAT(Γ^k_{NAE}) instance ϕ_i. Since all variables in ψ''_i are degree bounded by C there exists a constant l determined by C and Γ^k_{NAE} such that no variable in ϕ_i occurs in more than l constraints, i.e. ϕ_i is an instance of SAT(Γ^k_{NAE})-l. It now follows that ϕ is satisfiable if and only if at least one of the disjuncts ϕ_i is satisfiable. Hence Γ^k_{NAE} is sparsifiable. □

The proof of the following auxiliary lemma can be found in Appendix B.

Lemma 25. *Let S be a finite language such that $S \subseteq IN_2$ and let S' be a finite language such that $S' \subseteq BR$. Then, SAT(S) is LV-reducible to SAT(Γ^k_{NAE}), for some k dependent on S, and SAT(S') is LV-reducible to SAT($\Gamma^{k'}_{\mathrm{SAT}}$), for some k' dependent on S'.*

Since a language S is NP-complete if and only if $\langle S \rangle = BR$ or $\langle S \rangle = IN_2$, we can now prove that sparsification is possible between all finite NP-complete languages.

Theorem 26. ***(sparsification between all finite NP-complete languages)*** *Let S and S' be two finite languages such that SAT(S) and SAT(S') are NP-complete. Then, SAT(S) is sparsifiable into SAT(S').*

Proof. There are a few different cases depending on which co-clones that are generated by S and S': (1) $\langle S \rangle = \langle S' \rangle = IN_2$, (2) $\langle S \rangle = \langle S' \rangle = BR$, (3) $\langle S \rangle = IN_2$, $\langle S' \rangle = BR$, and (4) $\langle S \rangle = BR$, $\langle S' \rangle = IN_2$.

For case (1), assume that $\langle S \rangle = \langle S' \rangle = IN_2$. Let p denote the highest arity of a relation in S. If ϕ is a SAT(S) instance with n variables it can be reduced to a SAT(Γ^p_{NAE}) instance ϕ' with the same number of variables by Lemma 25. Then, according to the sparsification lemma for NAE-k-SAT, there exists a disjunction of SAT(Γ^k_{NAE})-l formulas such that $\phi' = \bigvee_{i=1}^{t} \phi_i$. Since $\langle S' \rangle = IN_2$, each ϕ_i can be implemented as a conjunction of constraints over S' with a linear amount of extra constraints and variables. Let ϕ'_i denote each such implementation. Then ϕ'_i is an instance of SAT(S')-l', for some l' determined by l and S'. Hence SAT(S) is sparsifiable into SAT(S').

Case (2) is analogous to case (1) but with Γ^k_{SAT} instead of Γ^k_{NAE}. Case (3) follows from case (2) since all finite languages are sparsifiable into Γ^k_{SAT} by Lemmas 23 and 25.

For case (4), assume that $\langle S \rangle = BR$ and $\langle S' \rangle = IN_2$. Let p denote the relation with the highest arity in S. If ϕ is a SAT(S) instance with n variables it can be LV-reduced to a SAT(Γ^p_{SAT}) instance ϕ' by lemma 25. Since Γ^k_{SAT} is sparsifiable there exists a disjunction such that $\phi' = \bigvee_{i=1}^{t} \phi_i$. By recapitulating the steps from Lemma 24 we can then reduce each ϕ_i to a SAT($\Gamma^{k+1}_{\mathrm{NAE}}$)-$l$ instance ϕ'_i. Then, since $\langle S' \rangle = IN_2$, each ϕ'_i can be implemented as a conjunction of constraints over S' such that no variable occurs in more than l' constraints, where l' is determined by l and S'. □

Santhanam and Srinivasan [26] have shown that the unrestricted SAT problem (which corresponds to an infinite constraint language) does not admit sparsification to arbitrary finite NP-complete languages. Consequently, it is a necessary condition that the languages in Theorem 26 are indeed finite.

