Strong Partial Clones and the Time Complexity of SAT Problems

Peter Jonsson[a], Victor Lagerkvist[b,*], Gustav Nordh[c], Bruno Zanuttini[d]

[a] *Department of Computer and Information Science, Linköpings Universitet, Sweden.*
[b] *Institut für Algebra, TU Dresden, Dresden, Germany*
[c] *Kvarnvägen 6, 53374, Hällekis, Sweden.*
[d] *GREYC, Normandie Université, UNICAEN, CNRS, ENSICAEN, France*

Abstract

Improving exact exponential-time algorithms for NP-complete problems is an expanding research area. Unfortunately, general methods for comparing the complexity of such problems is sorely lacking. In this article we study the complexity of SAT(S) with reductions increasing the amount of variables by a constant (CV-reductions) or a constant factor (LV-reductions). Using *clone theory* we obtain a partial order $\leq$ on languages such that SAT(S) is CV-reducible to SAT(S') if $S \leq S'$. With this ordering we identify the *computationally easiest* NP-complete SAT(S) problem (SAT($\{R\}$)), which is *strictly easier* than 1-in-3-SAT. We determine many other languages in $\leq$ and bound their complexity in relation to SAT($\{R\}$). Using LV-reductions we prove that the *exponential-time hypothesis* is false if and only if all SAT(S) problems are subexponential. This is extended to cover degree-bounded SAT(S) problems. Hence, using clone theory, we obtain a solid understanding of the complexity of SAT(S) with CV- and LV-reductions.

Keywords: satisfiability problems, computational complexity, clone theory, universal algebra, subexponential time

*Corresponding author.
Email addresses: peter.jonsson@liu.se (Peter Jonsson), victor.lagerqvist@tu-dresden.de (Victor Lagerkvist), gustav.nordh@gmail.com (Gustav Nordh), bruno.zanuttini@unicaen.fr (Bruno Zanuttini)

A preliminary version of this article appeared in *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA 2013), New Orleans, Louisiana USA.*

1. Introduction

This article is concerned with the time complexity of SAT(S) problems, *i.e.*, problems where we are given a finite set of Boolean relations S, and the objective is to decide whether a conjunction of constraints (where only relations from S are used) is satisfiable or not. We have divided this introductory section into three sections. We give a brief overview of SAT problems and describe how clone theory can be used for studying time complexity in Section 1.1. Given this approach, there are two kinds of reductions (*CV*- and *LV*-reductions) that are natural to study. We discuss these reductions and applications of them in Sections 1.2 and 1.3, respectively.

1.1. The Complexity of the Parameterized SAT($\cdot$) *Problem*

The class of SAT($\cdot$) problems is very rich and contains many problems that are highly relevant both theoretically and in practice. Since Schaefer's seminal dichotomy result [34], the computational complexity up to polynomial-time reducibility of SAT(S) is completely determined: we know for which S the problem SAT(S) is polynomial-time solvable and for which it is NP-complete, and these are the only possible cases. More recently, with a refined notion of AC^0 reductions, we have also gained a complete understanding of the SAT($\cdot$) problem for the complexity classes within P [2].

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 (where S consists of all clauses of length at most 3) is only known to be solvable in time $O(1.308^n)$ [16] (where n is the number of variables), and so it seems to be a much harder problem than, for instance, monotone 1-in-3-SAT (where S consists only of the relation $\{(0,0,1),(0,1,0),(1,0,0)\}$), which can be solved in time $O(1.0984^n)$ [42]. 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 Cygan *et al.* [9].

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

In this article, we initiate a systematic study of the relationships between the worst-case complexity of different SAT($\cdot$) problems, with respect to even more restricted reductions than AC^0 reductions. More precisely we are interested in reductions that only increase the amount of variables by a constant, *constant variable reductions*, (CV-reductions), and reductions that increase the amount of variables by a constant factor, *linear variable reductions* (LV-reductions). With these reductions it is possible to obtain a much more refined view of the seemingly large complexity differences between NP-complete SAT($\cdot$) problems.

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 but not faster. 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 (assuming P $\neq$ NP). 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 satisfying the property that if S and S' are comparable to each other in the lattice, then SAT(S) is not computationally harder than SAT(S'). To obtain such lattices, we exploit *clone theory* [25, 40]. The theory of total clones has proven to be very powerful when studying the complexity of SAT(S) and its multi-valued generalization known as *constraint satisfaction problems (CSP)* [8]. However, it is not clear how this theory can be used for studying the worst-case running times for algorithms and how to obtain CV-reductions between SAT($\cdot$) problems. We show how to use it for this purpose in Section 3, and our basic observation is that the lattice of *strong partial clones* [4, 5, 32] has the required properties. We would like to emphasize that this approach can be generalized in different ways, since it is

not restricted to Boolean problems and is applicable to other computational problems, such as counting and enumeration.

1.2. "Easy" problems and CV-reductions

As a concrete application of the clone theoretical approach and CV-reductions, we (in Section 4) identify the computationally easiest NP-complete SAT(S) problem. By "computationally easiest", we mean that if any NP-complete SAT(S) problem can be solved in $O(2^{(c+\varepsilon)n})$ time for all $\varepsilon > 0$, then so can the easiest problem. Observe that our notion of "easiest" does not rule out the existence of other constraint languages resulting in equally easy SAT($\cdot$) problems. The 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 [35] technique for constructing *weak bases* of Boolean relational clones. Obviously the first question that any astute reader would ask is exactly how easy SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is compared to other SAT($\cdot$) problems. We answer this question in Section 5 and relate the complexity of SAT($\{R_{1/3}^{\neq\neq\neq}\}$) to 1-in-3-SAT, and prove that SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is solvable in time $O(2^{(c+\varepsilon)n})$ for all $\varepsilon > 0$ if and only if 1-in-3-SAT is solvable in time $O(2^{(2c+\varepsilon)n})$ for all $\varepsilon > 0$. By 1-in-3-SAT we mean SAT(S) where S contains all ternary relations corresponding to exactly one of three literals being assigned to true, not to be confused with *monotone* 1-in-3-SAT. Hence SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is strictly easier than 1-in-3-SAT but still relatable to it within a small constant factor. Similar results are also proven for other languages that like $R_{1/3}^{\neq\neq\neq}$ contain a sufficient number of complementary arguments.

We note that there has been an interest in identifying extremely easy NP-complete problems before. For instance, van Rooij *et al.* [41] have shown that the Partition Into Triangles problem restricted to graphs of maximum degree four can be solved in $O(1.02445^n)$ time. They argue that practical algorithms may arise from this kind of studies, and the very same observation has been made by, for instance, Woeginger [43]. 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 SAT(S) problems. This is one of the major advantages in using the clone-theoretical approach when studying this kind of questions. Another reason to study such problems is that they, in some sense, are close to the borderline between problems in P 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 [22] and Schöning [39].

Having determined the easiest SAT($\cdot$) problem, it is natural to investigate other properties of the lattice of strong partial clones. We do this in Section 6 and focus on two aspects. First, we provide a partial classification of all Boolean constraint languages below monotone 1-in-3-SAT and among other prove that the relations $R_{1/3}^{\neq\neq} = \{(1,0,0,0,1), (0,1,0,1,0), (0,0,1,1,1)\}$ and $R_{1/3}^{\neq} = \{(1,0,0,0), (0,1,0,1), (0,0,1,1)\}$ reside in this structure. We conjecture that the strong partial clones corresponding to these languages *cover* each other in the sense that there are no languages of intermediate complexity in between. If this is true then SAT($\{R_{1/3}^{\neq\neq}\}$) and SAT($\{R_{1/3}^{\neq}\}$) can be regarded as the second easiest and third easiest SAT($\cdot$) problems, respectively. Combined with the results from Section 5 this also shows that all Boolean constraint languages below 1-in-3-SAT are in the worst case solvable in time $O(2^{(2c+\varepsilon)n})$ for all $\varepsilon > 0$ if the easiest problem SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is solvable in time $O(2^{(c+\varepsilon)n})$ for all $\varepsilon > 0$. Second, we show that both monotone 1-in-k-SAT and k-SAT correspond to different strong partial clones for every k and also that the strong partial clones corresponding to monotone 1-in-$(k+1)$-SAT and k-SAT are incomparable. These proofs do not require any particular complexity theoretical assumptions and may be interesting to compare with existing work on the complexity of k-SAT [17].

1.3. Subexponential complexity and LV-reductions

The second part of the paper (Section 8) is devoted to relating the complexity of SAT($\cdot$) problems to the EXPONENTIAL TIME HYPOTHESIS (ETH) [18], *i.e.*, the

hypothesis that k-SAT cannot be solved in subexponential 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.* [26].

To study the implications of the ETH for the SAT($\cdot$) problem we utilize LV-reductions instead of CV-reductions, since the former results in more powerful reductions but still preserves subexponential complexity. We let the results in the previous sections guide us by exploiting the SAT($\{R_{1/3}^{\neq\neq\neq}\}$) problem. This problem is CV-reducible (and thus trivially LV-reducible) to any NP-complete SAT(S), but the converse question of which SAT(S) problems are LV-reducible to SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is more challenging. By utilizing *sparsification* [17, 18], we can attack the more general problem of identifying degree bounded SAT(S)-DEG-B problems that are subexponential if and only if 3-SAT is subexponential. Here SAT(S)-DEG-B denotes the SAT(S) problem restricted to instances where each variable occurs in at most B constraints. We do this in Section 8.3 and prove that the exponential-time hypothesis holds if and only if either of SAT($\{R_{1/3}^{\neq\neq\neq}\}$)-DEG-2, SAT($\{R_{1/3}^{\neq\neq}\}$)-DEG-2 or SAT($\{R_{1/3}^{\neq}\}$)-DEG-2 cannot be solved in subexponential time. An important ingredient in the proof is the result (proven in Section 7) that SAT($\{R_{1/3}^{\neq\neq\neq}\}$)-DEG-2 is NP-complete. This also holds for SAT($\{R_{1/3}^{\neq\neq}\}$) and SAT($\{R_{1/3}^{\neq}\}$). We prove this by using results by Dalmau and Ford [10] combined with the fact that $R_{1/3}^{\neq\neq\neq}$, $R_{1/3}^{\neq\neq}$ and $R_{1/3}^{\neq}$ are not Δ-matroid relations. This should be contrasted with monotone 1-in-3-SAT or CNF-SAT, which are in P under the same restriction. We conclude that SAT($\{R_{1/3}^{\neq\neq\neq}\}$)-DEG-2, SAT($\{R_{1/3}^{\neq\neq}\}$)-DEG-2 and SAT($\{R_{1/3}^{\neq}\}$)-DEG-2 are all good examples of problems with extremely simple structures but which remain NP-complete.

Combining these results we show the following consequence: if ETH does not hold, then SAT(S)-DEG-B is subexponential for every B whenever S is finite. Thus, under LV-reductions, all SAT(S) problems and many SAT(S)-DEG-B problems are equally hard. Impagliazzo et al. [18] prove that many NP-complete problems in SNP (which contains the SAT($\cdot$) problem) are subexponential if and only if k-SAT is subexponential. Hence we strengthen this result when

restricted to SAT($\cdot$) problems. In the process, we also prove a stronger version of Impagliazzo *et al.*'s [18] sparsification lemma for k-SAT; namely that all finite Boolean constraint languages S and S' such that both SAT(S) and SAT(S') are NP-complete can be sparsified into each other. This can be contrasted with Santhanam's and Srinivasan's [33] negative result, which states that the same does not hold for the unrestricted SAT problem and, consequently, not for all infinite Boolean constraint languages.

2. Preliminaries

We begin by introducing the notation and basic results that will be used in the rest of this article, starting with Boolean satisfiability, followed by complexity notation and size-preserving reductions.

2.1. The Boolean SAT Problem

The set of all k-tuples over $\{0,1\}$ is denoted by $\{0,1\}^k$. A *k-ary relation* is a subset of $\{0,1\}^k$. If R is a k-ary relation then we let $\mathrm{ar}(R) = k$. The set of all finitary relations over $\{0,1\}$ is denoted by BR. A *constraint language* over $\{0,1\}$ is a finite set $S \subset \mathrm{BR}$. We insist that S is finite since this is essential for most results in the article.

Definition 1. The Boolean satisfiability problem over the constraint language $S \subset \mathrm{BR}$, denoted by 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 tuple of variables of length k_i, called the constraint scope, and R_i a k_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) or not, 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.

Typically we write $R(x_1, \ldots, x_k)$ instead of $((x_1, \ldots, x_k), R)$ to denote a constraint application of the relation R to the variables $x_1, \ldots, x_k$. Also, we

often view an instance of $\mathrm{SAT}(S)$ as a formula $\phi = R_1(\mathbf{x}_1) \wedge \ldots \wedge R_k(\mathbf{x}_k)$ where $R_1, \ldots, R_k$ are relations in S and each tuple of variables $\mathbf{x}_i$, $1 \leq i \leq k$, contains the same number of variables as the arity of R_i.

Example 1. *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 monotone Not-All-Equal 3-Sat can be expressed as $\mathrm{SAT}(\{R_{\mathrm{NAE}}\})$. Similarly, if we define the relation $R_{1/3}$ to consist of the three tuples $\{(0,0,1),(0,1,0),(1,0,0)\}$, then $\mathrm{SAT}(\{R_{1/3}\})$ corresponds to monotone 1-in-3-SAT.*

Constraint languages where negation is normally used need 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 $R_{\mathrm{NAE}}^{(+,-,-)} = \{0,1\}^3 \setminus \{(0,1,1),(1,0,0)\}$. More generally, we write Γ_{NAE}^k for the corresponding constraint language of not-all-equal relations (with all possible sign patterns) of arity k. We use the notation $\gamma_{\mathrm{NAE}}^k(\ell_1, \ldots, \ell_k)$ to denote $\mathrm{SAT}(\Gamma_{\mathrm{NAE}}^k)$ constraints, where each ℓ_i is unnegated or negated. In the same manner we write Γ_{SAT}^k for the constraint language consisting of all k-SAT relations of arity 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 relative order of the columns in the matrix representation does not matter since this only corresponds to a different order of the variables in a

constraint.

We will mainly be concerned by the time complexity of $\mathrm{SAT}(S)$ when S is finite and $\mathrm{SAT}(S)$ is NP-complete. It is thus convenient to introduce some simplifying notation: let $\mathcal{H}$ denote the set of all finite Boolean constraint languages S such that $\mathrm{SAT}(S)$ is NP-complete, and define the function $\mathsf{T} : S \to \mathbb{R}^+$ such that

$$\mathsf{T}(S) = \inf\{c \mid \mathrm{SAT}(S) \text{ can be solved in time } O(2^{c \cdot n})\}.$$

Let $c = \mathsf{T}(S)$ for some $S \in \mathcal{H}$. We see that $\mathrm{SAT}(S)$ is not necessarily solvable in time 2^{cn} but it can be solved in time $2^{(c+\varepsilon)n}$ for every $\varepsilon > 0$. If $\mathsf{T}(S) = 0$ (*i.e.*, when $\mathrm{SAT}(S)$ is solvable in time $2^{c \cdot n}$ for all $c > 0$), then we say that $\mathrm{SAT}(\mathcal{S})$ is a *subexponential* problem. The exponential-time hypothesis (ETH) is the hypothesis that k-SAT is not subexponential when $k \geq 3$ [17] or, equivalently, that $\mathsf{T}(\Gamma^k_{\mathrm{SAT}}) > 0$ whenever $k \geq 3$. Obtaining lower bounds for T is obviously difficult: for instance, $\mathsf{T}(\Gamma^3_{\mathrm{SAT}}) > 0$ implies P $\neq$ NP. However, it may be the case that P $\neq$ NP and $\mathsf{T}(\Gamma^3_{\mathrm{SAT}}) = 0$, in which case $\mathrm{SAT}(\Gamma^3_{\mathrm{SAT}})$ is a subexponential problem but not a member of P. A large number of upper bounds on T are known, though. For example we have $\mathsf{T}(\Gamma^3_{\mathrm{SAT}}) \leq \log_2(1.3334)$ [29] and $\mathsf{T}(\{R_{1/3}\}) \leq \log_2(1.0984)$ [42]. With the T function we can also define the notions of "easier than" and "strictly easier than" from the introduction in a more precise way.

Definition 2. Let $S, S' \in \mathcal{H}$. If $\mathsf{T}(S) \leq \mathsf{T}(S')$ then we say that $\mathrm{SAT}(S)$ is *easier than* $\mathrm{SAT}(S')$, and if $\mathsf{T}(S) < \mathsf{T}(S')$ then we say that $\mathrm{SAT}(S)$ is *strictly easier than* $\mathrm{SAT}(S')$.

Note that the second case can only occur if $\mathrm{SAT}(S')$ is not solvable in subexponential time.

We conclude this section with a few words about *bounded degree instances*. Let S be a constraint language and ϕ an instance of $\mathrm{SAT}(S)$. If x occurs in B constraints in ϕ, then we say that the *degree* of x is B. We let $\mathrm{SAT}(S)$-DEG-B denote the $\mathrm{SAT}(S)$ problem where each variable in the input is restricted to have

degree at most B. Similarly we let SAT(S)-OCC-B denote the SAT(S) problem where in each instance each variable can occur at most B times. The difference between the two notions is that in the latter case the total number of occurences of variables, also within constraints, are counted, while in the former only the degrees of variables are considered. For example, if $\phi = R_{1/3}(x, y, y) \wedge R_{1/3}(x, z, w)$, then x has degree 2 and y, z, w degree 1, but x and y have the same number of occurences. Obviously if SAT(S)-DEG-B is in P then SAT(S)-OCC-B is also in P, and if SAT(S)-OCC-B is NP-complete then SAT(S)-DEG-B is NP-complete. These restrictions have been studied before, and it is known that for every language S such that SAT(S) is NP-complete, there exists a B such that SAT(S)-OCC-B is NP-complete.

Theorem 3 (Jonsson *et al.* [21]). *If $S \in \mathcal{H}$, then there exists an integer B such that* SAT(S)*-OCC-B is NP-complete.*

Hence, the same also holds for SAT(S)-DEG-B.

2.2. LV-Reductions

Ordinary polynomial-time many-one reductions from SAT(S) to SAT(S') may increase the number of variables substantially—if we start with an instance ϕ of SAT(S) containing n variables, then the resulting instance ϕ' of SAT(S') will contain $p(n)$ variables for some polynomial p. This implies that polynomial-time reductions are not very useful for comparing and analyzing the precise time complexity of SAT problems. To keep the growth of the number of variables under control, we introduce *linear variable reductions*. Such reductions should be compared to the more complex but versatile class of SERF-reductions (Impagliazzo et al. [18]).

Definition 4. Let S and S' be two finite constraint languages. A total function f from SAT(S) to SAT(S') is a *many-one linear variable reduction with parameter $C \geq 0$ if for all* SAT(S) *instances ϕ with n variables:*

1. *ϕ is satisfiable if and only if $f(\phi)$ is satisfiable,*

2. *there are* $C \cdot n + O(1)$ *variables in* $f(\phi)$*, and*

3. $f(\phi)$ *can be computed in time* $O(poly(n))$*.*

We use the term *LV-reduction* as a shorter name for this kind of reduction and *constant variable reduction* (CV-reduction) to denote an LV-reduction with parameter 1. For simplicity, we choose to measure the time complexity of the reduction with respect to the number of variables instead of the size of the instance. This will make combinations of LV-reductions and sparsification (which will be used extensively in Section 8) easier to analyze, but it is not a real limitation since we consider finite constraint languages only: if an instance ϕ (over S) contains n variables, then the size of ϕ is polynomially bounded in n. To see this note that ϕ contains at most $n^k \cdot |S|$ constraints where k is the maximum arity of any relation in $\mathcal{S}$, since we have defined instances to be *sets* of constraints, and hence without repetitions. We have the following obvious but useful lemma.