6.3 SAT and the exponential-time hypothesis

The exponential-time hypothesis states that k-SAT is not sub-exponential [12] for $k \geq 3$. If one assumes that P $\neq$ NP, then this statement is plausible since it enforces a limit on the time complexity of

exponential algorithms. A problem that is sub-exponential if and only if k-SAT is sub-exponential is said to be *ETH-hard*. Impagliazzo et. al prove that many NP-complete problems such as k-colorability, clique and vertex cover are ETH-hard. An ETH-hard problem has the property that it is sub-exponential if and only if all problems in SNP are sub-exponential. In this section we prove that both SAT($R_{1/3}^{\neq\neq\neq}$) and SAT($R_{1/3}^{\neq\neq\neq}$)-2 are ETH-hard, and this implies that all finite, NP-complete languages are ETH-hard even for degree-bounded instances. This result does not exclude the possibility that there exist an extremely simple NP-complete problem that is not ETH-hard, it seems that such a problem is unlikely to reside in the Boolean domain.

Theorem 27. *The following statements are equivalent:*

1. *The exponential-time hypothesis is false.*
2. *SAT($R_{1/3}^{\neq\neq\neq}$)-2 is sub-exponential.*
3. *SAT($R_{1/3}^{\neq\neq\neq}$) is sub-exponential.*
4. *For every NP-complete and finite language S, SAT(S) is sub-exponential.*
5. *For every NP-complete and finite language S, SAT(S)-k is sub-exponential for every k.*
6. *There exists an NP-complete and finite language S such that SAT(S) is sub-exponential.*
7. *There exists an NP-complete and finite language S such that SAT(S)-k is sub-exponential for all k.*

Proof. We will prove that $1 \implies \ldots \implies 7 \implies 1$ and hence that all statements are equivalent.

$1 \implies 2$: If the exponential-time hypothesis is false then k-SAT is sub-exponential. But then $R_{1/3}^{\neq\neq\neq}$ must also be sub-exponential since it is the easiest NP-complete language. This immediately implies that SAT($R_{1/3}^{\neq\neq\neq}$)-2 is sub-exponential.

$2 \implies 3$: We must prove that SAT($R_{1/3}^{\neq\neq\neq}$) is sub-exponential if SAT($R_{1/3}^{\neq\neq\neq}$)-2 is sub-exponential. Let ϕ be a SAT($R_{1/3}^{\neq\neq\neq}$) instance with n variables. Due to Theorem 26 there exists a disjunction of SAT($R_{1/3}^{\neq\neq\neq}$)-l instances such that $\phi = \bigvee_{i=1}^{t} \phi_i$, where l is a constant that does not depend on n. Next assume that x is a variable in ϕ_i that occurs in $2 < k \leq l$ constraints. Since this is not a valid SAT($R_{1/3}^{\neq\neq\neq}$)-2 instance the degree of x must be lowered to 2.

Call two variables in an $R_{1/3}^{\neq\neq\neq}$-constraint *complementary* if one occurs in position i = 1, 2 or 3, and the other in position $i + 3$. It is easily verified that variables fulfilling this constraint are indeed each other's complement. Let $C_1, \ldots, C_k$ be an enumeration of the constraints that contains x. Without loss of generality we may assume that x always occur in position 1 and that it has a single complementary variable x' which occurs in position 4 in the same k constraints. For each three constraints,

$C_{i-1} = R_{1/3}^{\neq\neq\neq}(x, y_{i-1}, z_{i-1}, x', y'_{i-1}, z'_{i-1})$,
$C_i = R_{1/3}^{\neq\neq\neq}(x, y_i, z_i, x', y'_i, z'_i)$, and
$C_{i+1} = R_{1/3}^{\neq\neq\neq}(x, y_{i+1}, z_{i+1}, x', y'_{i+1}, z'_{i+1})$,

we can then lower the degree of x and x' by introducing the constraints C'_{i-1}, C'_i, and C'_{i+1}, which we define such that

$C'_{i-1} = R_{1/3}^{\neq\neq\neq}(x, y_{i-1}, z_{i-1}, x', y'_{i-1}, z'_{i-1})$,
$C'_i = R_{1/3}^{\neq\neq\neq}(x, y_i, z_i, x'', y'_i, z'_i)$, and
$C'_{i+1} = R_{1/3}^{\neq\neq\neq}(x''', y_{i+1}, z_{i+1}, x'', y'_{i+1}, z'_{i+1})$.