Lemma 5. *Let* S *and* S' *be two finite constraint languages such that* $\mathrm{SAT}(S)$ *can be solved in time* $O(poly(n) \cdot c^n)$*, where* n *denotes the number of variables. If there exists an LV-reduction from* $\mathrm{SAT}(S')$ *to* $\mathrm{SAT}(S)$ *with parameter* C*, then* $\mathrm{SAT}(S')$ *can be solved in time* $O(poly(n) \cdot d^n)$ *where* $d = c^C$*.*

In particular, if $\mathrm{SAT}(S)$ is subexponential, then $\mathrm{SAT}(S')$ is subexponential, too. We may alternatively view this lemma in terms of the function T: if there exists an LV-reduction from $\mathrm{SAT}(S')$ to $\mathrm{SAT}(S)$ with parameter C, then $\mathsf{T}(S') \leq \mathsf{T}(S) \cdot C$ and $\mathrm{SAT}(S')$ can be solved in time $O(2^{(\mathsf{T}(S) \cdot C + \varepsilon) \cdot n})$ for every $\varepsilon > 0$. Similarly, if $\mathrm{SAT}(S')$ is CV-reducible to $\mathrm{SAT}(S)$ then $\mathrm{SAT}(S')$ is easier than $\mathrm{SAT}(S)$, *i.e.*, $\mathsf{T}(S') \leq \mathsf{T}(S)$, and if $\mathrm{SAT}(S')$ is LV-reducible to $\mathrm{SAT}(S)$ with parameter $\frac{1}{c}$ for some $c > 1$ then $\mathrm{SAT}(S')$ is strictly easier than $\mathrm{SAT}(S)$, *i.e.*, $\mathsf{T}(S') < \mathsf{T}(S)$.

3. Clones and the Complexity of SAT

We will now show that the time complexity of SAT(S) is determined by the so-called *strong partial clone* associated with S. For a more in-depth background on SAT and algebraic techniques, we refer the reader to Böhler *et al.* [6] and Lau [25], respectively. Even though most of the results in this section hold for arbitrary finite domains, we present everything in the Boolean setting since this is the focus of the article. This section is divided into two parts where we first introduce clones of total functions, continued by clones of partial functions.

3.1. Clones and Co-Clones

Any k-ary function f on $\{0,1\}$ can be extended in a standard way to function on tuples over $\{0,1\}$ as follows: let R be an l-ary Boolean relation and let $t_1, t_2, \ldots, t_k \in R$. The l-tuple $f(t_1, t_2, \ldots, t_k)$ is defined as:

$$\begin{aligned} f(t_1, t_2, \ldots, t_k) = \big(&f(t_1[1], t_2[1], \ldots, t_k[1]), \\ &f(t_1[2], t_2[2], \ldots, t_k[2]), \\ &\vdots \\ &f(t_1[l], t_2[l], \ldots, t_k[l])\big), \end{aligned}$$

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

Definition 6. Let S be a Boolean constraint language and R an arbitrary relation from S. If f is a function 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 said to be *closed* (or *invariant*) under f. If all relations in S are closed under f then S is said to be closed under f. A function 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 functions F, the set of all relations that are invariant under all functions in F is denoted by Inv(F).

Example 2. *The ternary* majority *function f over the Boolean domain is the (unique) function satisfying $f(a,a,b) = f(a,b,a) = f(b,a,a) = a$ for $a, b \in \{0,1\}$.*

Let

$$R = \{(0,0,1),(1,0,0),(0,1,1),(1,0,1)\}.$$

It is then easy to verify that for every triple of tuples, $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z} \in R$*, we have* $f(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \in R$*. For example, if* $\boldsymbol{x} = (0,0,1), \boldsymbol{y} = (0,1,1)$ *and* $\boldsymbol{z} = (1,0,1)$*, then*

$$\begin{aligned} f(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) &= \big(f(\boldsymbol{x}[1],\boldsymbol{y}[1],\boldsymbol{z}[1]), f(\boldsymbol{x}[2],\boldsymbol{y}[2],\boldsymbol{z}[2]), f(\boldsymbol{x}[3],\boldsymbol{y}[3],\boldsymbol{z}[3])\big) \\ &= \big(f(0,0,1), f(0,1,0), f(1,1,1)\big) \\ &= (0,0,1) \in R. \end{aligned}$$

We conclude that R *is invariant under* f *or, equivalently, that* f *is a polymorphism of* R*.*

In constrast, if g *is the ternary* affine *function over the Boolean domain, defined by* $g(x,y,z) = x + y + z \pmod 2$*, then*

$$\begin{aligned} g(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) &= \big(g(\boldsymbol{x}[1],\boldsymbol{y}[1],\boldsymbol{z}[1]), g(\boldsymbol{x}[2],\boldsymbol{y}[2],\boldsymbol{z}[2]), g(\boldsymbol{x}[3],\boldsymbol{y}[3],\boldsymbol{z}[3])\big) \\ &= \big(g(0,0,1), g(0,1,0), g(1,1,1)\big) \\ &= (1,1,1) \notin R \end{aligned}$$

hence g *is not a polymorphism of* R*.*

Sets of functions of the form $\mathrm{Pol}(S)$ are referred to as *clones*. The lattice (under set inclusion) of all clones over the Boolean domain was completely determined by Post [31] and it is usually referred to as *Post's lattice*. It is visualized in Figure 1. 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 $\mathrm{SAT}(S)$ is determined, up to polynomial-time reductions, by the polymorphisms of S.

Theorem 7 (Jeavons [20]). *Let* S_1 *and* S_2 *be finite non-empty sets of Boolean relations. If* $\mathrm{Pol}(S_2) \subseteq \mathrm{Pol}(S_1)$*, then* $\mathrm{SAT}(S_1)$ *is polynomial-time many-one reducible to* $\mathrm{SAT}(S_2)$*.*

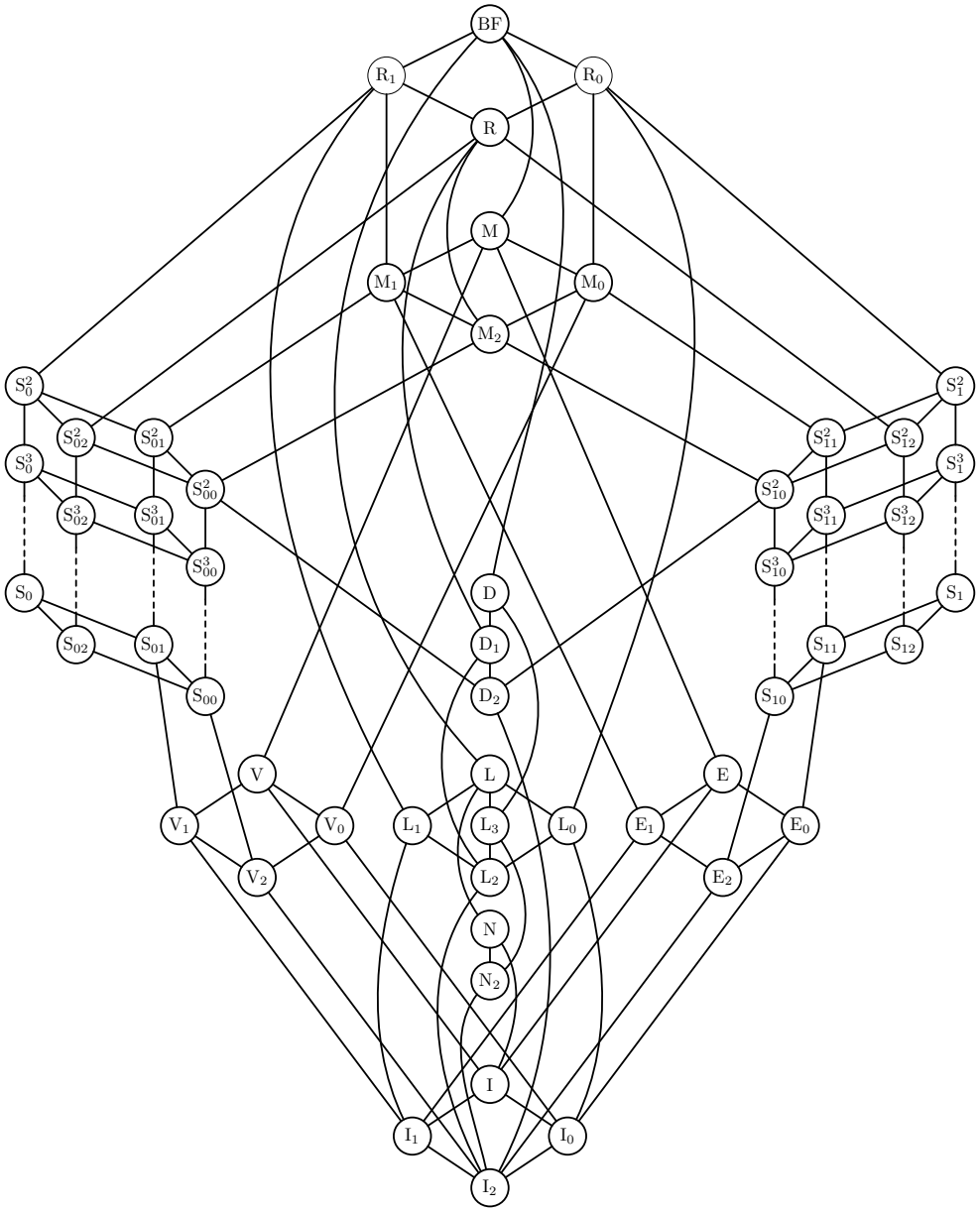

Figure 1: The lattice of Boolean clones.

Schaefer's classification of SAT(S) [34] 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 correspond to NP-complete SAT(S) problems. These are the clone I_2 consisting of all projections (*i.e.*, the functions 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 complement function $neg(0) = 1$, $neg(1) = 0$. One may note that $\mathrm{Pol}(\Gamma^k_{\mathrm{NAE}})$ is N_2, while $\mathrm{Pol}(\Gamma^k_{\mathrm{SAT}})$ is I_2 for all $k \geq 3$. If a set of relations S is invariant under the function neg, then we say that S is *closed under complement*. It is easy to see that $\mathrm{Inv}(\mathrm{I}_2)$ is the set of all Boolean relations (*i.e.*, BR) and $\mathrm{Inv}(\mathrm{N}_2)$ (which we denote IN_2) is the set of all Boolean relations that are closed under complement. More generally we use the notation I$\mathcal{C}$ for the co-clone $\mathrm{Inv}(\mathcal{C})$ (except for BR).

3.2. Strong Partial Clones

Theorem 7 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 instance sizes and may introduce large numbers of new variables. It also seems that the lattice of clones is not fine grained enough for this purpose—constraint languages that apparently have very different computational properties are assigned the very same clone, *e.g.*, both Γ^3_{SAT} and $\{R_{1/3}\}$ correspond to the clone I_2.

One way to get a more refined framework is to consider partial functions in Definition 6. By an n-ary partial function we here mean a map f from $X \subseteq \{0,1\}^n$ to $\{0,1\}$, and we say that $f(x_1, \ldots, x_n)$ is *defined* if $(x_1, \ldots, x_n) \in X$ and *undefined* otherwise. We sometimes call the set of values where the partial function is defined for the *domain* of the partial function. We then say that a relation R is closed under a partial function 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). This condition can more formally be stated as, for each sequence $t_1, \ldots, t_n \in R$, either $f(t_1, \ldots, t_n) \in R$ or there exists an i smaller than or equal to the arity of R such that $(t_1[i], \ldots, t_n[i])$ is

not included in the domain of f. The set of all partial functions preserving the relations in S, *i.e.*, the *partial polymorphisms* of S, is denoted by $\mathrm{pPol}(S)$ and is called a *strong partial clone*. It is known that strong partial clones equivalently can be defined as sets of partial functions which are closed under functional composition and closed under taking subfunctions [25]. By a subfunction of a partial function f, we here mean a partial function g whose domain is included in the domain of f, and which agrees with f for all values where it is defined. We remark that if one instead consider sets of partial functions closed under composition, but which are not necessarily closed under subfunctions, one obtains a *partial clone*. In this article we restrict ourself to strong partial clones since they can be defined via the $\mathrm{pPol}(\cdot)$ operator, which is in general not true for partial clones [37].

Example 3. *Consider again the relation R and the affine function g from Example 2 and let p be the partial function defined as $p(x,y,z) = g(x,y,z)$ except that it is undefined for $(1,1,0),(1,0,1),(0,1,1)$ and $(1,1,1)$. Now it can be verified that p is a partial polymorphism of R.*

Unlike the lattice of Boolean clones, the lattice of partial Boolean clones consists of an uncountably infinite number of partial clones [1], and despite being a well-studied mathematical object [25], its structure is far from being well-understood. For strong partial clones the situation does not differ much: the cardinality of the corresponding lattice is also known to be uncountably infinite. The proof is implicit in Alekseev and Voronenko [1] since their construction only utilizes strong partial clones.

Before we show that the lattice of strong partial clones is fine-grained enough to capture the complexity of $\mathrm{SAT}(S)$ problems, we need to present a Galois connection between sets of relations and sets of (partial) functions.

Definition 8. For any set $S \subseteq \mathrm{BR}$, 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 $S \subseteq \mathrm{BR}$ the set $\langle S \rangle_{\not\equiv}$ consists of all relations that can be expressed using relations from $S \cup \{=\}$ and conjunction. We call such implementations *quantifier-free primitive positive (q.f.p.p.) implementations*. Finally, for any set $S \subseteq \mathrm{BR}$ the set $\langle S \rangle_{\neq}$ consists of all relations that can be expressed using relations from S using conjunction and existential quantification. We call such implementations *equality-free primitive positive (e.f.p.p.) implementations*.

Example 4. *Let* $R_1 = \{(0,1),(1,0)\}$ *and* $R_2 = \{0,1\}^3 \setminus \{(0,0,0)\}$. *It is straightforward to verify that* $\Gamma^3_{\mathrm{SAT}} \subseteq \langle \{R_1, R_2\} \rangle$. *For instance, the relation* $R_3 = \{0,1\}^3 \setminus \{(1,1,1)\}$ *has the following p.p. (and e.f.p.p.) definition*

$$R_3(x,y,z) \equiv \exists x', y', z'. R_2(x', y', z') \wedge R_1(x, x') \wedge R_1(y, y') \wedge R_1(z, z').$$

One may also note that the relation $R_4 = \{(1,1,0,0),(0,0,1,1)\}$ *is a member of* $\langle R_1 \rangle_{\not\equiv}$. *Two possible q.f.p.p. definitions are*

$$R_4(x_1,x_2,x_3,x_4) \equiv R_1(x_1,x_3) \wedge R_1(x_1,x_4) \wedge R_1(x_2,x_3) \wedge R_1(x_2,x_4)$$

and

$$R_4(x_1,x_2,x_3,x_4) \equiv x_1 = x_2 \wedge x_3 = x_4 \wedge R_1(x_2,x_3).$$

The first implementation is additionally an e.f.p.p. implementation while the second is not.

Sets of relations of the form $\langle S \rangle$ and $\langle S \rangle_{\not\equiv}$ are referred to as *relational clones* (or *co-clones*) and *partial relational clones*, respectively. We note that the term partial relational clone, or co-clone, has been used in other contexts for different mathematical structures, *cf.* Chapter 20.3 in Lau [25]. For a co-clone $\langle S \rangle$ we say that S is a *base* for $\langle S \rangle$. The lattice of Boolean co-clones is visualized in Figure 2. It is easy to see that $\langle \cdot \rangle$ and $\langle \cdot \rangle_{\not\equiv}$ are closure operators, and there is a Galois connection between (partial) clones and (partial) relational clones given by the following result.

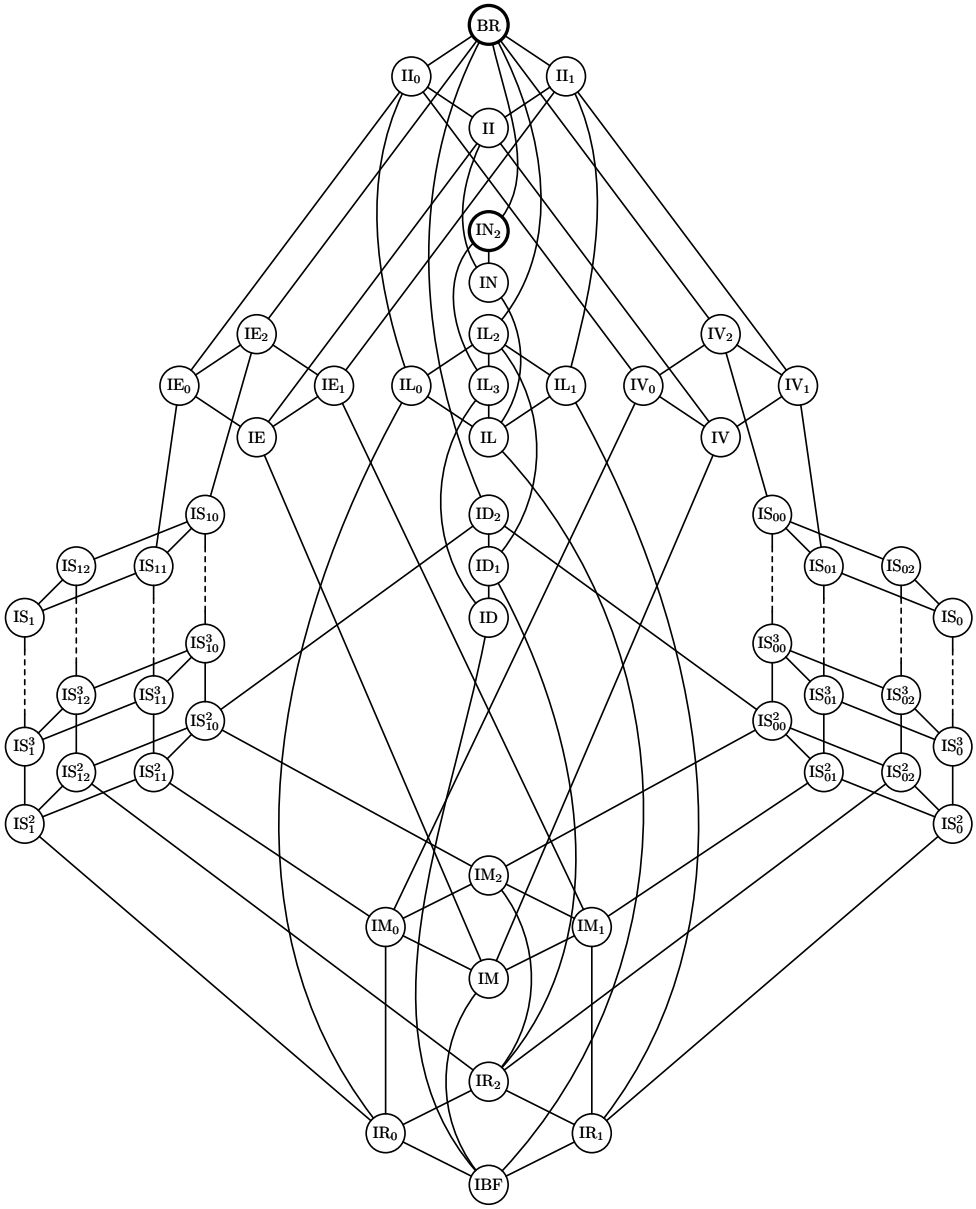

Figure 2: The lattice of Boolean co-clones. The co-clones where the SAT(·) problem is NP-hard are drawn in thick black.

Theorem 9. *[4, 5, 13, 32] Let S_1 and S_2 be constraint languages. Then $S_1 \subseteq \langle S_2 \rangle$ if and only if $\mathrm{Pol}(S_2) \subseteq \mathrm{Pol}(S_1)$, and $S_1 \subseteq \langle S_2 \rangle_{\sharp}$ if and only if $\mathrm{pPol}(S_2) \subseteq \mathrm{pPol}(S_1)$.*

We remark that there also exists a Galois connection between sets of the form $\langle S \rangle_{\neq}$ and composition-closed sets of *hyperfunctions* [7]. However, for our discourse, the above Galois connections are sufficient. We now give an analogous result to Theorem 7 which effectively shows that the complexity of $\mathrm{SAT}(S)$ is determined by the lattice of strong partial clones.

Theorem 10. *Let S_1 and S_2 be finite non-empty sets of Boolean relations. If $\mathrm{pPol}(S_2) \subseteq \mathrm{pPol}(S_1)$ then $\mathrm{SAT}(S_1)$ is CV-reducible to $\mathrm{SAT}(S_2)$.*

Proof. Given an instance ϕ of $\mathrm{SAT}(S_1)$ on n variables, we transform ϕ' (in $O(poly(n))$ time) into an equivalent instance ϕ' of $\mathrm{SAT}(S_2)$ containing at most n variables as follows. Since S_1 is fixed and finite, we can assume that the quantifier-free primitive positive implementation of each relation in S_1 by relations in S_2 has been precomputed and stored in a table (of fixed constant size). Every constraint $R(x_1, \ldots, x_k)$ in ϕ can be represented as

$$R_1(x_{11}, \ldots, x_{1k_1}) \wedge \ldots \wedge R_l(x_{l1}, \ldots, x_{lk_l})$$

where $R_1, \ldots, R_l \in S_2 \cup \{=\}$ and $x_{11}, \ldots, x_{lk_l} \in \{x_1, x_2, \ldots, x_k\}$. Replace the constraint $R(x_1, \ldots, x_k)$ with the constraints $R_1, \ldots, R_l$. If we repeat the same reduction for every constraint in ϕ, it results in an equivalent instance of $\mathrm{SAT}(S_2 \cup \{=\})$ 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 equivalent instance of $\mathrm{SAT}(S_2)$ having at most n variables. Finally, since S_1 is finite, there cannot be more than $n^p \cdot |S_1|$ constraints in I, where p is the highest arity of a relation in S_1. It follows that computing I' from I can be done in time $O(n^p)$, which concludes the proof. □

4. The Easiest NP-Complete SAT(S) 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. A crucial step of our analysis is the explicit construction by Schnoor and Schnoor [35] that, for each relational clone $\mathrm{I}\mathcal{C}$, gives a relation R such that $\mathrm{I}\mathcal{C} = \langle\{R\}\rangle$ and R has a q.f.p.p. implementation in every constraint language S such that $\langle S\rangle = \mathrm{I}\mathcal{C}$. Essentially, this construction gives the bottom element of the interval of all partial relational clones that are contained in $\mathrm{I}\mathcal{C}$ but in no other relational clone included in $\mathrm{I}\mathcal{C}$. To state Schnoor and Schnoor's result we need some additional notation. Given a natural number k the 2^k-ary relation COLS^k is the relation which contains all natural numbers from 0 to $2^k - 1$ as columns in the matrix representation, *i.e.*, each column

$$c_i = \begin{pmatrix} x_{i,1} \\ \vdots \\ x_{i,k} \end{pmatrix}$$

of COLS^k is a binary representation of i such that $x_{i,1} \dots x_{i,k} = i$. For any clone $\mathcal{C}$ and relation R we define $\mathcal{C}(R)$ to be the relation $\bigcap_{R' \in \mathrm{I}\mathcal{C}, R \subseteq R'} R'$, *i.e.*, the smallest extension of R that is preserved under every function in $\mathcal{C}$. A co-clone $\mathrm{I}\mathcal{C}$ is said to have *core-size* s if there exists relations R, R' such that $\langle R\rangle = \mathrm{I}\mathcal{C}$, $|R'| = s$ and $R = \mathcal{C}(R') = \mathrm{pPol}(R)(R')$.

Theorem 11 ([35]). *Let $\mathcal{C}$ be a clone and s be a core-size of $\mathrm{I}\mathcal{C}$. Then, for any base S of $\mathrm{I}\mathcal{C}$ it holds that $\mathcal{C}(\mathrm{COLS}^s) \in \langle S\rangle_{\not\equiv}$.*

Said otherwise, the relation $\mathcal{C}(\mathrm{COLS}^s)$ can be q.f.p.p. implemented by any base in the given co-clone. Note that $\mathcal{C}(\mathrm{COLS}^s)$ has exponential arity with respect to s. For practical considerations it is hence crucial that s is kept as small as possible. Minimal core-sizes for all Boolean co-clones have been

identified by Schnoor [36]. Moreover, Lagerkvist [23] gives a comprehensive list of relations of the form $\mathcal{C}(\mathrm{COLS}^s)$ expressed as Boolean formulas. We will use these results, but whenever needed we will recapitulate the essential steps so as to make the proofs more self-contained and easier to follow.

For any k-ary relation R we let $R^{l\neq}$, $l \leq k$, denote the $(k+l)$-ary relation defined as $R^{l\neq}(x_1, \ldots, x_{k+l}) \equiv R(x_1, \ldots, x_k) \wedge (x_1 \neq x_{k+1}) \wedge \ldots \wedge (x_l \neq x_{k+l})$. We often omit the numeral and explicitly write the number of inequalities in the relation. In the following subsections we prove that the easiest NP-complete SAT problem is $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$. In other words $R_{1/3}^{\neq\neq\neq}$ is the relation $\{001110,010101,100011\}$. Here and in the sequel we use $b_1 \cdots b_k$ as a shorthand for a tuple $(b_1, \ldots, b_k) \in \{0,1\}^k$. According to the earlier definition this relation is formed by taking $R_{1/3}$ and adding the complement to each of the columns, *i.e.*, $R_{1/3}^{\neq\neq\neq}(x_1, x_2, x_3, x_4, x_5, x_6)$ can be defined as $R_{1/3}(x_1, x_2, x_3) \wedge (x_1 \neq x_4) \wedge (x_2 \neq x_5) \wedge (x_3 \neq x_6)$. The relations $R_{1/3}^{\neq\neq}$ and $R_{1/3}^{\neq}$ are interpreted in the same way. The latter relations are related to the structure of partial co-clones in the "bottom of BR", which we expand upon in Section 6.

We are now in position to show that $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ is at least as easy as any other NP-complete $\mathrm{SAT}(\cdot)$ problem. We want to point out that there are other NP-complete constraint languages that are as easy as $\{R_{1/3}^{\neq\neq\neq}\}$. In fact, there exists an infinite number of such languages in the partial relational clone generated by $\{R_{1/3}^{\neq\neq\neq}\}$, *e.g.*, the language $\{(R_{1/3}^{\neq\neq\neq})^i\}$ for all i, with $(R_{1/3}^{\neq\neq\neq})^i(x_1, \ldots, x_{6i}) = R_{1/3}^{\neq\neq\neq}(x_1, \ldots, x_6) \wedge R_{1/3}^{\neq\neq\neq}(x_7, \ldots, x_{12}) \wedge \cdots \wedge R_{1/3}^{\neq\neq\neq}(x_{5i+1}, \ldots, x_{6i})$, but we prefer to work with $R_{1/3}^{\neq\neq\neq}$ since it is easy to describe, has reasonably low arity, and contains only three tuples.

Recall that $\mathrm{SAT}(S)$ is NP-complete if and only if $\langle S \rangle = \mathrm{BR}$ or $\langle S \rangle = \mathrm{IN}_2$. Accordingly, we first show that $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ is easier than $\mathrm{SAT}(S)$ for any S with $\langle S \rangle = \mathrm{BR}$, (Section 4.1), and then do the same for $\langle S \rangle = \mathrm{IN}_2$ (Section 4.2).

4.1. The Co-Clone BR

Let $\mathrm{I}_2 = \mathrm{Inv}(\mathrm{BR})$. Recall that I_2 is the smallest clone consisting of all projection functions. If R is a relation and $f_i^k \in \mathrm{I}_2$ is a projection function, it is

not hard to see that for all $t_1, \ldots, t_k$ it holds that $f_i^k(t_1, \ldots, t_k) = t_i$, and hence that $\mathrm{I}_2(R) = R$ for any relation R.

Lemma 12. *Let S be a constraint language such that $\langle S \rangle = \mathrm{BR}$. Then $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ is easier than $\mathrm{SAT}(S)$.*

Proof. We first construct the relation $\mathrm{I}_2(\mathrm{COLS}^s) = \mathrm{COLS}^s$, where s is a core-size of BR, and then prove that $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ is easier than $\mathrm{SAT}(\{\mathrm{COLS}^s\})$. First we see that $\mathrm{COLS}^1 = \{01\}$ and that $\langle \mathrm{COLS}^1 \rangle = \mathrm{IR}_2$. Since $\mathrm{IR}_2 \subset \mathrm{BR}$ this implies that 1 cannot be a core-size of BR. Similarly we see that 2 is not a core-size of BR since $\langle \mathrm{COLS}^2 \rangle = \langle \{0011, 0101\} \rangle = \mathrm{ID}_1 \subset \mathrm{BR}$. On the other hand it is easily seen that 3 is the minimal core-size of BR since $\mathrm{BR} = \langle R_{1/3} \rangle$, $\mathrm{I}_2(\mathrm{R}_{1/3}) = \mathrm{R}_{1/3}$ and $|R_{1/3}| = 3$. Now observe that $\mathrm{COLS}^3 = \{00001111, 00110011, 01010101\}$ is nothing else than $R_{1/3}^{\neq\neq\neq}$ with some columns rearranged and two constant columns adjoined. Hence, there is a reduction from $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ to $\mathrm{SAT}(\{\mathrm{COLS}^3\})$ where each constraint $R_{1/3}^{\neq\neq\neq}(x_1, \ldots, x_6)$ is replaced with

$$\mathrm{COLS}^3(F, x_1, x_2, x_6, x_3, x_5, x_4, T)$$

with F, T being two fresh variables shared between all constraints. Since the number of variables is augmented only by 2, this indeed shows that $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ is easier than $\mathrm{SAT}(\{\mathrm{COLS}^3\})$, which is easier than $\mathrm{SAT}(S)$ by Theorems 10 and 11. □

4.2. The Co-Clone IN_2

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 $R_{1/3}^{\neq\neq\neq}$. Let $\mathrm{N}_2 = \mathrm{Pol}(\mathrm{IN}_2)$, *i.e.*, the clone generated by the unary complement function *neg*. 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. We proceed accordingly to the derivation of $R_{1/3}^{\neq\neq\neq}$ and first determine the minimal core-size s of IN_2, calculate the relation $\mathrm{N}_2(\mathrm{COLS}^s)$, prove that $\mathrm{SAT}(\{R_{2/4}^{\neq\neq\neq\neq}\})$

is not harder than the SAT(·) problem for this relation, and last prove that SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is not harder than SAT($\{R_{2/4}^{\neq\neq\neq\neq}\}$).

Lemma 13. *Let S be a constraint language such that $\langle S\rangle = \mathrm{IN}_2$. Then the* SAT($\{R_{2/4}^{\neq\neq\neq\neq}\}$) *problem is not harder than* SAT(S).

Proof. IN_2. For $s = 1$ we see that $\langle(\mathrm{N}_2)\mathrm{COLS}^1\rangle = \langle\{01, 10\}\rangle = \mathrm{ID}$ and also for $s = 2$ that $\langle(\mathrm{N}_2)\mathrm{COLS}^2\rangle = \langle\{0011, 0101, 1100, 1010\}\rangle = \mathrm{ID}$. But $s = 3$ is indeed a core-size of IN_2 since $\langle R_{\mathrm{NAE}}\rangle = \mathrm{IN}_2$ and $R_{\mathrm{NAE}} = \mathrm{N}_2(\mathrm{R}_{1/3})$. We get the relation

$$(\mathrm{N}_2)\mathrm{COLS}^3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

by first calculating COLS^3 and then closing the resulting relation under unary complement. The relation $\mathrm{N}_2(\mathrm{COLS}^3)$ is nothing else than a rearranged version of $R_{2/4}^{\neq\neq\neq\neq}$ since every constraint $R_{2/4}^{\neq\neq\neq\neq}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$ is equivalent to $\mathrm{N}_2(\mathrm{COLS}^3)(\mathrm{x}_8, \mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_7, \mathrm{x}_3, \mathrm{x}_6, \mathrm{x}_5, \mathrm{x}_4)$. Since S can q.f.p.p. implement $\mathrm{N}_2(\mathrm{COLS})^3$ by Theorem 11 and since $\mathrm{N}_2(\mathrm{COLS}^3)$ can in turn q.f.p.p. implement $R_{2/4}^{\neq\neq\neq\neq}$, it follows that S can also q.f.p.p. implement $R_{2/4}^{\neq\neq\neq\neq}$. □

Both SAT($\{R_{1/3}^{\neq\neq\neq}\}$) and SAT($\{R_{2/4}^{\neq\neq\neq\neq}\}$) can be viewed as candidates for the easiest NP-complete SAT(S) problem. To prove that the former is not harder than the latter we have to give a size-preserving reduction from SAT($\{R_{1/3}^{\neq\neq\neq}\}$) to SAT($\{R_{2/4}^{\neq\neq\neq\neq}\}$). By using clone theory it is not hard to prove that q.f.p.p. definitions are insufficient for this purpose and that neither $R_{1/3}^{\neq\neq\neq} \in \langle R_{2/4}^{\neq\neq\neq\neq}\rangle_{\not\exists}$ nor $R_{2/4}^{\neq\neq\neq\neq} \in \langle R_{1/3}^{\neq\neq\neq}\rangle_{\not\exists}$.

Theorem 14. *$\langle R_{1/3}^{\neq\neq\neq}\rangle_{\not\exists}$ and $\langle R_{2/4}^{\neq\neq\neq\neq}\rangle_{\not\exists}$ are incomparable with respect to set inclusion.*

PROOF. We prove that neither of the partial co-clones can be a subset of the other.

For the first direction, assume towards contradiction that $\langle R_{1/3}^{\neq\neq\neq}\rangle_{\not\exists} \subseteq \langle R_{2/4}^{\neq\neq\neq\neq}\rangle_{\not\exists}$. Then $R_{1/3}^{\neq\neq\neq} \in \langle R_{2/4}^{\neq\neq\neq\neq}\rangle_{\not\exists}$, which means that there exists a q.f.p.p. implementation of $R_{1/3}^{\neq\neq\neq}$ in IN_2. This is however impossible, since this implies $\mathrm{BR} = \mathrm{IN}_2$, violating the strict inclusion structure in Post's lattice.

For the other direction, assume towards contradiction that $\langle R_{2/4}^{\neq\neq\neq\neq}\rangle_{\not\exists} \subseteq \langle R_{1/3}^{\neq\neq\neq}\rangle_{\not\exists}$. Then $\mathrm{pPol}(R_{1/3}^{\neq\neq\neq}) \subseteq \mathrm{pPol}(R_{2/4}^{\neq\neq\neq\neq})$. We show that there exists a partial function f with $f \in \mathrm{pPol}(R_{1/3}^{\neq\neq\neq})$ but $f \notin \mathrm{pPol}(R_{2/4}^{\neq\neq\neq\neq})$, and hence that the initial premise was false. Let f be the 4-ary function defined only on tuples containing two 0's and two 1's, and always returning 0. This function does not preserve $R_{2/4}^{\neq\neq\neq\neq}$ as can be seen when applying it to the tuples $(0,1,0,1,1,0,1,0), (1,0,0,1,0,1,1,0), (0,1,1,0,1,0,0,1)$ and $(1,0,1,0,0,1,0,1)$ from $R_{2/4}^{\neq\neq\neq\neq}$. But we now show that f preserves $R_{1/3}^{\neq\neq\neq}$, by showing that it is undefined on any sequence of four tuples from $R_{1/3}^{\neq\neq\neq}$. Indeed,

- taking three times the same tuple in such a sequence yields columns containing at least three 1's or three 0's,
- taking twice a tuple and twice another one always yields an all-0 and an all-1 column,
- taking two different tuples, and twice the third one yields no balanced column except for one.

Hence f preserves $R_{1/3}^{\neq\neq\neq}$ since it is always undefined when applied to any sequence of tuples from $R_{1/3}^{\neq\neq\neq}$. Therefore f is in $\mathrm{pPol}(R_{1/3}^{\neq\neq\neq}) \setminus \mathrm{pPol}(R_{2/4}^{\neq\neq\neq\neq})$, as desired. □

However, by using a relaxed notion of a q.f.p.p. definition in which a constant number of auxiliary variables are quantified over, we can still give a CV-reduction from $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ to $\mathrm{SAT}(\{R_{2/4}^{\neq\neq\neq\neq}\})$.

Lemma 15. $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ *is easier than* $\mathrm{SAT}(\{R_{2/4}^{\neq\neq\neq\neq}\})$.

PROOF. Let ϕ be an instance of SAT($\{R_{1/3}^{\neq\neq\neq}\}$) and let $C = R_{1/3}^{\neq\neq\neq}(x_1, x_2, x_3, x_4, x_5, x_6)$ 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 $Y_1 = 1$ since the complement of a satisfiable assignment is also a satisfiable assignment for constraint 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 it is indeed a CV-reduction. □

Since SAT(S) is NP-complete if and only if $\langle S\rangle = \mathrm{BR}$ or $\langle S\rangle = \mathrm{IN}_2$, Lemma 12 together with Lemma 15 gives that SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is the easiest NP-complete SAT(S) problem.

Theorem 16. *If $S \in \mathcal{H}$, then* SAT($\{R_{1/3}^{\neq\neq\neq}\}$) *is easier than* SAT($S$).

5. Complexity Bounds for SAT($\{R_{1/3}^{\neq\neq\neq}\}$) and Related Problems

As mentioned earlier, our notion of "easiest problem" does not rule out the possibility that there exist languages that have the exact same time complexity as SAT($\{R_{1/3}^{\neq\neq\neq}\}$). Proving that a problem SAT(S) is strictly easier than a problem SAT(S'), *i.e.*, that $\mathsf{T}(S') > \mathsf{T}(S)$, is of course in general much harder than giving a CV-reduction between the problems. In this section we relate the complexity between relations of the form $R^{k\neq}$, where R is k-ary, and the language Γ_R^{ext} obtained by expanding R with all sign patterns. Recall from Section 4 that $R^{l\neq}$ is the $(k+l)$-ary relation obtained from R by adding l new arguments that are the complements of the l first. In this section we mainly consider the special case $R^{k\neq}$, $\mathrm{ar}(R) = k$, which means that we add one argument i' for each argument i in the original relation R, such that $t[i'] = 1 - t[i]$ for all tuples $t \in R^{k\neq}$.

For such relations we not only prove that $T(\{R^{k\neq}\})$ is strictly smaller than $\mathsf{T}(\Gamma_R^{\mathrm{ext}})$, but that $\mathsf{T}(\Gamma_R^{\mathrm{ext}}) = 2\,\mathsf{T}(\{R^{k\neq}\})$. Said otherwise, we prove that under

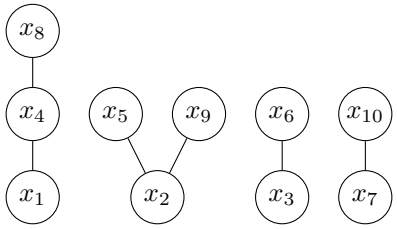

Figure 3: The complement graph $G(I)$ of the instance I in Example 5.

the ETH, $\mathrm{SAT}(\{R^{k\neq}\})$ is solvable in time $2^{(c+\epsilon)n}$ for all $\epsilon > 0$, *if and only if* $\mathrm{SAT}(\{\Gamma_R^{\mathrm{ext}}\})$ is solvable in time $2^{(2c+\epsilon)n}$ for all $\epsilon > 0$. This gives tight bounds on the complexity of relations containing a sufficient number of complementary arguments and languages containing all sign patterns.

We will in fact give a slightly more general proof. For this we need a few additional definitions. Let R be a k-ary Boolean relation, $l \leq k$, and let $R^{l\neq}(x_1, \ldots, x_{k+l})$ be a $\mathrm{SAT}(\{R^{l\neq}\})$ constraint. Say that x_i and x_{k+i}, $i \in \{1, \ldots, l\}$, are *complementary variables*. Now given an instance $I = (V, C)$ of $\mathrm{SAT}(\{R^{l\neq}\})$, we define the *complement graph* of I to be the undirected graph $G(I) = (V, E)$, with $E = \{\{x, y\} \mid x, y \text{ are complementary for some } C_i \in C\}$. In other words, the vertices of $G(I)$ are the variables of I, and two variables have an edge between them if and only if they are complementary in some constraint in C.