Here, x'' and x''' are fresh variables. Since the new variables occur in the same complementary positions it follows that $x = x''' = \neg x' = \neg x''$, and that C_{i-1}, C_i, C_{i+1} are satisfiable if and only if C'_{i-1}, C'_i, C'_{i+1} are satisfiable. If the procedure is repeated iteratively for all $C_1, \ldots, C_k$ the degree of x or any newly introduced variable in $C'_1, \ldots, C'_k$ is at most 2. Let ϕ'_i denote the formula obtained when the procedure is repeated for all variables occurring k times, $2 < k \leq l$. The total number of variables needed is then bounded by the linear expression $l'n$, where l' is a constant determined by l.

Since no variable in ϕ'_i occurs in more than two constraints, we can then use a sub-exponential algorithm for SAT($R_{1/3}^{\neq\neq\neq}$)-2, and answer yes if and only if at least one ϕ'_i is satisfiable. Hence SAT($R_{1/3}^{\neq\neq\neq}$) is sub-exponential if SAT($R_{1/3}^{\neq\neq\neq}$)-2 is sub-exponential.

$3 \implies 4$: Assume that $R_{1/3}^{\neq\neq\neq}$ is sub-exponential. Let S be an arbitrary finite language and ϕ be an instance of SAT(S). According to the general sparsification result (Theorem 26), ϕ can be sparsified into $\bigvee_{i=1}^{t} \phi_i$, where each ϕ_i is an instance of SAT($R_{1/3}^{\neq\neq\neq}$)-l. But then we can simply use a sub-exponential algorithm for SAT($R_{1/3}^{\neq\neq\neq}$) and answer yes if and only if at least one of the disjuncts is satisfiable.

$4 \implies 5$: Trivial.

5 $\Longrightarrow$ 6: Assume that SAT(S) is NP-complete and that SAT(S)-k is sub-exponential for every k. Let ϕ be a SAT(S) instance. Then according to Theorem 26, there exists a disjunction $\bigvee_{i=1}^{t} \phi_i$ of SAT(S)-l formulas, for some constant l. Since SAT(S)-k is sub-exponential for every k we can use a sub-exponential algorithm for SAT(S)-l and answer yes if and only if at least one of the disjuncts is satisfiable.

6 $\Longrightarrow$ 7: Trivial.

7 $\Longrightarrow$ 1: Assume that SAT(S) is NP-complete and that SAT(S)-k is sub-exponential for all k. According to Theorem 26), any instance of SAT(Γ^k_{SAT}) can be expressed as a disjunction of SAT(S)-l instances for some constant l. But since SAT(S)-k is sub-exponential for all k we can simply use a sub-exponential algorithm for SAT(S)-l and answer yes if and only if at least one of the disjuncts is satisfiable. □

7 Research directions and open questions

We will now discuss some research directions and pose some open questions. After having shown that SAT($R^{\neq\neq\neq}_{1/3}$) is the easiest NP-complete SAT problem, it is tempting to try to determine bounds on its time complexity. We are currently working on a branch-and-bound algorithm for this problem and preliminary results show that this algorithm runs in $O(\alpha^n)$ time where $\alpha \approx 1.05$. We are fairly convinced that the time complexity can be significantly lowered by a more careful analysis of the algorithm.

We have proved that SAT(S) is sub-exponential if and only if SAT(S)-k is sub-exponential for all k. This result is inconclusive since it does not rule out the possibility that a language is sub-exponential and NP-complete for some k, but that the sub-exponential property is lost for larger values. Hence, it would be interesting to (dis)prove that SAT(S) is sub-exponential if and only if there exists some k such that SAT(S)-k is NP-complete and sub-exponential. This holds for SAT($R^{\neq\neq\neq}_{1/3}$) so it does not seem impossible that the result holds for all languages. We also remark that bounding the degree of variables is not the only possible structural restriction: many attempts at establishing structurally based complexity results are based on the tree-width (or other width parameters) of some graph representation of the constraints, cf. [7,10]. A particularly interesting example is Marx's [20] result that connects ETH with structural restrictions: if ETH holds, then solving the CSP problem for instances whose primal graph has treewidth k requires $n^{\Omega(k/\log k)}$ time.

A natural continuation of this research is to generalize the methods in Section 3 to other problems. Generalizing them to constraint satisfaction problems over finite domains appears to be effortless, and such a generalisation would give us a tool for studying problems such as k-colourability and its many variations. Lifiting the results to infinite-domain constraints appears to be more difficult, but it may be worthwhile: Bodirsky and Grohe [1] have shown that *every* computational decision problem is polynomial-time equivalent to such a constraint problem. Hence, this may lead to general methods for studying the time complexity of computational problems. Another interesting generalisation is to study problems that are not satisfiability problems, e.g. enumeration problems, counting problems, and non-monotonic problems such as abduction and circumscription.

As we have already mentioned, a drawback of Theorem 7 is that the structure of the Boolean partial clone lattice is far from well-understood (and even less well-understood when generalized to larger domains). Hence, it would be interesting to look for lattices that have a granularity somewhere in between the clone lattice and the partial clone lattice. One plausible candidate is the lattice of *frozen* partial clones that were introduced in [23]. A *frozen* implementations is a primitive positive implementation where we are only allowed to existentially quantify over variables that are frozen to a constant (i.e., variables that are constant over all solutions). For more details about frozen partial clones (e.g., the Galois connection between frozen partial clones and frozen partial relational clones), we refer the reader to [23]. We remark that the complexity of SAT(S) is determined by the frozen partial clones and that the lattice of frozen partial clones is indeed coarser than the lattice than the lattice of partial clones as there are examples of infinite chains of partial clones that collapse to a single frozen partial clone [11,23].

Acknowledgments

We thank Magnus Wahlström for helpful discussions on the topic of this paper.

References

1. M. Bodirsky and M. Grohe. Non-dichotomies in constraint satisfaction complexity. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP-2008)*, pp. 184–196, 2008.
2. V. Bodnarchuk, L. Kaluzhnin, V. Kotov, and B. Romov. Galois theory for post algebras. I. *Cybernetics and Systems Analysis*, 5(3):1–10, 1969.
3. V. Bodnarchuk, L. Kaluzhnin, V. Kotov, and B. Romov. Galois theory for post algebras. II.. *Cybernetics and Systems Analysis*, 5(5):1–9, 1969.
4. E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with boolean blocks, part I: Post's lattice with applications to complexity theory. *ACM SIGACT-Newsletter*, 34, 2003.
5. A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.
6. M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNFSAT. In *Proceedings of the 27th Annual IEEE Conference on Computational Complexity (CCC-2012)*. To appear. Preprint available from `http://arxiv.org/abs/1112.2275`
7. V. Dalmau, Ph. Kolaitis, and M. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP-2002)*, pp. 310–326, 2002.
8. V. Dalmau and D. Ford. Generalized satisfiability with limited occurrences per variable: A study through delta-matroid parity. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS-2003)*, pp. 358–367, 2003.
9. D. Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, pp. 228–250, 1968.
10. M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1), 2007.
11. L. Haddad. Infinite chains of partial clones containing all selfdual monotonic partial functions. *Multiple-valued Logic and Soft Computing*, 18(2):139–152, 2012.
12. R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.
13. R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
14. G. Istrate. Looking for a version of Schaefer's dichotomy theorem when each variable occurs at most twice. Technical report 652, Computer Science Department, The University of Rochester, 1997.
15. P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.
16. P. Jonsson, A. Krokhin, and F. Kuivinen. Hard constraint satisfaction problems have hard gaps at location 1. *Theoretical Computer Science*, 410(38-40):3856–3874, 2009.
17. R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, 1975.
18. D. Lau. *Function Algebras on Finite Sets*. Springer, Berlin, 2006.
19. D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.
20. D. Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010.
21. C. Moore and J. Robson. Hard tiling problems with simple tiles. *Discrete & Computational Geometry*, 26(4):573–590, 2001.
22. R. Moser and D. Scheder. A full derandomization of Schoening's k-SAT algorithm. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC-2011)*, pp. 245–252, 2011.
23. G. Nordh and B. Zanuttini. Frozen Boolean partial co-clones. In *Proceedings of the 39th International Symposium on Multiple-Valued Logic (ISMVL-2009)*, pp. 120–125, 2009.
24. E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.
25. B. Romov. The algebras of partial functions and their invariants. *Cybernetics and Systems Analysis*, 17(2):157–167, 1981.
26. R. Santhanam and S. Srinivasan. On the limits of sparsification. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP-2012)*. To appear. Preprint available from `http://eccc.hpi-web.de/report/2011/131/`