Example 5. *Let $I = (V, C)$ be an instance of $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ where $C = \{R_{1/3}^{\neq\neq\neq}(x_1, x_2, x_3, x_4, x_5, x_6), R_{1/3}^{\neq\neq\neq}(x_4, x_2, x_7, x_8, x_9, x_{10}))$. Then $G(I) = (V, E)$ where $E = \{\{x_1, x_4\}, \{x_4, x_8\}, \{x_2, x_5\}, \{x_3, x_6\}, \{x_2, x_9\}, \{x_7, x_{10}\}\}$. This graph is visualized in Figure 5. Note that $G(I)$ has four connected components.*

Lemma 17. *Let $R^{l\neq}$ be a $(k+l)$-ary Boolean relation, and let $\Gamma_R^{\mathrm{ext}} = \{R^{(s_1,\ldots,s_k)} \mid s_1, \ldots, s_k \in \{-, +\}\}$. If there is a constant c such that for all instances $I = (V, C)$ of $\mathrm{SAT}(\{R^{l\neq}\})$, the number of connected components of $G(I)$ is at most $\frac{|V|}{c} = \frac{n}{c}$, then $\mathrm{SAT}(\{R^{l\neq}\})$ LV-reduces to $\mathrm{SAT}(\Gamma_R^{\mathrm{ext}})$ with parameter $\frac{c}{2}$.*

Proof. Let $I = (V, C)$ be an instance of $\mathrm{SAT}(\{R^{l\neq}\})$ and let $G(I) = (V, E)$

be the corresponding complement graph. We reduce (V, C) to an equivalent instance (V', C') of $\mathrm{SAT}(\{\Gamma_R^{\mathrm{ext}}\})$ with $\frac{n}{c}$ variables as follows. First, we choose one variable of V per connected component in $G(I)$ and define V' to be the set of these. For each variable $x \in V$ we write $[x]$ for the representative of the connected component of $G(I)$ that contains x.

Observe that if there is an even-length (respectively odd-length) path between x and $[x]$ in $G(I)$, then x and $[x]$ must have the same value (respectively the opposite value) in all satisfying assignments of I. Moreover, if there is both an even-length and an odd-length path between x and $[x]$, then I must be unsatisfiable. Since this can be detected in time linear in the size of I, we henceforth assume that this is not the case.

Now let $R^{l\neq}(x_{i_1}, \ldots, x_{i_k}, x_{i_{k+1}}, \ldots, x_{i_{k+l}})$ be a constraint in C. We first replace each variable x_{i_j} with $[x_{i_j}]$ (respectively $\overline{[x_{i_j}]}$) if there is an even-length (respectively odd-length) path between x_{i_j} and $[x_{i_j}]$. We obtain an expression of the form $R^{l\neq}(\ell_{i_1}, \ldots \ell_{i_k}, \ell_{i_{k+1}}, \ldots, \ell_{i_{k+l}})$, and by construction, ℓ_{i_j} and $\ell_{i_{k+j}}$ are opposite literals for all $j = 1, \ldots, l$. Finally, from this expression we define the constraint $R^{s_1,\ldots,s_k}(x'_{i_1}, \ldots, x'_{i_k})$, with for all j, $s_j = +$ and $x'_{i_j} = x$ if ℓ_{i_j} is a positive literal x, and $s_j = -$ and $x'_{i_j} = x$ if ℓ_{i_j} is a negative literal $\bar{x}$. We define C' to be the set of all such constraints.

Clearly, the resulting formula (V', C') is an instance of $\mathrm{SAT}(\{\Gamma_R^{\mathrm{ext}}\})$, which contains at most $\frac{|V|}{c}$ variables by assumption, and which by construction is satisfiable if and only if so is (V, C). Hence this construction is an LV-reduction from $\mathrm{SAT}(\{R^{l\neq}\})$ to $\mathrm{SAT}(\Gamma^{\mathrm{ext}})$ with parameter $\frac{1}{c}$. □

For relations of the form $R^{k\neq}$, where $\mathrm{ar}(R) = k$, we get the following.

Lemma 18. *Let R be a k-ary Boolean relation. Then for all instances $I = (V, C)$ of $\mathrm{SAT}(\{R^{k\neq}\})$, either I is trivially unsatisfiable, or the number of connected components of $G(I)$ is at most $\frac{|V|}{2} = \frac{n}{2}$.*

Proof. Let $I = (V, C)$ be an instance of $\mathrm{SAT}(\{R^{k\neq}\})$. Since R has arity k, every $x \in V$ has at least one complementary variable y. If $x = y$ then x is

complementary to itself, which means that I is trivially unsatisfiable. Otherwise we get that for each variable $x \in V$ there is at least one $y \in V$, $x \neq y$, such that y occurs in the same connected component of $G(I)$. Hence each connected component of $G(I)$ contains at least 2 variables, and the result follows. □

We are now in position to give the main result of this section.

Theorem 19. *Let $R^{k\neq}$ be a 2k-ary Boolean relation and let $\Gamma_R^{\text{ext}} = \{R^{(s_1,\ldots,s_k)} \mid s_1,\ldots,s_k \in \{-,+\}\}$. Then $\mathsf{T}(\Gamma_R^{\text{ext}}) = 2\,\mathsf{T}(\{R^{k\neq}\})$.*

Proof. We prove $\mathsf{T}(\{R^{k\neq}\}) \leq \frac{\mathsf{T}(\Gamma_R^{\text{ext}})}{2}$ and $\mathsf{T}(\Gamma_R^{\text{ext}}) \leq 2\,\mathsf{T}(\{R^{k\neq}\}$. The former inequality follows from Lemma 17 and 18. For the latter inequality we give an LV-reduction from $\text{SAT}(\Gamma_R^{\text{ext}})$ to $\text{SAT}(\{R^{k\neq}\})$ with parameter 2.

Let $I = (V, C)$ be an instance of $\text{SAT}(\Gamma_R^{\text{ext}})$. For every $x_i \in V$ introduce a fresh variable x_i' and let V' be the resulting set of variables. Then, for every constraint $R^{(s_1,\ldots,s_k)}(x_{i_1},\ldots,x_{i_k})$ create a new constraint $R^{k\neq}(y_{i_1},\ldots,y_{i_{2k}})$ where

- for $i_j \leq k$, $y_{i_j} = x_{i_j}$ if $s_{j_l} = +$, and $y_{i_j} = x'_{i_j}$ if $s_{j_l} = -$, and
- for $i_j > k$, $y_{i_j} = x_{i_j-k}$ if $s_{i_j-k} = -$, and $x_{j_l} = x'_{i_j-k}$ if $s_{i_j-k} = +$.

Let C' be the set of constraints resulting from this transformation. Then (V', C') is satisfiable if and only if (V, C) is satisfiable, and since $|V'| = 2|V|$, the reduction is an LV-reduction with parameter 2, which concludes the proof. □

Let $\Gamma_{R_{1/3}}^{\text{ext}} = \{R_{1/3}^{(s_1,s_2,s_3)} \mid s_1, s_2, s_3 \in \{-,+\}\}$ be the language corresponding to 1-in-3-SAT with all possible sign patterns. As is easily verified Lemmas 17 and 18 give an LV-reduction from $\text{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ to $\text{SAT}(\Gamma_{R_{1/3}}^{\text{ext}})$ with parameter $\frac{1}{2}$. This in turn implies not only that $\text{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ is strictly easier than $\text{SAT}(\Gamma_{R_{1/3}}^{\text{ext}})$ (Lemma 5) but gives a precise bound on the difference in complexity between these two problems.

Corollary 20. $\mathsf{T}(\Gamma_{R_{1/3}}^{\text{ext}}) = 2\,\mathsf{T}(\{R_{1/3}^{\neq\neq\neq}\})$.

We can give similar bounds for $\mathrm{SAT}(\{R_{2/4}^{\neq\neq\neq\neq}\})$ and $\mathrm{SAT}(\Gamma_{2/4}^{\mathrm{ext}})$, where $\Gamma_{2/4}^{\mathrm{ext}} = \{R_{2/4}^{(s_1,s_2,s_3,s_4)} \mid s_1, s_2, s_3, s_4 \in \{-,+\}\}$, *i.e.*, $R_{2/4}^{\neq\neq\neq\neq}$ expanded with all sign patterns.

Corollary 21. $\mathsf{T}(\Gamma_{2/4}^{\mathrm{ext}}) = 2\,\mathsf{T}(\{R_{2/4}^{\neq\neq\neq\neq}\})$.

Assuming the ETH is true it is proven in [17] that there for every $k \geq 3$ exist a $k' > k$ such that k-SAT is strictly easier than k'-SAT. However, this result leaves quite a lot of gaps since it is difficult to estimate exactly how large these gaps in complexity are, and if this holds for any other languages besides k-SAT. Our results are much more precise in the sense that we for every Boolean relation of the form $R^{k\neq}$ can find a natural constraint language S such that $\mathrm{SAT}(\{R^{k\neq}\})$ is strictly easier than $\mathrm{SAT}(S)$.

6. Partial Co-Clones Covering BR

Having determined the least element COLS^3 in the partial co-clone lattice covering BR it is natural to investigate other structural properties of this lattice. More formally, given a co-clone $\mathrm{I}\mathcal{C}$, this question can be rephrased as determining the sublattice induced by the set of partial co-clones $\mathcal{I}(\mathrm{I}\mathcal{C}) = \{\mathrm{I}\mathcal{C}' \mid \mathrm{I}\mathcal{C}' = \langle \mathrm{I}\mathcal{C}' \rangle_{\not\exists}, \langle \mathrm{I}\mathcal{C}' \rangle = \mathrm{I}\mathcal{C}\}$. In the case of BR the partial co-clone $\langle \mathrm{COLS}^3 \rangle_{\not\exists}$ is the smallest element in this sublattice while the largest element is simply the set of all Boolean relations. Unfortunately we cannot hope to fully describe this sublattice since it is of uncountably infinite cardinality [38]. A more manageable strategy is to instead only consider partial co-clones that are *finitely generated*, *i.e.*, all $\mathrm{I}\mathcal{C}$ such that there exists a finite S such that $\mathrm{I}\mathcal{C} = \langle S \rangle_{\not\exists}$. Instead of the set $\mathcal{I}(\mathrm{I}\mathcal{C})$ we then consider the set $\mathcal{I}_{\mathrm{fin}}(\mathrm{I}\mathcal{C}) = \{\mathrm{I}\mathcal{C}' \mid \mathrm{I}\mathcal{C}' = \langle \mathrm{I}\mathcal{C}' \rangle_{\not\exists}, \langle \mathrm{I}\mathcal{C}' \rangle = \mathrm{I}\mathcal{C}$ and $\mathrm{I}\mathcal{C}'$ is of finite order$\}$. We note that finite constraint languages are not expressive enough to fully characterize BR or IN_2.

Theorem 22. *There is no finite S such that $\langle S \rangle_{\not\exists} = \mathrm{BR}$ or $\langle S \rangle_{\not\exists} = \mathrm{IN}_2$.*

Proof. We only sketch the proof for the case of BR. Assume there exists a finite S such that $\langle S \rangle_{\not\exists} = \mathrm{BR}$. Then $\mathrm{pPol}(S) = \mathrm{pPol}(\mathrm{BR}) = \{f \mid f$ is a subfunction

of a projection function$\}$ $= \mathrm{I}_2$. In particular, we get that $\mathrm{pPol}(S)$ can be generated from any projection function. However, this contradicts Lagerkvist & Wahlström [24, Theorem 8], where it is proven that $\mathrm{pPol}(S)$ cannot be generated from a finite set of partial functions whenever S is a finite constraint language such that $\langle S \rangle = \mathrm{BR}$. □

Despite this we believe that even partial classifications of $\mathcal{I}(\mathrm{BR})$ or $\mathcal{I}_{\mathrm{fin}}(\mathrm{BR})$ could be of interest when comparing worst-case running times of NP-hard $\mathrm{SAT}(\cdot)$ problems. In the rest of this section we therefore provide such a partial classification, and in particular concentrate on languages corresponding to k-SAT, monotone 1-in-k-SAT, and finite languages between monotone 1/3-SAT and COLS^3. To accomplish this we need to introduce some additional relations. Recall that Γ^k_{SAT} is the language consisting of all relations corresponding to k-SAT clauses.

- $R_{1/k} = \{(x_1, \ldots, x_k) \in \{0,1\}^k \mid \Sigma_{i=1}^k x_i = 1\}$,
- $\Gamma_{1/k} = \{R_{1/1}, \ldots, R_{1/k}\}$,
- $\Gamma_{\mathrm{XSAT}} = \bigcup_{i=1}^{\infty} \Gamma_{1/i}$,
- $\Gamma_{\mathrm{SAT}} = \bigcup_{i=1}^{\infty} \Gamma^i_{\mathrm{SAT}}$,
- $R^0 = \{(x_1, \ldots, x_k, 0) \mid (x_1, \ldots, x_k) \in R\}$,
- $R^1 = \{(x_1, \ldots, x_k, 1) \mid (x_1, \ldots, x_k) \in R\}$,

where R in the last two cases denotes an arbitrary k-ary Boolean relation. The results are summarized in Figure 6. That the inclusions are correct is proven in Section 6.1 and 6.2. We stress that this is indeed a partial classification. For example, for any relation R such that $\langle R^{\neq\neq\neq 01}_{1/3} \rangle_{\not\exists} \subset \langle R \rangle_{\not\exists} \subset \langle \Gamma_{1/3} \rangle_{\not\exists}$, it holds that $\langle R \rangle_{\not\exists} \subset \langle \{R, R_{1/2}\} \rangle_{\not\exists} \subset \langle \Gamma_{1/3} \rangle_{\not\exists}$ (since it is easy to prove that $R_{1/2} \notin \langle R \rangle_{\not\exists}$ and $R_{1/1} \notin \langle \{R, R_{1/2}\} \rangle_{\not\exists}$). It is also not difficult to find languages between $\langle \Gamma^k_{\mathrm{SAT}} \rangle_{\not\exists}$ and $\langle \Gamma^{k+1}_{\mathrm{SAT}} \rangle_{\not\exists}$ since for any $R \in \Gamma^{k+1}_{\mathrm{SAT}}$ it holds that $\langle \Gamma^k_{\mathrm{SAT}} \rangle_{\not\exists} \subset \langle \Gamma^k_{\mathrm{SAT}} \cup \{R\} \rangle_{\not\exists} \subset \langle \Gamma^{k+1}_{\mathrm{SAT}} \rangle_{\not\exists}$. We discuss this in more detail in Section 9.

The inclusions in Figure 6 are of particular importance when determining upper bounds on the complexities of SAT(·) problems, since $\mathsf{T}(S) \leq \mathsf{T}(S')$ if $\langle S \rangle_{\not\exists} \subseteq \langle S' \rangle_{\not\exists}$. With the help of the results from Section 5 we can in fact get tight bounds on the complexity for all languages below $\Gamma_{1/3}$.

Corollary 23. Let S be a constraint language such that $\langle S \rangle = \mathrm{BR}$ and $\langle S \rangle_{\not\exists} \subseteq \langle \Gamma_{1/3} \rangle_{\not\exists}$. Then $\mathsf{T}(\{R_{1/3}^{\neq\neq\neq}\}) \leq \mathsf{T}(\mathsf{S}) \leq 2\,\mathsf{T}(\{\mathsf{R}_{1/3}^{\neq\neq\neq}\})$.

Proof. The lower bound $\mathsf{T}(\{R_{1/3}^{\neq\neq\neq}\}) \leq \mathsf{T}(S)$ follows directly from Theorem 16. For the upper bound we note that for $\Gamma_{R_{1/3}}^{\mathrm{ext}} = \{R_{1/3}^{(s_1,s_2,s_3)} \mid s_1, s_2, s_3 \in \{-,+\}\}$ it holds that $\langle \Gamma_{R_{1/3}}^{\mathrm{ext}} \rangle_{\not\exists} \supseteq \langle \Gamma_{1/3} \rangle_{\not\exists}$ and hence that $\langle S \rangle_{\not\exists} \subseteq \langle \Gamma_{R_{1/3}}^{\mathrm{ext}} \rangle_{\not\exists}$. By applying Lemma 19 it then follows that $\mathsf{T}(S) \leq \mathsf{T}(\Gamma_{R_{1/3}}^{\mathrm{ext}}) = 2\,\mathsf{T}(\{R_{1/3}^{\neq\neq\neq}\})$. □

Hence, even if we do not currently know whether the cardinality of the set $\{\langle S \rangle_{\not\exists} \mid \langle S \rangle = \mathrm{BR}, \langle S \rangle_{\not\exists} \subseteq \langle \Gamma_{1/3} \rangle_{\not\exists}\}$ is finite or infinite, we still obtain tight complexity bounds for all these languages with respect to $\mathsf{T}(\{R_{1/3}^{\neq\neq\neq}\})$.

6.1. Partial Co-Clones Below $\langle \Gamma_{1/3} \rangle_{\not\exists}$

We begin by explicating the structure of partial co-clones $\langle S \rangle_{\not\exists}$ such that $\langle S \rangle_{\not\exists} \subseteq \langle \Gamma_{1/3} \rangle_{\not\exists}$. For this we introduce the relations $R_{1/3}^{\neq\neq\neq 01}$, $R_{1/3}^{\neq\neq 01}$, $R_{1/3}^{\neq 01}$ and $R_{1/3}^{01}$. According to the definitions in the preceding section the relations are simply $R_{1/3}^{\neq\neq\neq}$, $R_{1/3}^{\neq\neq}$, $R_{1/3}^{\neq}$ and $R_{1/3}$ with two additional constant columns adjoined. Moreover, as is easily verified, $R_{1/3}^{\neq\neq\neq 01}$ is simply COLS^3 with permuted arguments, hence $\langle R_{1/3}^{\neq\neq\neq 01} \rangle_{\not\exists}$ is equal to $\langle \mathrm{COLS}^3 \rangle_{\not\exists}$, which is the smallest element in $\mathcal{I}(\mathrm{BR})$.

Lemma 24. *The following inclusions hold.*

1. $\langle R_{1/3}^{\neq\neq\neq 01} \rangle_{\not\exists} \subset \langle R_{1/3}^{\neq\neq 01} \rangle_{\not\exists} \subset \langle R_{1/3}^{\neq 01} \rangle_{\not\exists} \subset \langle R_{1/3}^{01} \rangle_{\not\exists} \subset \langle R_{1/3} \rangle_{\not\exists} \subset \langle \Gamma_{1/3} \rangle_{\not\exists}$,
2. $\langle R_{1/3}^{01} \rangle_{\not\exists} \subset \langle R_{1/3}^{1} \rangle_{\not\exists} \subset \langle \Gamma_{1/3} \rangle_{\not\exists}$,
3. $\langle R_{1/3}^{\neq\neq\neq} \rangle_{\not\exists} \subset \langle R_{1/3}^{\neq\neq} \rangle_{\not\exists} \subset \langle R_{1/3}^{\neq} \rangle_{\not\exists} \subset \langle \Gamma_{1/3} \rangle_{\not\exists}$,
4. $\langle R_{1/3}^{\neq\neq\neq 01} \rangle_{\not\exists} \subset \langle R_{1/3}^{\neq\neq\neq} \rangle_{\not\exists}$, $\langle R_{1/3}^{\neq\neq 01} \rangle_{\not\exists} \subset \langle R_{1/3}^{\neq\neq} \rangle_{\not\exists}$, $\langle R_{1/3}^{\neq 01} \rangle_{\not\exists} \subset \langle R_{1/3}^{\neq} \rangle_{\not\exists}$,
5. $\langle R_{1/3}^{\neq\neq\neq 1} \rangle_{\not\exists} \subset \langle R_{1/3}^{\neq\neq 1} \rangle_{\not\exists} \subset \langle R_{1/3}^{\neq 1} \rangle_{\not\exists} \subset \langle \Gamma_{1/3} \rangle_{\not\exists}$,

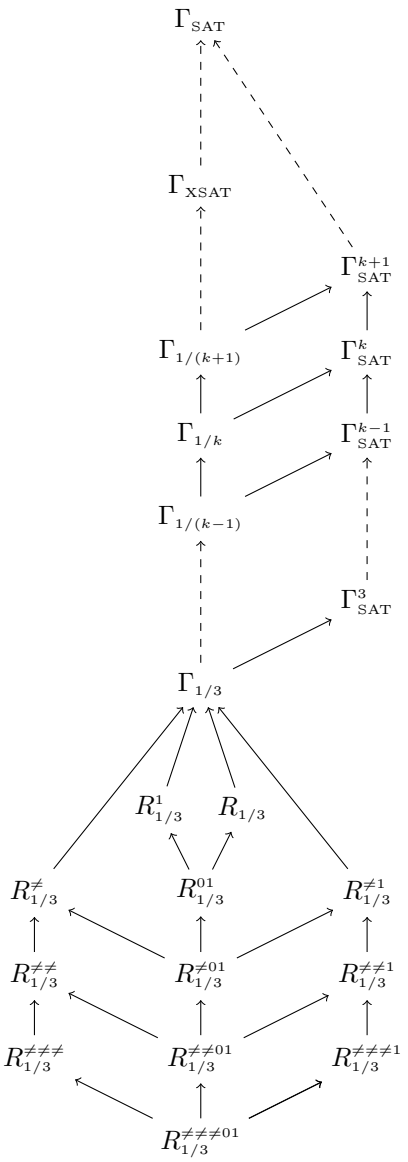

Figure 4: The structure of some partial co-clones in $\mathcal{I}(\mathrm{BR})$. A directed arrow from S to S' means $\langle S\rangle_{\not\exists} \subset \langle S'\rangle_{\not\exists}$ and hence also $\mathsf{T}(S) \leq \mathsf{T}(S')$. Some trivial inclusions, for example that every node Γ satisfies $\langle\Gamma\rangle_{\not\exists} \subseteq \langle\Gamma_{\mathrm{SAT}}\rangle_{\not\exists}$, have been omitted in the figure.