27. Th. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC-1978)*, pp. 216–226, 1978.
28. H. Schnoor and I. Schnoor. New algebraic tools for constraint satisfaction. In N. Creignou, Ph. Kolaitis, and H. Vollmer, editors, *Complexity of Constraints*, Schloss Dagstuhl, Germany.
29. U. Schöning. A low and a high hierarchy within NP. *Journal of Computer and System Sciences*, 27(1):14–28, 1983.
30. Á. Szendrei. *Clones in Universal Algebra*, volume 99 of *Seminaires de Mathématiques Supérieures*. University of Montreal, 1986.
31. J. van Rooij, M. van Kooten Niekerk, and H. Bodlaender. Partition into triangles on bounded degree graphs. In *Proceedings of SOFSEM 2011: Theory and Practice of Computer Science*, pp. 558–569, 2011.
32. M. Wahlström. *Algorithms, Measures and Upper Bounds for Satisfiability and Related Problems*. PhD thesis, Linköping University, 2007.
33. G. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial Optimization - Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–208, 2003.

A Additional proofs for section 4

Proof. (Lemma 11) Let R_{NAE} be not-all-equal-SAT as defined in Section 2 and recall that $\langle R_{\mathrm{NAE}} \rangle = IN_2$. By Theorem 8 it is enough to show that the irreduntant core R^{irr} of $R_{\mathrm{NAE}}^{[ext]}$ can implement $R_{2/4}^{\neq\neq\neq\neq}$ with a p.p.q implementation. Since the cardinality of R_{NAE} is 6, R^{irr} will have arity $2^6 = 64$, consist of 12 tuples where the columns of the six first tuples are the binary numbers from 0 to 63, and the six last tuples the complements of the six first. The matrix representation is therefore as follows.

$$R^{irr} = \begin{pmatrix} 0\,0\cdots 1\,1 \\ 0\,0\cdots 1\,1 \\ 0\,0\cdots 1\,1 \\ 0\,0\cdots 1\,1 \\ 0\,0\cdots 1\,1 \\ 0\,1\cdots 0\,1 \\ \vdots\,\vdots\,\ddots\,\vdots \\ 1\,0\cdots 1\,0 \end{pmatrix}$$

Let $x_1, \ldots, x_{64}$ denote the columns in the matrix and the variables in the relation. We must now prove that R^{irr} can implement $R_{2/4}^{\neq\neq\neq\neq}$ with a p.p.q implementation. Note that x_1 and x_2 only differ in one row. If we identified x_1 and x_2 we would therefore get a relation where that tuple was removed. But since $R_{2/4}^{\neq\neq\neq\neq}$ only has 6 tuples we want to identify two columns that differ in 6 positions.

$$x_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, x_8 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Therefore identifying x_1 and x_8 will remove rows 4, 5, 6 and 10, 11, 12 from R^{irr}. If we then collapse identical columns the resulting matrix will be:

$$\begin{pmatrix} 0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,1\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,0\,1\,0\,1 \\ 1\,1\,1\,1\,0\,0\,0\,0 \\ 1\,1\,0\,0\,1\,1\,0\,0 \\ 1\,0\,1\,0\,1\,0\,1\,0 \end{pmatrix}$$

It can be verified that this relation is nothing else than a rearranged version of $R_{2/4}^{\neq\neq\neq\neq}$. Hence, since S can p.p.q. define R^{irr}, which in turn can p.p.q. define $R_{2/4}^{\neq\neq\neq\neq}$, it follows that S can also p.p.q. define $R_{2/4}^{\neq\neq\neq\neq}$. □