6. $\langle R_{1/3}^{\neq\neq\neq 01}\rangle_{\not\exists} \subset \langle R_{1/3}^{\neq\neq\neq 1}\rangle_{\not\exists}$, $\langle R_{1/3}^{\neq\neq 01}\rangle_{\not\exists} \subset \langle R_{1/3}^{\neq\neq 1}\rangle_{\not\exists}$, $\langle R_{1/3}^{\neq 01}\rangle_{\not\exists} \subset \langle R_{1/3}^{\neq 1}\rangle_{\not\exists}$.

Proof. We only show that the inclusions hold for the languages in (1) since the cases (2), (3), (4), (5) and (6) follow a very similar structure.

For each inclusion $\langle R_1\rangle_{\not\exists} \subseteq \langle R_2\rangle_{\not\exists}$ we prove that $R_1 \in \langle R_2\rangle_{\not\exists}$, and hence also that $\langle R_1\rangle_{\not\exists} \subseteq \langle R_2\rangle_{\not\exists}$, by giving a q.f.p.p. definition of R_1 in $\{R_2\}$. First note that $\langle R_{1/3}\rangle_{\not\exists} \subseteq \langle \Gamma_{1/3}\rangle_{\not\exists}$ since $R_{1/3} \in \Gamma_{1/3}$. We can then implement $R_{1/3}^{01}$ with $R_{1/3}$ by enforcing the constant variables c_0 and c_1 by an additional constraint, *i.e.*, $R_{1/3}^{01}(x_1, x_2, x_3, c_0, c_1) \equiv R_{1/3}(x_1, x_2, x_3) \wedge R_{1/3}(c_0, c_0, c_1)$. To implement $R_{1/3}^{\neq 01}$ with $R_{1/3}^{01}$ the procedure is similar but we also need to ensure that x_1' is assigned the opposite value of x_1, which can be done by the implementation $R_{1/3}^{\neq 01}(x_1, x_2, x_3, x_1', c_0, c_1) \equiv R_{1/3}^{01}(x_1, x_2, x_3, c_0, c_1) \wedge R_{1/3}^{01}(x_1, x_1', c_0, c_0, c_1)$. The proofs for $R_{1/3}^{\neq\neq 01}$ and $R_{1/3}^{\neq\neq\neq 01}$ are entirely analogous.

To show a proper inclusion $\langle R\rangle_{\not\exists} \subset \langle R'\rangle_{\not\exists}$ between every pair of relations R and R' we provide a partial function f which is a polymorphism of R but not of R'. Define the ternary minority function f such that it maps any 3-tuple to the value that occurs least in it, or, in the case where all three arguments are equal, to the repeating value. For example $f(0,0,1) = 1$ but $f(0,0,0) = 0$. Observe that whenever f is applied to three tuples with one or two repetitions it always yields one of these tuples. Hence, with the relations $R_{1/3}, R_{1/3}^{01}, R_{1/3}^{\neq 01}, R_{1/3}^{\neq\neq 01}, R_{1/3}^{\neq\neq\neq 01}$, we only need to consider the case when f is applied to the three distinct tuples in each relation.

$\langle R_{1/3}\rangle_{\not\exists} \subset \langle \Gamma_{1/3}\rangle_{\not\exists}$: Let $f(1) = 0$ and undefined otherwise. Then $f \notin \mathrm{pPol}(\Gamma_{1/3})$ since f does not preserve $R_{1/1} = \{(1)\}$. On the other hand f does preserve $R_{1/3}$ since it will be undefined for any $t \in R_{1/3}$.

$\langle R_{1/3}^{01}\rangle_{\not\exists} \subset \langle R_{1/3}\rangle_{\not\exists}$: Let f_1 be the partial ternary minority function which is undefined for the tuple $(0,0,0)$. Then f_1 does not preserve $R_{1/3}$ since the tuple $(f_1(0,0,1), f_1(0,1,0), f_1(1,0,0)) = (1,1,1) \notin R_{1/3}$. However, f_1 is a partial polymorphism of $R_{1/3}^{01}$ since in whatever order the tree tuples $(0,0,1,0,1)$,$(0,1,0,0,1)$ and $(1,0,0,0,1)$ from $R_{1/3}^{01}$ are taken f_1 will always be undefined for $(0,0,0)$.

$\langle R_{1/3}^{\neq 01}\rangle_{\not\exists} \subset \langle R_{1/3}^{01}\rangle_{\not\exists}$: The reasoning is similar as in the previous case except

that we define a minority function f_2 which is undefined for the tuples $(1,1,0)$, $(1,0,1)$ and $(0,1,1)$. As can be verified f_2 does not preserve $R^{01}_{1/3}$ since the tuple $(f_1(0,0,1), f_1(0,1,0), f_1(1,0,0), f_1(0,0,0), f_1(1,1,1)) = (1,1,1,0,1) \notin R^{01}_{1/3}$. It is however a partial polymorphism of $R^{\neq 01}_{1/3}$ since, regardless of the order in which the tuples are taken in, it will be undefined for the fourth column which will always be one of $(1,1,0)$, $(1,0,1)$ or $(0,1,1)$.

$\langle R^{\neq\neq 01}_{1/3}\rangle_{\not\exists} \subset \langle R^{\neq 01}_{1/3}\rangle_{\not\exists}$: Define f_3 to be the ternary minority function except that it is undefined for $(1,0,1)$ and $(0,1,1)$. Then f_3 does not preserve $R^{\neq 01}_{1/3}$ since $(f_3(0,0,1), f_3(0,1,0), f_3(1,0,0), f_3(1,1,0), f_3(0,0,0), f_3(1,1,1)) = (1,1,1,0,0,1) \notin R^{\neq 01}_{1/3}$. It will however always be undefined for the fourth or fifth column of $R^{\neq\neq 01}_{1/3}$ since one of these will be either $(1,0,1)$ or $(0,1,1)$.

$\langle R^{\neq\neq\neq 01}_{1/3}\rangle_{\not\exists} \subset \langle R^{\neq\neq 01}_{1/3}\rangle_{\not\exists}$: The reasoning is the same as in the previous case but with a minority function f_4 undefined for $(0,1,1)$. □

Observe that in the second part of the proof we exploit the Galois connection between strong partial clones and partial co-clones. This results in much more concise proofs since proving a strict inclusion $\langle R_1\rangle_{\not\exists} \subset \langle R_2\rangle_{\not\exists}$ with relational tools is tantamount to the proof that it is impossible to find a q.f.p.p. definition of R_2 in $\langle R_1\rangle_{\not\exists}$.

The inclusions in Lemma 24 do not rule out the possibility that some of the partial co-clones in Figure 6 collapse to a single partial co-clone. To rule out this we also need to prove that every pair of partial co-clones $\langle S\rangle_{\not\exists}$ and $\langle S'\rangle_{\not\exists}$ that are not connected in Figure 6 are in fact incomparable with respect to set inclusion. Here we only sketch the proof that $\langle R^{\neq}_{1/3}\rangle_{\not\exists}$ and $\langle R_{1/3}\rangle_{\not\exists}$ are incomparable since the other cases are similar. First, define the ternary function f such that $f(0,0,1) = 0, f(0,1,0) = 1, f(1,0,0) = 0, f(0,1,1) = 0$, and let it be undefined otherwise. Then $f \in \mathrm{pPol}(R_{1/3})$ (since f applied to tuples from $R_{1/3}$ always yields $(0,0,1), (0,1,0), (1,0,0) \in R_{1/3}$), but $f \notin \mathrm{pPol}(R^{\neq}_{1/3})$ since $(f(0,0,1), f(0,1,0), f(1,0,0), (0,1,1)) = (0,1,0,0) \notin R^{\neq}_{1/3}$. For the other direction define the ternary function g such that $g(0,0,1) = g(0,1,0) = g(1,0,0) = 1$ but is undefined otherwise. A quick check shows that $g \in \mathrm{pPol}(R^{\neq}_{1/3})$ (since g is

always undefined for these tuples) but $g \notin \mathrm{pPol}(R_{1/3})$.

6.2. Partial Co-Clones Above $\langle\Gamma_{1/3}\rangle_{\not\exists}$

In this section we focus on 1-in-k-SAT and k-SAT and prove that both these languages form infinitely ascending chains of partial relational clones. We stress that these results hold independently of any complexity theoretical assumptions — the k-SAT result was, *e.g.*, previously only known to hold if the ETH is true. This may be seen as further evidence that the ETH is plausible. However, note that this does not contradict the possibility that the ETH is false, since the fact that $\langle\Gamma^k_{\mathrm{SAT}}\rangle_{\not\exists} \subset \langle\Gamma^{k+1}_{\mathrm{SAT}}\rangle_{\not\exists}$ does not necessarily imply that the running times of k-SAT and $(k+1)$-SAT differs.

In the following proofs we let $\mathbf{1^k}$ and $\mathbf{0^k}$ denote k-ary tuples of ones and zeroes, respectively, *i.e.*,

$$\mathbf{1^k} = (\underbrace{1,\ldots,1}_{\mathbf{k}}), \text{and } \mathbf{0^k} = (\underbrace{0,\ldots,0}_{\mathbf{k}}).$$

Theorem 25. *$\langle\Gamma^k_{\mathrm{SAT}}\rangle_{\not\exists} \subset \langle\Gamma^{k+1}_{\mathrm{SAT}}\rangle_{\not\exists}$ for each $k \geq 1$.*

Proof. For $k = 1$ and $k = 2$ the result follows immediately from Post's lattice. Assume $k \geq 3$. By definition the language Γ^k_{SAT} contain all relations representable by the formulas $(x_1 \vee \ldots \vee x_k)$, $(\neg x_1 \vee \ldots \vee x_k)$, …, $(\neg x_1 \vee \ldots \vee \neg x_k)$, for all possible sign patterns. For simplicity we denote relations by their defining formulas and simply write $(l_1 \vee \ldots \vee l_k)$, where l_i is a literal, for a relation in Γ^k_{SAT}. For example $(x_1 \vee \ldots \vee x_k)$ is the relation $\{0,1\}^k \setminus \{\mathbf{0^k}\}$, and $(\neg x_1 \vee \ldots \vee \neg x_k)$ is the relation $\{0,1\}^k \setminus \{\mathbf{1^k}\}$.

Without loss of generality we also assume that $\langle\Gamma^k_{\mathrm{SAT}}\rangle_{\not\exists}$ is generated by the $k+1$ relations $(x_1 \vee \ldots \vee x_k)$, $(\neg x_1 \vee \ldots \vee x_k)$, …, $(\neg x_1 \vee \ldots \vee \neg x_k)$, since any other sign pattern can be represented as a permutation of these.

It is easy to prove that $\langle\Gamma^k_{\mathrm{SAT}}\rangle_{\not\exists} \subseteq \langle\Gamma^{k+1}_{\mathrm{SAT}}\rangle_{\not\exists}$ by giving explicit q.f.p.p. definitions of Γ^k_{SAT} in $\langle\Gamma^{k+1}_{\mathrm{SAT}}\rangle_{\not\exists}$, *e.g.*, $(x_1 \vee \ldots \vee x_k) \equiv (x_1 \vee \ldots x_{k-1} \vee x_k \vee x_k)$.

For the strict inclusion we give a function f such that $f \in \mathrm{pPol}(\Gamma^k_{\mathrm{SAT}})$ but $f \notin \mathrm{pPol}(\Gamma^{k+1}_{\mathrm{SAT}})$, thus proving that $\mathrm{pPol}(\Gamma^{k+1}_{\mathrm{SAT}}) \subset \mathrm{pPol}(\Gamma^k_{\mathrm{SAT}})$ and therefore that

$\langle \Gamma^k_{\mathrm{SAT}} \rangle_{\not\exists} \subset \langle \Gamma^{k+1}_{\mathrm{SAT}} \rangle_{\not\exists}$. For each $k \geq 3$ let the k-ary function f_k be defined as $f^k(x_1, \ldots, x_k) = 0$ if $\Sigma_{i=1}^k x_i = 1$, and undefined otherwise. We prove that for every $k \geq 3$ it holds that $f^{k+1} \in \mathrm{pPol}(\Gamma^k_{\mathrm{SAT}})$ but $f^{k+1} \notin \mathrm{pPol}(\Gamma^{k+1}_{\mathrm{SAT}})$. By definition f_k preserves the negative clause $(\neg x_1 \vee \ldots \vee \neg x_k)$ since it is either undefined or returns 0. Now let $R_k \in \Gamma^k_{\mathrm{SAT}}$ be the relation corresponding to the clause $(\neg x_1 \vee \ldots \vee \neg x_i \vee x_{i+1} \vee \ldots \vee x_k)$, $i \geq 1$. If f^{k+1} does not preserve R_k it, when applied componentwise to the tuples of R_k, must be able to return a tuple not included in R_k. However, since R_k contains all k-tuples except for

$$(\underbrace{1, \ldots, 1}_{\text{i}}, \underbrace{0, \ldots, 0}_{\text{k - i}}),$$

f^{k+1} must preserve R_k since f^{k+1} never returns the value 1. Now let R'_k be the relation corresponding to the clause $(x_1 \vee \ldots \vee x_k)$. If f^{k+1} is applied componentwise to $k+1$ tuples the only case in which it can return a tuple not included in R'_k is if returns the all-zero tuple $\mathbf{0^k}$. Since f^{k+1} is only defined when exactly one argument is equal to 1, the only possibility for obtaining this is if the first k tuples $t_1, \ldots, t_k$ are of the form

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

But if the $(k+1)$-th tuple is anything else than $\mathbf{0^k}$, which is not included in R'_k, there will be at least one column that contains two ones. Hence f^{k+1} will be undefined for this sequence of arguments and therefore trivially preserves R'_k.

Last, let R_{k+1} be the relation corresponding to the positive clause $(x_1 \vee \ldots \vee x_{k+1})$. This relation is included in $\Gamma^{k+1}_{\mathrm{SAT}}$ but not in Γ^k_{SAT}. Let $t_1, \ldots t_{k+1}$ be the tuples

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

Then we see that for each $i \leq k{+}1$ it holds that $f^{k+1}(t_1[i], t_2[i], \ldots, t_k[i], t_{k+1}[i]) = 0$. Hence $f^{k+1}(t_1, \ldots t_{k+1}) = \mathbf{0^{k+1}} \notin R_{k+1}$ from which it follows that f^{k+1} does not preserve $\Gamma^{k+1}_{\text{SAT}}$. □

Next we prove that a similar separation result can be obtained for the sequence of $\Gamma_{1/k}$ languages.

Theorem 26. *$\langle \Gamma_{1/k} \rangle_{\not\exists} \subset \langle \Gamma_{1/(k+1)} \rangle_{\not\exists}$ for each $k \geq 1$.*

Proof. Clearly $\langle \Gamma_{1/k} \rangle_{\not\exists} \subseteq \langle \Gamma_{1/(k+1)} \rangle_{\not\exists}$ since $\Gamma_{1/k} \subseteq \Gamma_{1/(k+1)}$. For the proper inclusion let the k-ary partial function g^k be defined as $g^k(x_1, \ldots, x_k) = 0$ if $\Sigma_{i=1}^{k} x_i = 1$ and $g^k(x_1, \ldots, x_k) = 1$ if $(x_1, \ldots, x_k) = \mathbf{0^k}$. The proof consists of two parts: first, we prove that $g^{k+1} \in \text{pPol}(\Gamma_{1/k})$ for every $k \geq 1$; second, we prove that $g^k \notin \text{pPol}(\Gamma_{1/k})$ for every $k \geq 1$.

Let $R_{1/k'} \in \Gamma_{1/k}$, $k' \leq k$. Note that $|R_{1/k'}| = k' < k{+}1$. For every application $g^{k+1}(t_1, \ldots, t_{k+1})$ where $t_1, \ldots, t_{k+1} \in R_{1/k'}$ it therefore follows that at least two tuples are repeated, and hence that there exists an $1 \leq i \leq k'$ such that $t_1[i] + \ldots + t_{k+1}[i] > 1$. By definition g^{k+1} is not defined for this sequence of arguments from which it follows that $g^{k+1}(t_1, \ldots, t_{k+1})$ is undefined, and hence that g^{k+1} preserves $R_{1/k'}$. Since $R_{1/k'}$ was chosen arbitrarily it then follows that g^{k+1} preserves $\Gamma_{1/k}$.

Next let $k \geq 1$. We prove that $g^k \notin \text{pPol}(\Gamma_{1/k})$. To see this simply note that $f(t_1, \ldots, t_k)$, where $t_1, \ldots, t_k \in R_{1/k}$ and $t_i \neq t_j$ for all $i, j \in [1, k]$, is defined and returns the all zero tuple, which is not included in $R_{1/k}$. Hence $g^k \notin \text{pPol}(\Gamma_{1/k})$.

Combining these two facts yields that $g^{k+1} \in \text{pPol}(\Gamma_{1/k})$ but $g^{k+1} \notin \text{pPol}(\Gamma_{1/(k+1)})$. Hence $\text{pPol}(\Gamma_{1/k}) \supset \text{pPol}(\Gamma_{1/(k+1)})$, which implies that $\langle \Gamma_{1/k} \rangle_{\not\exists} \subset \langle \Gamma_{1/(k+1)} \rangle_{\not\exists}$. □

It is easy to see that $\langle\Gamma_{1/k}\rangle_{\exists\neq} \subset \langle\Gamma^k_{\text{SAT}}\rangle_{\exists\neq}$. To rule out the possibility that $\langle\Gamma_{1/(k+1)}\rangle_{\exists\neq} \subseteq \langle\Gamma^k_{\text{SAT}}\rangle_{\exists\neq}$ or $\langle\Gamma^k_{\text{SAT}}\rangle_{\exists\neq} \subseteq \langle\Gamma_{1/k}\rangle_{\exists\neq}$ we prove that these are in fact incomparable.

Theorem 27. *$\langle\Gamma_{1/(k+1)}\rangle_{\exists\neq}$ and $\langle\Gamma^k_{\text{SAT}}\rangle_{\exists\neq}$ are incomparable for each $k \geq 3$.*

Proof. We must prove that there exists partial functions f, g such that $f \in \text{pPol}(\Gamma^k_{\text{SAT}})$, $f \notin \text{pPol}(\Gamma_{1/(k+1)})$ and $g \in \text{pPol}(\Gamma_{1/(k+1)})$ but $g \notin \text{pPol}(\Gamma^k_{\text{SAT}})$. For this let f^k and g^k be defined as in the proofs of Theorems 25 and 26 and recall that $f^{k+1} \in \text{pPol}(\Gamma^k_{\text{SAT}})$ and $g^{k+1} \in \text{pPol}(\Gamma_{1/k})$.

Let $k \geq 3$. It is easy to see that $g^{k+2} \notin \text{pPol}(\Gamma^k_{\text{SAT}})$ since g^{k+2} does not preserve the negative clause $(\neg x_1 \vee \ldots \vee \neg x_k)$ when applied to $k+2$ tuples $t_1 = \ldots = \ldots = t_{k+2} = \mathbf{0^k}$. Similarly f^{k+1} does not preserve $R_{1/k+1} \in \Gamma_{1/(k+1)}$ since $f(t_1, \ldots, t_{k+1}) = \mathbf{0^{k+1}} \notin R_{1/k+1}$ when all $t_1, \ldots, t_{k+1} \in R_{1/k+1}$ are distinct. □

It is also not hard to prove that $\langle\Gamma_{\text{XSAT}}\rangle_{\exists\neq}$ and $\langle\Gamma^k_{\text{SAT}}\rangle_{\exists\neq}$ are incomparable for each $k \geq 3$. The direction $\langle\Gamma^k_{\text{SAT}}\rangle_{\exists\neq} \not\subseteq \langle\Gamma_{\text{XSAT}}\rangle_{\exists\neq}$ follows from Theorem 27. For the direction $\langle\Gamma_{\text{XSAT}}\rangle_{\exists\neq} \not\subseteq \langle\Gamma^k_{\text{SAT}}\rangle_{\exists\neq}$ one can, *e.g.*, use the partial function $f(0,1) = f(1,0) = 0$, and which is undefined otherwise, since $f \in \text{pPol}(\Gamma_{\text{XSAT}})$ but $f \notin \text{pPol}(\Gamma^k_{\text{SAT}})$.

7. LV-Reductions and Bounded-Degree Instances

In this section we explicate a relationship between LV-reductions and degree-bounded SAT(S)-DEG-B problems. In particular, as will be made clear in Section 8 when we relate SAT($\{R^{\neq\neq\neq}_{1/3}\}$) to the ETH, this relationship is very useful when proving that two problems are equally hard with respect to subexponential complexity. We also tackle the problem of determining the smallest B such that SAT(S)-DEG-B is NP-complete and provide these bounds for $R^{\neq\neq\neq}_{1/3}$, $R^{\neq\neq}_{1/3}$, $R^{\neq}_{1/3}$ and $R_{1/3}$.

Recall that a relation R can be e.f.p.p. implemented from a set of relations S if it can be expressed by only using the relations in S, conjunction, and existential quantification.

Lemma 28. *Let S and S' be two finite constraint languages. If $S' \subseteq \langle S \rangle$ and the equality relation Eq can be e.f.p.p. defined in S, then* SAT(S')*-DEG-B is LV-reducible to* SAT(S)*-DEG-C where C only depends on S, S', and B. The parameter of the LV-reduction depends only on S, S', and B.*

Proof. Let ϕ be a SAT(S')-DEG-B-instance with n variables. Since each variable can occur in at most B constraints, there cannot be more than $n \cdot B$ constraints in total. Each such constraint is of the form $R(x_1, \ldots, x_k)$ where $R \in S'$. Since $S' \subseteq \langle S \rangle$, the relation R can then be expressed as a conjunction of constraints over $S \cup \{=\}$ with a set of existentially quantified variables: $\exists y_1, \ldots, y_l . \bigwedge_{i=1}^{m} \psi_i(y_{i_1}, \ldots, y_{i_{\mathrm{ar}(\psi_i)}})$, where each $\psi_i \in S \cup \{=\}$ and $\{y_{i_1}, \ldots, y_{i_{\mathrm{ar}(\psi_i)}}\} \subseteq \{x_1, \ldots, x_k\} \cup \{y_1, \ldots, y_l\}$. Furthermore, each $=$-relation can be e.f.p.p. defined in S so we may without loss of generality assume that $=$ is not used in the definition.

Hence the number of extra variables for each constraint depends only on the relations in 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 $(n \cdot B)t$, which is linear with respect to n since B and t are fixed values.

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

The condition that the equality relation *Eq* must be e.f.p.p. definable in S can be replaced by other conditions such as $S' \subseteq \langle S \rangle_{\neq}$. We have made this particular choice since, due to the next lemma, it is very convenient when working with members of $\mathcal{H}$ (recall that $\mathcal{H}$ denotes the set of finite constraint languages resulting in NP-complete SAT($\cdot$) problems).

Lemma 29. *If $S \in \mathcal{H}$, then Eq is e.f.p.p. implementable in S.*

Proof. Let *Neq* denote the disequality relation $\{(0,1),(1,0)\}$. Since *Neq* is closed under complement, it follows that *Neq* can always be p.p. defined in S (since $\langle S \rangle = \mathrm{BR}$ or $\langle S \rangle = \mathrm{IN}_2$). Let $\mathit{Neq}(x,y) \equiv \exists z_1, \ldots, z_n . \tau(x, y, z_1, \ldots, z_n)$

denote the p.p. implementation with the least number of occurrences of "=". Assume there exists an equality $v = w$ in τ. Clearly, this cannot be of the type $x = y$ since we are implementing $Neq(x, y)$. We can thus identify the variable v with w and still have an p.p. implementation of Neq — this violates the minimality condition. It follows that the implementation contains no occurrences of "=", *i.e.*, it is an e.f.p.p. implementation. We can now e.f.p.p. implement Eq as follows since we are working over a two-element domain:

$$Eq(x, y) \equiv \exists u.Neq(x, u) \wedge Neq(u, y)$$

or equivalently the implementation

$$Eq(x, y) \equiv \exists u, z_1, \ldots, z_n, w_1, \ldots, w_n.\tau(x, u, z_1, \ldots, z_n) \wedge \tau(u, y, w_1, \ldots, w_n),$$

which is clearly an e.f.p.p. implementation. □

We get the following result by combining Lemmas 28 and 29.

Corollary 30. Let $S, S' \in \mathcal{H}$ such that $S' \subseteq \langle S \rangle$ and let B be an integer. Then, there exists an LV-reduction from SAT(S')-DEG-B to SAT(S)-DEG-C where C only depends on S, S' and B.

In contrast to Lemma 35, Corollary 30 shows that there exists an LV-reduction from SAT(S)-DEG-B to SAT($\{R_{1/3}^{\neq\neq\neq}\}$) for every $S \in \mathcal{H}$ and arbitrary B. This apparently simple observation will be important in the next section where we study connections between SAT($\cdot$) problems and the exponential-time hypothesis. Corollary 30 also suggests that for languages $S \in \mathcal{H}$ it is useful to know the largest B such that SAT(S)-DEG-B is in P and the smallest C such that SAT(S)-DEG-C is NP-complete. For SAT($\{R_{1/3}\}$)-OCC-B this value is already known since for $B = 2$ it can be reduced to the problem of finding a perfect matching in a graph [19], but for $B = 3$ it is NP-complete even for planar instances [28]. Unsurprisingly the same also holds for SAT($\{R_{1/3}\}$)-DEG-2 as made clear in the following theorem.

Theorem 31. SAT($\{R_{1/3}\}$)*-DEG-2 is solvable in polynomial time.*

PROOF. We show that SAT($\{R_{1/3}\}$)-DEG-2 is polynomial-time reducible to SAT($\{R_{1/3}\}$)-OCC-2. Let $I = (V, C)$ be an instance of SAT($\{R_{1/3}\}$)-DEG-2. We transform I into an instance I' of SAT($\{R_{1/3}\}$)-OCC-2 which is satisfiable if and only if I is satisfiable.

For this we consider each variable that occurs at least twice in some constraint and at least one more time in another. Because I is an instance of SAT($\{R_{1/3}\}$)-DEG-2, such a variable cannot occur in a third constraint; moreover, if a variable occurs only once in two different constraints, then it already has at most two occurrences. Observe that if a variable occurs three times in a constraint then the instance is trivially unsatisfiable, and since this can be detected in time linear in the size of I, we assume that this is not the case.

So let $C_i \in C$ be a constraint of the form $R_{1/3}(x_i, x_i, y_i)$, let $C_j \in C$ be a constraint where x_i occurs in, and let C_k be a constraint where y_i occurs in. Since $R_{1/3}$ is a symmetric relation in the sense that $R_{1/3}(x, y, z) = R_{1/3}(x, z, y) = R_{1/3}(y, x, z) = R_{1/3}(y, z, x) = R_{1/3}(z, x, y) = R_{1/3}(z, y, x)$ we may without loss of generality assume that x_i occurs in the first position of C_j and that y_i occurs in the first position of C_k. We perform a case study depending on C_j and C_k (if it exists).

First observe that for any assignment f satisfying I it holds $f(x_i) = 0$ and $f(y_i) = 1$. Up to permutation of arguments there are a now a few different cases to consider:

1. $C_j = R_{1/3}(x_i, x_j, y_i)$ (and hence, C_k does not exist),
2. $C_j = R_{1/3}(x_i, x_i, x_j)$, $C_k = R_{1/3}(y_i, x_k, x_k)$,
3. $C_j = R_{1/3}(x_i, x_i, x_j)$, $C_k = R_{1/3}(y_i, x_k, x'_k)$,
4. $C_j = R_{1/3}(x_i, x_j, x'_j)$, $C_k = R_{1/3}(y_i, x_j, x_j)$,
5. $C_j = R_{1/3}(x_i, x_j, x'_j)$, $C_k = R_{1/3}(y_i, x_j, x_k)$,
6. $C_j = R_{1/3}(x_i, x_j, x'_j)$, $C_k = R_{1/3}(y_i, x_k, x_k)$,
7. $C_j = R_{1/3}(x_i, x_j, x'_j)$, $C_k = R_{1/3}(y_i, x_k, x'_k)$,

where x_j, x'_j, x_k, x'_k denote variables all distinct from each other and from x_i and y_i. We have not included the cases that (1) trivially result in unsatisfiable instances or (2) is simply a permutation of case (1) – (7). For example $C_j = R_{1/3}(x_i, y_i, y_i)$ is unsatisfiable since $f(y_i) = 1$ in any satisfying assignment f, and if $C_j = R_{1/3}(y_i, x_i, x_i)$ then this is simply a permutation of $R_{1/3}(x_i, x_i, y_i)$ and can safely be removed.

In each case it is easy to show how the constraints can be replaced by valid SAT($\{R_{1/3}\}$)-OCC-2 constraints. In case (1) we introduce a fresh variable x'_j and replace $R_{1/3}(x_i, x_j, y_i)$ with $R_{1/3}(x'_j, x_j, y_i)$. This ensures that x_i and y_i both occur only two times. In case (2) we introduce two fresh variables x'_j and x'_k and replace $C_j = R_{1/3}(x_i, x_i, x_j)$ and $C_k = R_{1/3}(y_i, x_k, x_k)$ with $R_{1/3}(x'_j, x'_j, x_j)$ and $R_{1/3}(y_i, x'_k, x_k)$. Case (3) is very similar to case (2) and in case (4) we introduce a fresh variable x''_j and replace $C_k = R_{1/3}(y_i, x_j, x_j)$ with $R_{1/3}(y_i, x''_j, x_j)$. For case (5) first note that $f(x_i) = f(x_j) = f(x_k) = 0$ and that $f(y_i) = f(x'_j) = 1$ in any satisfying assignment f. To ensure that x_i only occurs two times we introduce four fresh variables z_i, z'_i, z''_i, z'''_i, and replace $C_j = R_{1/3}(x_i, x_j, x'_j)$ and $C_k = R_{1/3}(y_i, x_j, x_k)$ with $R_{1/3}(z_i, z'_i, z'_i)$, $R_{1/3}(y_i, x_j, x_k)$, $R_{1/3}(z_i, z''_i, z'''_i)$ and $R_{1/3}(z''_i, z'''_i, x'_j)$. For any satisfying assignment f it then holds that $f(x_i) = f(x_j) = f(x_k) = f(z'_i) = f(z''_i) = f(z'''_i) = 0$, and $f(y_i) = f(x'_j) = f(z_i) = 1$.

In case (6) we replace $R_{1/3}(x_i, x_j, x'_j)$ and $R_{1/3}(y_i, x_k, x'_k)$ with the constraints $R_{1/3}(x'_i, x_j, x'_j)$, $R_{1/3}(y_i, x'_i, x''_i)$, $R_{1/3}(y'_i, x''_i, x'''_i)$, $R_{1/3}(y'_i, x_k, x'_k)$, where x'_i, x''_i, x'''_i and y'_i are fresh variables. For any satisfying assignment f it then holds that $f(x_i) = f(x'_i) = f(x''_i) = f(x'''_i) = 1 - f(y_i) = 1 - f(y'_i)$, and $f(x_j) = 1 - f(x'_j)$. Last, the remaining case (7) is similar to case (6) with the exception that x_k and x'_k only occurs one time in C_k.

Repeating this procedure for every constraint then results in a satisfiability equivalent instance of SAT($\{R_{1/3}\}$)-OCC-2. Moreover the reduction runs in polynomial time with respect to n since the total number of constraints is bounded by $O(n^3)$, which concludes the proof. □

It might be expected that the same holds for SAT($\{R^{\neq\neq\neq}_{1/3}\}$), SAT($\{R^{\neq\neq}_{1/3}\}$)

and SAT($\{R^{\neq}_{1/3}\}$) since they are all as easy as SAT($\{R_{1/3}\}$). Contrary to intuition this is however not the case: SAT($\{R^{\neq\neq\neq}_{1/3}\}$)-DEG-$B$ is NP-complete even for $B = 2$. This also holds for the relations $R^{\neq\neq}_{1/3}$ and $R^{\neq}_{1/3}$. To prove this we first note that $R^{\neq\neq\neq}_{1/3}$, $R^{\neq\neq}_{1/3}$, and $R^{\neq}_{1/3}$ are not Δ-matroid relations.

Definition 32. (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 Hamming distance between x and y. Then x' is a step from x to y if $d(x, x') = 1$ and $d(x, y) = d(x, x') + d(x', y)$. 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 $y) \rightarrow (x' \in R \vee \exists x'' \in R$ that is a step from x' to $y)$.

Lemma 33. *$R^{\neq}_{1/3}, R^{\neq\neq}_{1/3}$ and $R^{\neq\neq\neq}_{1/3}$ are not Δ-matroid relations.*

Proof. We begin with $R^{\neq\neq\neq}_{1/3}$. Let $x = 001110$ and $y = 010101$. These are both elements in $R^{\neq\neq\neq}_{1/3}$. Let $x' = 000110$. Then $d(x, x') = 1$ and $d(x, y) = d(x, x') + d(x', y) = 1 + 3 = 4$, so x' is a step from x to y. For $R^{\neq\neq\neq}_{1/3}$ to be a Δ-matroid relation we must have the following: either $x' \in R^{\neq\neq\neq}_{1/3}$, or there exists a x'' that is a step from x' to y. Since neither of the disjuncts are true, it follows that $R^{\neq\neq\neq}_{1/3}$ is not a Δ-matroid relation. For $R^{\neq\neq}_{1/3}$ and $R^{\neq}_{1/3}$ the proofs are similar but using the tuples $00111, 01010$ and $0011, 1000$ as starting points instead. □

The hardness results then follow from Theorem 3 in Dalmau and Ford [10], which states that SAT(S)-OCC-2 is NP-complete if S contains a relation that is not a Δ-matroid, and from the fact that SAT(S)-OCC-2 is a special case of SAT(S)-DEG-2.

Theorem 34.

- SAT($\{R^{\neq}_{1/3}\}$)*-DEG-2 is NP-complete.*
- SAT($\{R^{\neq\neq}_{1/3}\}$)*-DEG-2 is NP-complete.*
- SAT($\{R^{\neq\neq\neq}_{1/3}\}$)*-DEG-2 is NP-complete.*

8. The Exponential-Time Hypothesis

Even though SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is the easiest NP-complete SAT($\cdot$) problem, we do not hope to prove or disprove that the problems SAT($\{R_{1/3}^{\neq\neq\neq}\}$) or SAT($\{R_{1/3}^{\neq\neq\neq}\}$)-DEG-2 are solvable in polynomial time, since this would settle the P = NP question. A more assailable question is whether the problem can be solved in subexponential time. If yes, then we are none the wiser about P $\stackrel{?}{=}$ NP; but if no, then P $\neq$ NP. As a tool for studying subexponential problems, Impagliazzo et al. [18] proved a *sparsification* lemma for k-SAT. Intuitively, the process of sparsification means that a SAT(S) 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 finite constraint languages S and S' for which SAT(S) and SAT(S') are NP-complete, and use this to prove that SAT($\{R_{1/3}^{\neq\neq\neq}\}$)-DEG-2 is subexponential if and only if the exponential-time hypothesis is false. Due to sparsification we can also prove that *all* such SAT(S) problems are subexponential if and only if *one* of them is subexponential (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 constraint languages.

We first observe that LV-reductions are insufficient for these purposes since, under the assumption that coNP $\not\subseteq$ P/poly, SAT(Γ_{SAT}^k) for $k \geq 4$ is not LV-reducible to SAT($\{R_{1/3}^{\neq\neq\neq}\}$).

Lemma 35. *There is no LV-reduction from* SAT(Γ_{SAT}^k)*,* $k \geq 4$*, to* SAT($\{R_{1/3}^{\neq\neq\neq}\}$) *unless coNP* $\subseteq$ *P/poly (and the polynomial hierarchy collapses).*

Proof. We assume that $k = 4$ since the other cases are entirely similar. Assume (with the aim of reaching a contradiction) that there is an LV-reduction from SAT(Γ_{SAT}^4) to SAT($\{R_{1/3}^{\neq\neq\neq}\}$). We know that there is an LV-reduction (with parameter 1) from SAT($\{R_{1/3}^{\neq\neq\neq}\}$) to SAT($\Gamma_{\text{SAT}}^3$) by Theorem 10. Taken together, this would imply the existence of an LV-reduction from SAT(Γ_{SAT}^4) to SAT(Γ_{SAT}^3). Dell & van Melkebeek [12, Corollary 1] show that, under the hypothesis that

coNP $\not\subseteq$ P/poly, polynomial-time many-one reductions cannot reduce the density of n-variable SAT(Γ^d_{SAT}) instances to $O(n^c)$ clauses for any constant $c < d$. Assume there is an LV-reduction (which is a restricted kind of polynomial-time many-one reduction) with parameter C from SAT(Γ^4_{SAT}) to SAT(Γ^3_{SAT}). Let ϕ be an arbitrary instance (with n variables) of SAT(Γ^4_{SAT}) and ϕ' the resulting SAT(Γ^3_{SAT}) instance. The instance ϕ' contains at most $C \cdot n + O(1)$ variables so it contains at most $O(8(C \cdot n)^3) = O(n^3)$ distinct constraints. This contradicts Dell & van Melkebeek's result. □

However, as we show in the forthcoming sections, this restriction can be overcome with the help of sparsification.

8.1. Two Notions of Sparsification

Sparsification in its original formulation by Impagliazzo *et al.* [18] is only defined for k-SAT formulas. We generalize the definition to arbitrary constraint languages, which leads to our first version of sparsification.

Definition 36 (Sparsification$_1$). Let S and S' be two finite constraint languages. We say that S is *sparsifiable*$_1$ into S' if, for all $\varepsilon > 0$ and for all SAT(S) instances ϕ (with n variables), there exists a set $\{\phi_1, \ldots, \phi_t\}$ of SAT(S') instances such that

1. ϕ is satisfiable if and only if at least one ϕ_i is satisfiable,
2. ϕ_i contains at most $C \cdot n$ constraints, where C only depends on ε, S and S',
3. $t \leq 2^{\varepsilon n}$, and
4. $\{\phi_1, \ldots, \phi_t\}$ can be computed in $O(poly(n) \cdot 2^{\varepsilon n})$ time.

Note that nothing in the definition says that S and S' cannot be the same constraint language. If so, we simply say that S is *sparsifiable*$_1$. Impagliazzo *et al.* [18] prove the following result for k-SAT.

Lemma 37 (sparsification$_1$ lemma for k-SAT). *Γ^k_{SAT} is sparsifiable$_1$ for $k \geq 3$.*

We will often use a slightly different formulation of sparsification which we denote *sparsification$_2$*. In the second version, we focus on the degree of variables rather than the number of constraints.

Definition 38 (Sparsification$_2$). Let S and S' be two finite constraint languages. We say that S is *sparsifiable$_2$* into S' if, for all $\varepsilon > 0$ and for all SAT(S) instances ϕ (with n variables), there exists a set $\{\phi_1, \ldots, \phi_t\}$ of SAT(S') instances such that

1. ϕ is satisfiable if and only if at least one ϕ_i is satisfiable,
2. ϕ_i is a SAT(S')-DEG-B instance, where B only depends on ε, S and S',
3. ϕ_i contains at most $D \cdot n$ variables, where D only depends on ε, S and S',
4. $t \leq 2^{\varepsilon n}$, and
5. $\{\phi_1, \ldots, \phi_t\}$ can be computed in $O(poly(n) \cdot 2^{\varepsilon n})$ time.