B Additional proofs for section 6

Proof. (Lemma 20) Assume that SAT(S') can be solved in time $O(c^n)$ but not in $O(c^{n\epsilon})$ for any $0 < \epsilon < 1$. Since SAT(S) is sub-exponential it can be solved in time $O(c^{\epsilon n})$ for all $\epsilon > 0$. Assume that the LV-reduction from SAT(S') to SAT(S) implies that the resulting instance contains at most $C \cdot n$ variables where C is a constant. This will make SAT(S') solvable in time $O(c^{(C \cdot n)\epsilon})$ for all $\epsilon > 0$ which contradicts the assumption. □

Proof. (Lemma 21) Let ϕ be a SAT(S')-k-instance with n variables. Since each variable can occur in at most k constraints there cannot be more than $n \cdot k$ constraints in total. Each such constraint is of the form $R(x_1, \ldots, x_i)$ where $R \in S'$. By assumption R can then be expressed as a conjunction of constraints over $S \cup \{=\}$ with a set of existentially quantified variables: $\exists y_1, \ldots, y_j. \bigwedge \psi(Y)$, where each $\psi \in S \cup \{=\}$ and $Y \subseteq \{x_1, \ldots, x_i\} \cup \{y_1, \ldots, y_j\}$.

Hence the number of extra variables for each constraint depends on the relations from S'. Let t denote the largest amount of variables that is required for implementing a constraint. In the worst case the total amount of new variables in the reduction is then $(n \cdot k)t$, which is linear with respect to n since k and t are fixed values.

Since the reduction only increases the amount of variables with a linear factor it is indeed an LV-reduction, which concludes the lemma. □

Proof. (Lemma 25) Let ϕ be an instance of SAT(S) with n variables, and let $R \in S$ be a relation with arity p. By definition, $R = \{0,1\}^p \setminus E$, where E is a set of p-ary tuples over $\{0,1\}$ that describes the excluded tuples in the relation. Since all relations in S must be closed under complement we can partition E into E_1 and E_2 where each tuple in E_2 is the complement of a tuple in E_1.

Let $|E_1| = |E_2| = N$ and $e_1, \ldots, e_N$ be an enumeration of the elements in E_1. Let $e_i = (b_{i,1}, \ldots, b_{i,p})$, $b_{i,j} \in \{0,1\}$.

If $R(x_1, \ldots, x_p)$ is a constraint in ϕ, then it can be expressed by the SAT(Γ^p_{NAE}) formula $\psi_1 \wedge \ldots \wedge \psi_N$, where each $\psi_i = \gamma^p_{\mathrm{NAE}}(y_1, \ldots, y_p)$, and $y_j = x_j$ if $b_{i,j}$ is 0, and $y_j = \neg x_j$ if $b_{i,j}$ is 1. Each such constraint represents one of the excluded tuples in E_1 and one of the excluded tuples in E_2, and as such the formula as a whole is satisfiable if and only if $R(x_1, \ldots, x_p)$ is satisfiable. The same procedure can be repeated for all the other relations in S. Moreover, since no extra variables are introduced and the number of new constraints is bounded by S, the reduction is an LV-reduction.

Let $R \in S'$ be a relation with arity p and ϕ an instance of SAT(S') with n variables. By definition, $R = \{0,1\}^p \setminus E$, where E is a set of p-ary tuples over $\{0,1\}$ that describes the excluded tuples in the relation.

Let $|E| = N$ and $e_1, \ldots, e_N \in E$ be an enumeration of its elements. Let $e_i = (b_{i,1}, \ldots, b_{i,p})$, $b_{i,j} \in \{0,1\}$.

If $R(x_1, \ldots, x_p)$ is a constraint in ϕ, then it can be expressed by the SAT(Γ^p_{SAT}) formula $\phi_1 \wedge \ldots \wedge \phi_N$, where each $\phi_i = \gamma^p_{\mathrm{SAT}}(y_1, \ldots, y_p)$, and $y_j = x_j$ if $b_{i,j}$ is 0, and $y_j = \neg x_j$ if $b_{i,j}$ is 1. Each constraint represents exactly one of the excluded tuples in R, and as such the formula as a whole is satisfiable if and only if $R(x_1, \ldots, x_p)$ is satisfiable. The same prodecure can be repeated for all the other relations in S. Moreover, since no extra variables are introduced and the number of new constraints are bounded by S', the reduction is an LV-reduction.