It should come as no surprise that sparsification$_1$ and sparsification$_2$ are tightly related concepts. It is easy to see that sparsification$_2$ implies sparsification$_1$: assume that every variable in ϕ_i has degree $\leq B$ and ϕ_i contains at most $D \cdot n$ variables. Then, variable x can appear in at most B distinct constraints and hence there can be at most $(B \cdot D) \cdot n$ distinct constraints in ϕ_i.

It is also the case that sparsification$_1$ implies sparsification$_2$ under a mild additional assumption.

Lemma 39. *Assume S is sparsifiable$_1$ into S'. Then, S is sparsifiable$_2$ into S' if the equality relation is e.f.p.p. definable in S'.*

Proof. Assume $Eq(x, y) \equiv \exists z_1, \ldots, z_k.\tau(x, y, z_1, \ldots, z_k)$ is an e.f.p.p. implementation of Eq. Let p denote the maximum degree of any variable in this implementation and let $q = k + 2$ denote the total number of variables. Let

ϕ be a SAT(S) instance and let $\{\phi_1, \ldots, \phi_t\}$ be a sparsification$_1$ into S' (for some $\varepsilon > 0$). Let k be the maximum arity of the relations in S'. Let us consider ϕ_i and assume V is the set of variables occurring in ϕ_i. Arbitrarily choose a variable $x \in V$ and assume that x appears m times in ϕ_i. Introduce m fresh variables $x_1, \ldots, x_m$ and constraints

$$Eq(x_1, x_2), Eq(x_2, x_3), \ldots, Eq(x_{m-1}, x_m).$$

Finally, replace the i-th occurrence of x in ϕ_i with the variable x_i. Do this for all variables $x \in V$ and let ϕ'_i denote the resulting formula. Note the following:

1. The formula ϕ_i contains at most $C \cdot n$ constraints so it contains at most $C \cdot n \cdot k$ variable occurrences. This implies that ϕ'_i contains at most $(C \cdot k) \cdot n + (C \cdot k \cdot q) \cdot n$ variables where $(C \cdot k \cdot q) \cdot n$ is an upper bound on the number of variables used for implementing the Eq constraints.

2. Each variable occurring in ϕ'_i has degree at most $1 + 2p$: it appears in at most one position in the "original constraints" and in $2p$ positions in the constraints that e.f.p.p. implements Eq.

It follows that $\{\phi'_1, \ldots, \phi'_t\}$ is a sparsification$_2$ of ϕ. □

Whenever we study constraint languages in $\mathcal{H}$, we may actually view the two definitions as equivalent due to Lemma 29. Since $\Gamma^k_{\text{SAT}} \in \mathcal{H}$ for $k \geq 3$ we immediately get the following.

Lemma 40. *Γ^k_{SAT} is sparsifiable$_2$.*

We also get the following simple connection between sparsification and subexponential complexity.

Lemma 41. *Arbitrarily choose $S, S' \in \mathcal{H}$ and assume that S is sparsifiable$_2$ into S'. If SAT(S')-DEG-B is subexponential for all B, then SAT(S) is subexponential.*

PROOF. Assume that we want to solve SAT(S) in time $O(poly(n) \cdot 2^{n\varepsilon})$ for some $\varepsilon > 0$. For every $\alpha > 0$ and B, there exists a time $O(poly(n) \cdot 2^{n\alpha})$ algorithm $A_{\alpha,B}$ for SAT(S')-DEG-B. Arbitrarily choose an instance ϕ of SAT(S) with n variables. Sparsify ϕ with parameter $\varepsilon/2$ into SAT(S') and let $\{\phi_1, \ldots, \phi_t\}$ be the resulting set of SAT(S')-DEG-B instances (where B is only dependent on S, S', and ε). This takes $O(poly(n) \cdot 2^{(\varepsilon/2)n})$ time, $t \leq 2^{(\varepsilon/2)n}$ holds, and each ϕ_i contains $C \cdot n$ variables for some constant C that only depends on ε. Let $\alpha = \varepsilon/2C$ and apply $A_{\alpha,B}$ to the instances $\phi_1, \ldots, \phi_t$. If there is a satisfiable ϕ_i, then ϕ is satisfiable, and otherwise it is not. The total time for this is $O(poly(C \cdot n) \cdot t \cdot 2^{(\varepsilon/2C) \cdot Cn)}) = O(poly(n) \cdot 2^{(\varepsilon/2)n} \cdot 2^{(\varepsilon/2)n}) = O(poly(n) \cdot 2^{\varepsilon n})$. □

8.2. Sparsification Within $\mathcal{H}$

In order to prove the general sparsification result (*i.e.*, that all constraint languages $S, S' \in \mathcal{H}$ can be sparsified$_2$ into each other), we first prove that Γ^k_{NAE} is sparsifiable$_2$.

Lemma 42 (sparsification$_2$ lemma for NAE-k-SAT). *Γ^k_{NAE} is sparsifiable$_2$.*

PROOF. Arbitrarily choose $\varepsilon > 0$ and let ϕ be a SAT(Γ^k_{NAE}) instance with n variables. The proof proceeds in five steps.

1. ϕ is LV-reduced into a Γ^k_{SAT} instance ψ,
2. ψ is sparsified$_2$ which yields t Γ^k_{SAT} instances $\psi_1, \ldots, \psi_t$,
3. each ψ_i is LV-reduced into a $\Gamma^{k+1}_{\text{NAE}}$ instance ψ'_i,
4. each ψ'_i is LV-reduced into a $\Gamma^{k+1}_{\text{NAE}}$ instance ψ''_i in which all variables have a bounded degree,
5. each ψ''_i is LV-reduced into a Γ^k_{NAE} instance ϕ_i.

Steps 1 and 2. Note that $\gamma^k_{\text{NAE}}(x_1, \ldots, x_k)$ has the following q.f.p.p. implementation in terms of γ^k_{SAT}:

$$\gamma^k_{\text{NAE}}(x_1, \ldots, x_k) \equiv \gamma^k_{\text{SAT}}(x_1, \ldots, x_k) \wedge \gamma^k_{\text{SAT}}(\neg x_1, \ldots, \neg x_k).$$

In other words, if $\gamma^k_{\text{NAE}}(x_1, \ldots, x_k)$ is a constraint from ϕ, then it is satisfiable if and only if $\gamma^k_{\text{SAT}}(x_1, \ldots, x_k) \wedge \gamma^k_{\text{SAT}}(\neg x_1, \ldots, \neg x_k)$ is satisfiable. We can therefore form an equivalent $\text{SAT}(\Gamma^k_{\text{SAT}})$ instance ψ by adding the complement of every γ^k_{NAE}-constraint. By the sparsification$_2$ lemma for k-SAT (Lemma 40), it follows that ψ can be sparsified$_2$ into $\{\psi_1, \ldots, \psi_t\}$.

Step 3. Arbitrarily choose ψ_i, $1 \leq i \leq t$, and let V denote the set of variables appearing in it. We begin by introducing one fresh variable X. For each constraint $\gamma^k_{\text{SAT}}(x_1, \ldots, x_k) \in \psi_i$, we let $\gamma^{k+1}_{\text{NAE}}(x_1, \ldots, x_k, X)$ be the corresponding $\gamma^{k+1}_{\text{NAE}}$-constraint, and we let ψ'_i be the resulting $\text{SAT}(\Gamma^{k+1}_{\text{NAE}})$ instance. Then, ψ_i is satisfiable if and only if ψ'_i is satisfiable: if ψ_i is satisfied by the assignment f, then ψ'_i is satisfied by the assignment f' defined such that $f'(x) = f(x)$ if $x \in V$ and $f'(X) = 0$. For the other direction, assume ψ'_i is satisfied by the assignment f'. If $f'(X) = 0$, then each constraint has at least one literal that is not assigned 0 by f, by what it follows that ψ_i must be satisfiable. If $f'(X) = 1$, then we note that the complement f'' of f' defined as $f''(x) = 1 - f'(x)$ is a satisfying assignment, too, and we can apply the same reasoning as above.

Since ψ was sparsified$_2$, the degree of the variables in ψ_i is bounded by some constant B. It follows that the number of constraints in ψ_i is at most $B \cdot |V|$ and, consequently, that variable X has at most degree $B \cdot |V|$ in ψ'_i.

Step 4. Now, consider ψ'_i and let W denote the degree of X. We continue by proving that the degree of X can be reduced to a constant value. To do this, we e.f.p.p. implement the equality relation in $\Gamma^{k+1}_{\text{NAE}}$:

$$Eq(x, y) \equiv \exists z. \gamma^{k+1}_{\text{NAE}}(x, z, \ldots, z) \wedge \gamma^{k+1}_{\text{NAE}}(z, \ldots, z, y).$$

We see that the highest degree of any variable in this definition is $2k$.

To decrease the degree of X, we introduce W fresh variables $X_1, \ldots, X_W$ and the following chain of equality constraints:

$$Eq(X_1, X_2) \wedge Eq(X_2, X_3) \wedge \ldots \wedge Eq(X_{W-1}, X_W).$$

Thereafter, we replace the occurrence of variable X in ψ'_i with variable X_i. Let the resulting instance be ψ''_i and note that ψ'_i is satisfiable if and only if ψ''_i

is satisfiable. Also note that the maximum degree B'' of any variable in ψ_i'' is bounded by $\max\{B, 2k, 3\}$ (where B is the degree of variables in V, $2k$ is the degree of variables in the construction of Eq, and 3 is the degree of the variables $X_1, \ldots, X_W$). Furthermore, we introduce no more than $2W$ new variables—the variables $X_1, \ldots, X_W$ and one auxiliary variable per Eq constraint. Hence,

- ψ'' contains at most $|V| + 2W \leq |V| + 2 \cdot B \cdot |V| = (2B+1)|V|$ variables, and
- every variable in ψ'' has degree at most B'' where B'' depends only on B and k.

Step 5. We know that Eq is e.f.p.p. definable in Γ^k_{NAE} (by the construction in Step 4.) and we know that no variable occurs more than B'' times in ψ''. Consequently, we can apply Lemma 28 and LV-reduce ψ_i'' to an equivalent $\text{SAT}(\Gamma^k_{\text{NAE}})$-DEG-$C$ instance ϕ_i for some C that only depends on B'' and k.

We now verify that the formula $\{\phi_1, \ldots, \phi_t\}$ is indeed a sparsification of ϕ.

1. ϕ is satisfiable if and only if at least one ϕ_i is satisfiable. This is clear from the transformations above.
2. ϕ_i is a $\text{SAT}(\Gamma^k_{\text{NAE}})$-DEG-$C$ instance, where C only depends on ε and k. Tracing the proof backwards, we see that
   (a) C depends only on B'' and k,
   (b) B'' depends only on B and k, and
   (c) B depends only on ε and k.
3. ϕ_i contains at most $D'' \cdot n$ variables where D'' only depends on ε and k. The instance ψ_i'' contains at most $(2B+1) \cdot |V|$ variables where $|V| \leq D \cdot n$ for some D that only depends on ε and k, and B depends on ε and k only. This implies that there exists a D' (that only depends on ε and k) such that the number of variables in ϕ_i is $\leq D' \cdot D \cdot n$ by Lemma 28.

4. $t \leq 2^{\varepsilon n}$. The transformation from ϕ to ψ in step 1 does not introduce any new variables. Hence, the value t is determined by the initial sparsification only and the bound follows from Lemma 40.

5. $\{\phi_1, \ldots, \phi_t\}$ can be computed in $O(poly(n) \cdot 2^{\varepsilon n})$ time. We begin with a $\mathrm{SAT}(\Gamma^k_{\mathrm{NAE}})$ instance ϕ (with n variables) and construct the $\mathrm{SAT}(\Gamma^k_{\mathrm{SAT}})$ instance ψ. This can clearly be done in linear time and the size of ϕ is linearly related to the size of ψ. In particular, ϕ contains as many variables as ψ. Thus, the sparsification$_2$ process takes $O(poly(n) \cdot 2^{\varepsilon n})$ time and each instance ψ'_i is only linearly larger than ψ. From this point, we only apply a series of LV-reductions to each ψ'_i. Since they can be computed in time $O(poly(n))$, the time bound follows.

By combining step 1 to step 5 we then get the required sparsification of Γ^k_{NAE}. □

The following auxiliary lemma establishes that for any finite constraint language S such that $S \subset \mathrm{BR}$ (resp. $S \subset \mathrm{IN}_2$) it holds that $\mathrm{SAT}(S)$ is LV-reducible to k-SAT (resp. NAE-k-SAT) for some k.

Lemma 43. *Let S be a finite constraint language such that $S \subset \mathrm{IN}_2$ (respectively $S \subset \mathrm{BR}$). Let K denote the maximum arity of the relations in S. Then $\mathrm{SAT}(S)$ is LV-reducible (with parameter 1) to $\mathrm{SAT}(\Gamma^K_{\mathrm{NAE}})$ (respectively $\mathrm{SAT}(\Gamma^K_{\mathrm{SAT}})$).*

Proof. We give the proof for $S \subset \mathrm{IN}_2$; the case $S \subset \mathrm{BR}$ is analogous.

Let ϕ be an instance of $\mathrm{SAT}(S)$ with n variables and let $R \in S$ be a relation with arity k. By definition, $R = \{0,1\}^k \setminus E$, where E is a set of k-ary tuples over $\{0,1\}$ that describes the excluded tuples in the relation. Since all relations in S are 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 and assume that $e_i = (b_{i,1}, \ldots, b_{i,k})$, $b_{i,j} \in \{0,1\}$. If $R(x_1, \ldots, x_k)$ is a constraint in ϕ, then it can be expressed by the $\mathrm{SAT}(\Gamma^k_{\mathrm{NAE}})$ formula $\psi_1 \wedge \ldots \wedge \psi_N$, where each $\psi_i = \gamma^k_{\mathrm{NAE}}(y_1, \ldots, y_k)$, 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 $R(x_1, \ldots, x_k)$ 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 linear in the size of ϕ (because S is fixed and finite), the reduction is an LV-reduction from $\mathrm{SAT}(S)$ to to $\mathrm{SAT}(\Gamma^K_{\mathrm{NAE}})$. □

We can now prove that sparsification is possible between all S, S' in $\mathcal{H}$.

Theorem 44. ***(general sparsification)*** *Arbitrarily choose $S, S' \in \mathcal{H}$. Then, S is sparsifiable$_2$ into S'.*

Proof. Throughout the proof, let k denote the highest arity of any relation in S. There are a few different cases depending on which co-clones are generated by S and S':

1. $\langle S \rangle = \langle S' \rangle = \mathrm{IN}_2$,
2. $\langle S \rangle = \langle S' \rangle = \mathrm{BR}$,
3. $\langle S \rangle = \mathrm{IN}_2$, $\langle S' \rangle = \mathrm{BR}$
4. $\langle S \rangle = \mathrm{BR}$, $\langle S' \rangle = \mathrm{IN}_2$.

Case 1. Arbitrarily choose $\varepsilon > 0$. Every $\mathrm{SAT}(S)$ instance ϕ with n variables can be reduced to a $\mathrm{SAT}(\Gamma^k_{\mathrm{NAE}})$ instance ϕ' with the same number of variables by Lemma 43. We sparsify$_2$ this instance according to Lemma 42 and obtain the set $\{\phi'_1, \ldots, \phi'_t\}$. Each ϕ'_i is an instance of $\mathrm{SAT}(\Gamma^k_{\mathrm{NAE}})$-DEG-$B$ for some B that only depends on S, k, and ε. By Lemma 29, the relation Eq is e.f.p.p. implementable in Γ^k_{NAE}. We can thus apply Lemma 28 (since $\langle S' \rangle = \mathrm{IN}_2$ implies that $\Gamma^k_{\mathrm{NAE}} \subseteq \mathrm{IN}_2 = \langle S' \rangle$) and LV-reduce this instance to an instance ϕ''_i of $\mathrm{SAT}(S')$-DEG-C for some C that only depends on S, S', and B. One can now easily verify that $\{\phi''_1, \ldots, \phi''_t\}$ is a sparsification of ϕ.

Case 2. Analogous to Case 1 by using Γ^k_{SAT} instead of Γ^k_{NAE}.

Case 3. By Lemma 43 it follows that SAT(S) can be LV-reduced (with parameter 1) to SAT(Γ^k_{SAT}). Thereafter we can proceed as in Case 2.

Case 4. Arbitrarily choose $\varepsilon > 0$. If ϕ is a SAT(S) instance with n variables, it can be LV-reduced (with parameter 1) to a SAT(Γ^k_{SAT}) instance ϕ' by Lemma 43. Sparsify$_2$ this instance (with parameter ε) into $\{\phi'_1, \ldots, \phi'_t\}$ according to Lemma 42. By recapitulating steps 3 and 4 from the proof of Lemma 42, we can then LV-reduce each ϕ'_i to a SAT($\Gamma^{k+1}_{\text{NAE}}$)-DEG-$B$ instance ϕ''_i. We can now apply Lemma 28 (since $\langle S' \rangle = \text{IN}_2$ implies that $\Gamma^{k+1}_{\text{NAE}} \subseteq \text{IN}_2 = \langle S' \rangle$) and LV-reduce this instance to an instance ϕ'''_i of SAT(S')-DEG-C for some C that only depends on S, S', and B. It is now straightforward to verify that $\phi'''_1, \ldots, \phi'''_t$ is a sparsification$_2$ of ϕ. □

Santhanam and Srinivasan [33] have shown that the unrestricted SAT problem (which corresponds to an infinite constraint language) does not admit sparsification to arbitrary finite constraint languages such that SAT($\cdot$) is NP-complete. Consequently, it is a necessary condition that the constraint languages in Theorem 44 are indeed finite. Related questions are discussed by Dell & van Melkebeek [12].

8.3. SAT and the Exponential-Time Hypothesis

A related question to the exponential-time hypothesis is which problems can be proved to be subexponential if and only if k-SAT is subexponential with the help of a size-preserving reduction. Impagliazzo *et al.* [18] prove that many NP-complete problems such as k-colorability, clique and vertex cover are as hard as k-SAT with respect to subexponential complexity. In this section, we prove that both SAT($\{R^{\neq\neq\neq}_{1/3}\}$) and SAT($\{R^{\neq\neq\neq}_{1/3}\}$)-DEG-2 are subexponential if and only if k-SAT is subexponential, and, as a consequence, that this is true for all finite constraint languages S for which SAT(S) is NP-complete — even for degree-bounded instances. The following lemma together with the general sparsification result from the previous section are the crucial ingredients in the proof of the main result (Theorem 46).

Lemma 45. *If* $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$*-DEG-2 is subexponential, then* $\mathrm{SAT}(\{R_{1/3}^{\neq\neq\neq}\})$ *is subexponential.*

PROOF. Let *Neq* denote the disequality relation $\{(0,1),(1,0)\}$ and Eq_k the k-ary equality relation $\{(0,\ldots,0),(1,\ldots,1)\}$.

Claim 1. $R_{1/3}^{\neq\neq\neq}$ can e.f.p.p. implement $Neq(x,y)$ such that every variable in the implementation has degree 1. Merely observe that $Neq(x,y)$ has the e.f.p.p. implementation $\exists z,w,z',w'.R_{1/3}^{\neq\neq\neq}(x,z,w,y,z',w')$.

Claim 2. $R_{1/3}^{\neq\neq\neq}$ can e.f.p.p. implement $Eq_3(x_1,x_2,x_3)$ such that variables $x_1,\ldots,x_3$ have degree 1 and all other variables have degree at most 2. Consider the following implementation:

$$\begin{aligned} Eq_3(x_1,x_2,x_3) \equiv \exists x_1', z, w, z', w', h_1, \ldots, h_4, h_5. \\ Neq(x_1,x_1') \wedge R_{1/3}^{\neq\neq\neq}(z,w,h_1,x_1',x_2,h_2) \wedge Neq(z,z') \wedge \\ Neq(w,w') \wedge R_{1/3}^{\neq\neq\neq}(z',w',h_3,x_3,h_4,h_5). \end{aligned}$$

It is straightforward to verify that it satisfies the degree bounds stated above. To see that it actually implements Eq_3, first note that it is possible to assign the variables $z,w,z',w',h_1,\ldots,h_5$ values that satisfy the formula for every assignment σ such that $\sigma(x_1)=\sigma(x_2)=\sigma(x_3)$. Next, we show that every satisfying assignment forces x_1,x_2,x_3 to take the same value. We first verify that an arbitrary satisfying assignment σ satisfies $\sigma(z)\neq\sigma(w)$, *i.e.*, $\sigma(z)=1-\sigma(w)$. Assume $\sigma(z)=0$. Then, $\sigma(z')=1\Rightarrow\sigma(w')=0\Rightarrow\sigma(w)=1$. One can similarly show that if $\sigma(z)=1$, then $\sigma(w)=0$. Now let σ be an arbitrary satisfying assignment. We see that

- $\sigma(x_1)=1-\sigma(x_1')=\sigma(z)=1-\sigma(w)=\sigma(x_2)$ and
- $\sigma(x_3)=1-\sigma(z')=\sigma(z)=1-\sigma(x_1')=\sigma(x_1)$.