□

C Figures

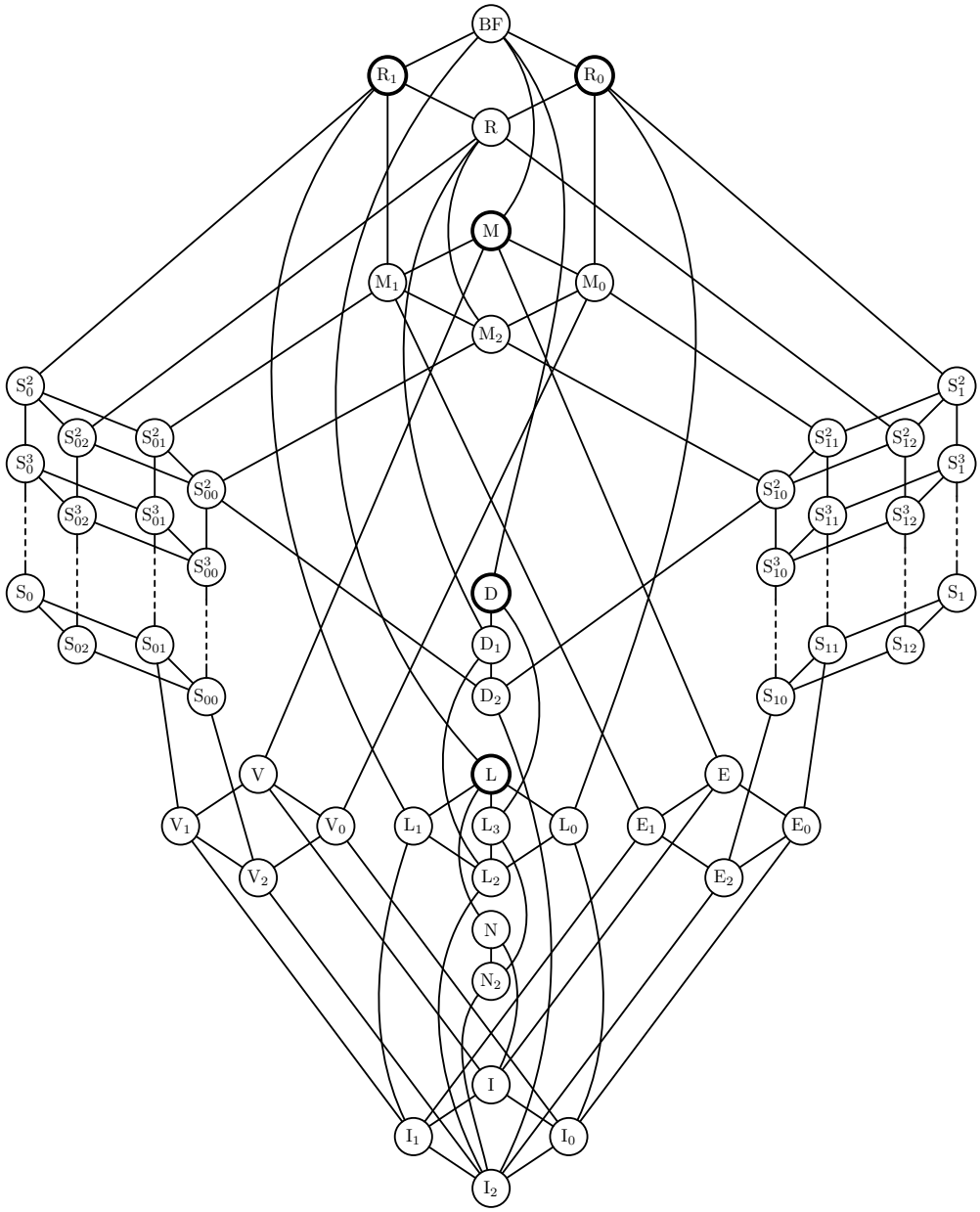

Fig. 1. The lattice of Boolean clones.

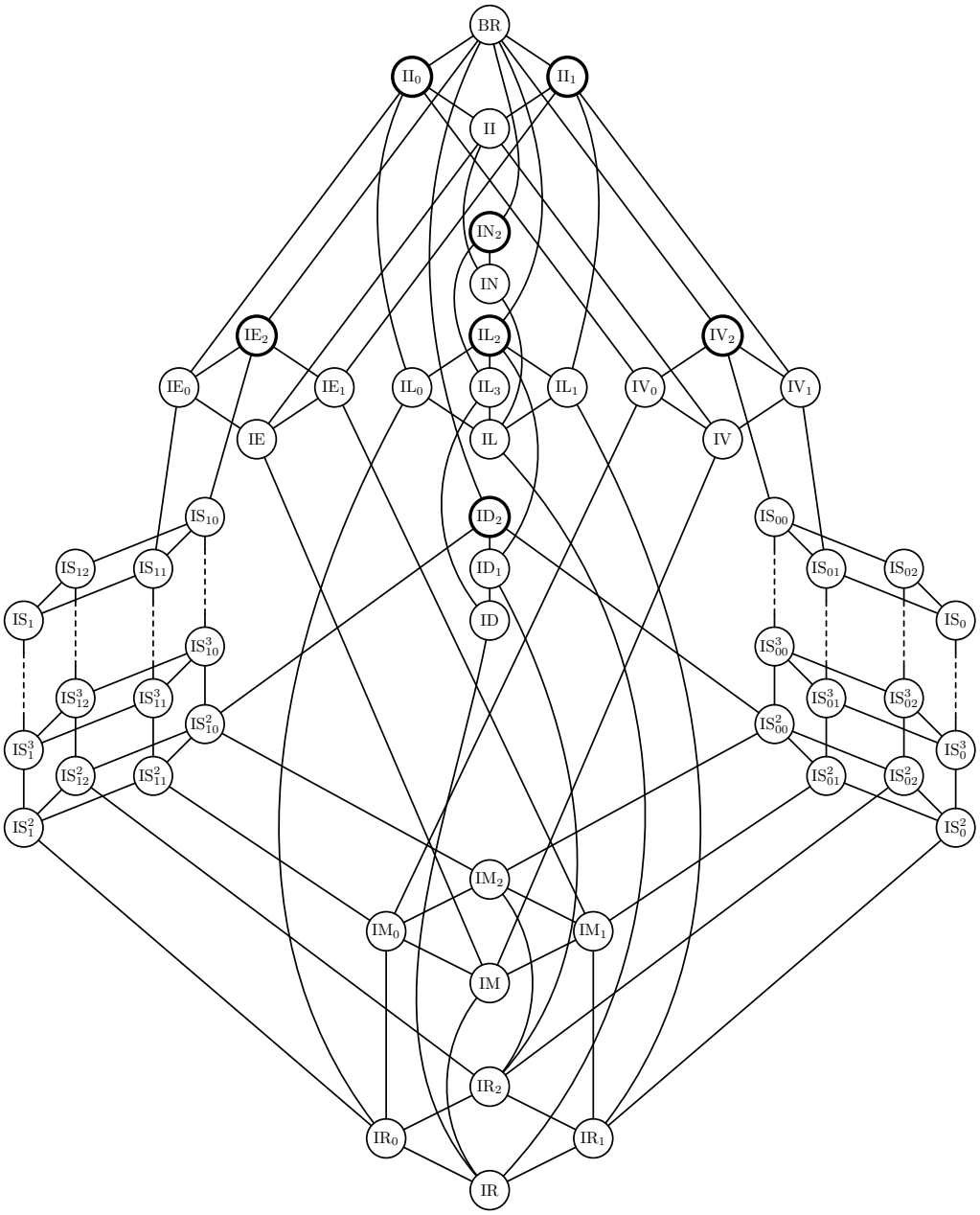

Fig. 2. The lattice of Boolean co-clones.