This implies that $\sigma(x_1)=\sigma(x_2)=\sigma(x_3)$ and that we have an implementation of Eq_3.

Claim 3. $R_{1/3}^{\neq\neq\neq}$ can e.f.p.p. implement $Eq_k(x_1,\ldots,x_k)$ such that $x_1,\ldots,x_k$ have degree 1 and all other variables have degree at most 2. The case when

$k \leq 3$ follows from Claim 2. If we have an implementation of Eq_k satisfying the degree bounds, then

$$Eq_{k+1}(x_1, \ldots, x_{k+1}) \equiv \exists x'.Eq_k(x_1, \ldots, x_{k-1}, x') \wedge Eq_3(x', x_k, x_{k+1})$$

is a valid implementation that satisfies the degree bounds, too.

Arbitrarily choose $\varepsilon > 0$ and let ψ be an arbitrary instance of SAT($\{R_{1/3}^{\neq\neq\neq}\}$) with n variables. Sparsify$_2$ ψ (with parameter ε) into $\{R_{1/3}^{\neq\neq\neq}\}$ according to Theorem 44 and let $\{\psi_1, \ldots, \psi_t\}$ be the result. Each ψ_i is an instance of SAT($\{R_{1/3}^{\neq\neq\neq}\}$)-DEG-$B$ where B only depends on ε. Let V be the set of variables in ψ_i. Assume that x is a variable in V that has degree $2 < B' \leq B$. Replace the j-th occurrence of x with the fresh variable x_j. Finally, use the construction in Claim 3 and add the constraint $Eq_{B'}(x_1, \ldots, x_{B'})$. Do this for all variables occurring in V and let ψ_i' denote the resulting instance. It is not hard to see that ψ_i' is indeed an instance of SAT($\{R_{1/3}^{\neq\neq\neq}\}$)-DEG-2. We need to calculate the number of variables in ψ_i'. The variables in ψ_i have at most degree B so ψ' contains at most $B \cdot |V|$ variables if we do not count the variables introduced by the Eq_k constraints. For simplicity, we assume that we have added $|V|$ Eq_B constraints. Let D be the number of variables in the implementation of Eq_3 in Claim 2. An Eq_B constraint is constructed by "chaining together" B Eq_3 constraints—this is done by introducing $B - 3$ new variables. Hence, one Eq_B constraint contains $B + (B - 3) + B \cdot D$ variables. All in all, the instance ψ_i contains at most $|V| \cdot (B + (B - 3) + B \cdot D) + |V| \cdot B = |V| \cdot (B(D + 3) - 3)$ variables. We see that $B(D + 3) - 3$ depends on ε only since D is a universal constant.

Assume now that we want to solve SAT($\{R_{1/3}^{\neq\neq\neq}\}$) in time $O(poly(n) \cdot 2^{n\varepsilon})$ for some $\varepsilon > 0$. For every $\alpha > 0$, there exists a time $O(poly(n) \cdot 2^{n\alpha})$ algorithm A_α for SAT($\{R_{1/3}^{\neq\neq\neq}\}$)-DEG-2. Arbitrarily choose an instance ψ of SAT($\{R_{1/3}^{\neq\neq\neq}\}$) with n variables. Apply the transformation described above with parameter $\varepsilon/2$ and let $\{\psi_1, \ldots, \psi_t\}$ be the resulting set of SAT($\{R_{1/3}^{\neq\neq\neq}\}$)-DEG-2 instances. This takes $O(poly(n) \cdot 2^{(\varepsilon/2)n})$ time, $t \leq 2^{(\varepsilon/2)n}$, and each ψ_i contains $C \cdot n$ variables for some constant C that only depends on ε. Let $\alpha = \varepsilon/2C$ and apply

A_α to the instances $\psi_1, \ldots, \psi_t$. If there is a satisfiable ψ_i, then ψ is satisfiable and, otherwise, not. This process takes time $O(poly(C \cdot n) \cdot t \cdot 2^{(\varepsilon/2C)\cdot Cn)}) = O(poly(n) \cdot 2^{(\varepsilon/2)n} \cdot 2^{(\varepsilon/2)n}) = O(poly(n) \cdot 2^{\varepsilon n})$. □

This lemma can be viewed as yet another proof of NP-hardness for SAT($\{R^{\neq\neq\neq}_{1/3}\}$)-DEG-2. The transformation from ψ_i to ψ'_i is a polynomial-time reduction from SAT($\{R^{\neq\neq\neq}_{1/3}\}$)-DEG-$B$ to SAT($\{R^{\neq\neq\neq}_{1/3}\}$)-DEG-2 (for any fixed B), and SAT($\{R^{\neq\neq\neq}_{1/3}\}$)-DEG-$B$ is NP-hard for some B by Theorem 3.

Theorem 46. *The following statements are equivalent:*

1. *The exponential-time hypothesis is false.*
2. SAT($\{R^{\neq}_{1/3}\}$)*-DEG-2 is subexponential.*
3. SAT($\{R^{\neq\neq}_{1/3}\}$)*-DEG-2 is subexponential.*
4. SAT($\{R^{\neq\neq\neq}_{1/3}\}$)*-DEG-2 is subexponential.*
5. SAT($\{R^{\neq\neq\neq}_{1/3}\}$) *is subexponential.*
6. *For every* $S \in \mathcal{H}$, SAT(S) *is subexponential.*
7. *For every* $S \in \mathcal{H}$, SAT(S)*-DEG-*B *is subexponential for every* B*.*
8. *There exists an* $S \in \mathcal{H}$ *such that* SAT(S) *is subexponential.*
9. *There exists an* $S \in \mathcal{H}$ *such that* SAT(S)*-DEG-*B *is subexponential for all* B*.*

Proof. 1 $\implies$ 2: If the exponential-time hypothesis is false then 3-SAT is subexponential. But then $R^{\neq}_{1/3}$ must also be subexponential since it is as easy as 3-SAT, which immediately implies that SAT($\{R^{\neq}_{1/3}\}$)-DEG-2 is subexponential.

2 $\implies$ 3: Assume that SAT($\{R^{\neq}_{1/3}\}$)-DEG-2 is subexponential. We give a size-preserving reduction from SAT($\{R^{\neq\neq}_{1/3}\}$)-DEG-2 to SAT($\{R^{\neq}_{1/3}\}$)-DEG-2. Let ϕ be an instance of SAT($\{R^{\neq\neq}_{1/3}\}$)-DEG-2 with n variables and $2n$ constraints. Then we replace every constraint $R^{\neq\neq}_{1/3}(x_1, x_2, x_3, x'_1, x'_2)$ with

$R^{\neq}_{1/3}(x_1, x''_2, x_3, x'_1) \wedge R^{\neq}_{1/3}(x_2, a, b, x'_2) \wedge R^{\neq}_{1/3}(x''_2, a, b, x'''_2)$, where x'_2, x''_2, x'''_2, a and b are fresh variables. As can be verified $\sigma(x''_2) = \sigma(x_2) = 1 - \sigma(x'_2)$ for all models σ, and, furthermore, the degree of any newly introduced variables is at most two. If we repeat the procedure for every constraint in ϕ we get an equivalent instance ϕ' of SAT($\{R^{\neq}_{1/3}\}$)-DEG-2 where the number of variables is bounded by $10n$.

3 $\Longrightarrow$ 4: Same reduction as in the previous case.

4 $\Longrightarrow$ 5: By Lemma 45.

5 $\Longrightarrow$ 6: Assume that SAT($\{R^{\neq\neq\neq}_{1/3}\}$) is subexponential. Clearly, SAT($\{R^{\neq\neq\neq}_{1/3}\}$)-DEG-$B$ is subexponential for all B, too. According to the general sparsification result (Theorem 44), (S) can be sparsified into $(\{R^{\neq\neq\neq}_{1/3}\})$. Hence, SAT($S$) is subexponential by Lemma 41.

6 $\Longrightarrow$ 7, 6 $\Longrightarrow$ 8, 7 $\Longrightarrow$ 9, 8 $\Longrightarrow$ 9: Trivial.

9 $\Longrightarrow$ 1: Assume that SAT(S)-DEG-B is subexponential for all B. By Theorem 44, SAT(Γ^3_{SAT}) can be sparsified into SAT(S) so the implication follows from Lemma 41.

When proving 5 $\Longrightarrow$ 6 we do not require that the instances have bounded degree. Hence, there may be a more direct way of proving the implication without using sparsification: one may for instance think of LV-reductions from SAT(S) to SAT($\{R^{\neq\neq\neq}_{1/3}\}$). We remind the reader that the existence of such reductions (for all $S \in \mathcal{H}$) are unlikely since they imply coNP $\subseteq$ P/poly by Lemma 35.

9. Research Directions and Open Questions

In this article we have provided a general framework to study the complexity of NP-hard problems and applied this to the parameterized Boolean satisfiability problem. In particular this has allowed us to get a more nuanced view of the complexity of SAT($\cdot$) than possible with the algebraic approach in Jeavons [20]. In other words studying the complexity of a problem with our approach provides

much more information than merely stating that it is polynomial-time solvable or NP-complete. Througout the article we have also seen that clone theoretical results in fact can yield highly practical applications. For example the proof that SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is the easiest NP-complete SAT($\cdot$) problem is based on determining the largest element in an interval of strong partial clones — a result that might otherwise be believed to be of purely theoretical interest. With this result we could also obtain a full understanding of the complexity of SAT(S) with respect to subexponential complexity for all finite languages S. We therefore deem it likely that further investigations in partial clone theory, and in particular improving the classification in Section 6, can lead to a better understanding of the complexity differences between SAT($\cdot$) problems.

We will now discuss some research directions and pose some open questions.

The complexity of SAT($\{R_{1/3}^{\neq\neq\neq}\}$)

After proving that SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is the easiest NP-complete SAT(S) problem, it is tempting to try to determine bounds on its time complexity. Lemma 17 and 18 gives an LV-reduction to 1-IN-3-SAT with parameter $\frac{1}{2}$. It would be interesting to determine whether it is possible to construct better algorithms for SAT($\{R_{1/3}^{\neq\neq\neq}\}$) and SAT($\{R_{1/3}^{\neq\neq\neq}\}$)-DEG-2 that are better than reductions to 1-IN-3-SAT.

The relative complexity between SAT($\{R_{1/3}^{\neq\neq\neq}\}$), SAT($\{R_{1/3}^{\neq\neq}\}$) *and* SAT($\{R_{1/3}^{\neq}\}$)

Since our definition of 'easier' means 'not harder than' we have not excluded the possibility that there exists other constraint languages that are as easy as $R_{1/3}^{\neq\neq\neq}$. Ultimately one would like to prove that SAT($\{R_{1/3}^{\neq\neq\neq}\}$) is strictly easier than SAT($\{R_{1/3}^{\neq\neq}\}$), SAT($\{R_{1/3}^{\neq}\}$), and SAT($\{R_{1/3}\}$). The techniques used in Lemma 17 however appear to fall short from the goal since they require that the language contains all possible sign patterns, which is of course not true for $\{R_{1/3}^{\neq\neq}\}$, $\{R_{1/3}^{\neq}\}$, and $\{R_{1/3}\}$. At the same time adding additional sign patterns to a language does in general not decrease the performance of branch-and-reduce algorithms but only introduces more cases. Hence a possible starting point

might be to find examples of languages S and S', where S' is obtained from S by closing it under sign patterns, but where $\mathrm{SAT}(S')$ is still CV-reducible to $\mathrm{SAT}(S)$.

Partial co-clones covering BR

Another line of research is to continue the investigation of the partial co-clone lattice in the bottom of BR. The inclusions proved for $\langle R_{1/3}^{\neq\neq\neq 01}\rangle_{\not\exists}$, $\langle R_{1/3}^{\neq\neq 01}\rangle_{\not\exists}$, $\langle R_{1/3}^{\neq 01}\rangle_{\not\exists}$ and $\langle R_{1/3}^{01}\rangle_{\not\exists}$ are incomplete in the sense that we have not excluded the existence of other partial co-clones in these intervals. A priori there could exist any – finite, infinite or uncountably infinite – number of partial co-clones besides the ones given and our belief in the conjecture is mainly based on the seemingly impossibility to construct a language resulting in a partial co-clone that lies inbetween. Another related task is to consider the relation $R_{2/3} = \{(1,1,0),(1,0,1),(0,1,1)\}$ instead of $R_{1/3}$ and determine the partial co-clones between $\langle R_{2/3}\rangle_{\not\exists}$ and $\langle R_{2/3}^{\neq\neq\neq 01}\rangle_{\not\exists} = \langle R_{1/3}^{\neq\neq\neq 01}\rangle_{\not\exists}$.

Frozen partial co-clones

As we have already mentioned, a drawback of Theorem 10 is that the structure of the Boolean strong 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 strong partial clone lattice. One plausible candidate is the lattice of *frozen* partial clones introduced by Nordh and Zanuttini [30]. A *frozen* implementation 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 Nordh and Zanuttini [30]. We remark that the complexity of $\mathrm{SAT}(S)$ is determined by the frozen partial clones and that the lattice of frozen partial clones is indeed coarser than the lattice of partial clones. For example the partial co-clones below $\langle \Gamma_{1/3}\rangle_{\not\exists}$ in

Figure 6 collapse to the four frozen partial co-clones generated by $R_{1/3}^{\neq\neq\neq}$, $R_{1/3}^{\neq\neq}$ $R_{1/3}$ and $R_{1/3}$. There are also examples of infinite chains of partial clones that collapse to a single frozen partial clone [15, 30]. This suggests that in many cases, frozen partial co-clones result in considerable simplifications in comparison to partial co-clones.

SAT(S)-*DEG-B and the ETH*

We have proved that SAT(S) is subexponential if and only if SAT(S)-DEG-B is subexponential for all B. This result is inconclusive since it does not rule out the possibility that a SAT(S)-DEG-B problem is subexponential and NP-complete for some B, but that the subexponential property is lost for larger values. Hence, it would be interesting to (dis)prove that SAT(S) is subexponential if and only if there exists some B such that SAT(S)-DEG-B is NP-complete and subexponential. This holds for SAT($\{R_{1/3}^{\neq\neq\neq}\}$) so it does not seem impossible that the result holds for all constraint 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.* [11, 14]. A particularly interesting example is Marx's [27] 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.

Constraint satisfaction problems

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 generalization would give us a tool for studying problems such as k-colorability and its many variations. Lifting the results to infinite-domain constraints appears to be more difficult, but it may be worthwhile: Bodirsky and Grohe [3] 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 generalization is to study problems that are not satisfiability problems, *e.g.*, enumeration problems, counting problems, and nonmonotonic reasoning problems such as abduction and inference under circumscription.

Acknowledgments

We thank Magnus Wahlström for helpful discussions on the topic of this article, and in particular for suggesting the proof strategy in Lemma 17.

Bruno Zanuttini is supported by grant ANR-10-BLAN-0210 (TUPLES) from the French National Research Agency (ANR). Peter Jonsson is partially supported by the Swedish Research Council (VR) under grants 2009-4431 and 2012-3239. Victor Lagerkvist is partially supported by the National Graduate School in Computer Science (CUGS), Sweden, the Swedish Research Council (VR) under grant 2008-4675, and the DFG-funded project "Homogene Strukturen, Bedingungserfüllungsprobleme, und topologische Klone" (Project number 622397).

References

[1] V. B. Alekseev and A. A. Voronenko, *On some closed classes in partial two-valued logic*, Discrete Mathematics and Applications, 4 (1994), pp. 401–419.

[2] E. Allender, M. Bauland, N. Immerman, H. Schnoor, and H. Vollmer, *The complexity of satisfiability problems: Refining Schaefer's theorem*, in Mathematical Foundations of Computer Science 2005, J. Jędrzejowicz and A. Szepietowski, eds., vol. 3618 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 71–82.

[3] M. Bodirsky and M. Grohe, *Non-dichotomies in constraint satisfaction complexity*, in Proceedings of the 35th international colloquium on Automata,

Languages and Programming, Part II, ICALP 2008, Berlin, Heidelberg, 2008, Springer-Verlag, pp. 184–196.

[4] V. G. Bodnarchuk, L. A. Kaluzhnin, V. N. Kotov, and B. A. Romov, *Galois theory for Post algebras. I*, Cybernetics and Systems Analysis, 5 (1969), pp. 243–252.

[5] ——, *Galois theory for Post algebras. II*, Cybernetics and Systems Analysis, 5 (1969), pp. 531–539.

[6] 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).

[7] F. Börner, *Basics of Galois connections*, in Complexity of Constraints, N. Creignou, P. G. Kolaitis, and H. Vollmer, eds., vol. 5250 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 38–67.

[8] A. A. Bulatov, P. Jeavons, and A. A. Krokhin, *Classifying the complexity of constraints using finite algebras*, SIAM Journal on Computing, 34 (2005), pp. 720–742.

[9] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström, *On problems as hard as CNF-SAT*, in Proceedings of the 2012 IEEE Conference on Computational Complexity, CCC 2012, Washington, DC, USA, 2012, IEEE Computer Society, pp. 74–84.

[10] V. Dalmau and D. Ford, *Generalized satisfiability with limited occurrences per variable: A study through delta-matroid parity*, in Mathematical Foundations of Computer Science 2003, Branislav Rovan and Peter Vojtáš, eds., vol. 2747 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2003, pp. 358–367.

[11] V. Dalmau, Ph. G.. Kolaitis, and M. Y.. Vardi, *Constraint satisfaction, bounded treewidth, and finite-variable logics*, in Principles and

Practice of Constraint Programming - CP 2002, Pascal Van Hentenryck, ed., vol. 2470 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 310–326.

[12] H. Dell and D. van Melkebeek, *Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses*, in Proceedings of the 42nd ACM symposium on Theory of computing, STOC 2010, New York, NY, USA, 2010, ACM, pp. 251–260.

[13] D. Geiger, *Closed systems of functions and predicates*, Pacific Journal of Mathematics, 27 (1968), pp. 95–100.

[14] M. Grohe, *The complexity of homomorphism and constraint satisfaction problems seen from the other side*, Journal of the ACM, 54 (2007), pp. 1–24.

[15] L. Haddad, *Infinite chains of partial clones containing all selfdual monotonic partial functions*, Multiple-Valued Logic and Soft Computing, 18 (2012), pp. 139–152.

[16] T. Hertli, *3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general*, SIAM Journal on Computing, 43 (2014), pp. 718–729.

[17] R. Impagliazzo and R. Paturi, *On the complexity of k-SAT*, Journal of Computer and System Sciences, 62 (2001), pp. 367 – 375.

[18] R. Impagliazzo, R. Paturi, and F. Zane, *Which problems have strongly exponential complexity?*, Journal of Computer and System Sciences, 63 (2001), pp. 512–530.

[19] G. I. Istrate, *Looking for a version of Schaefer's dichotomy theorem when each variable occurs at most twice*, tech. report, Rochester, NY, USA, 1997.

[20] P. Jeavons, *On the algebraic structure of combinatorial problems*, Theoretical Computer Science, 200 (1998), pp. 185–204.

[21] P. Jonsson, A. Krokhin, and F. Kuivinen, *Hard constraint satisfaction problems have hard gaps at location 1*, Theoretical Computer Science, 410 (2009), pp. 3856–3874.

[22] R. E. Ladner, *On the structure of polynomial time reducibility*, Journal of the ACM, 22 (1975), pp. 155–171.

[23] V. Lagerkvist, *Weak bases of Boolean co-clones*, Information Processing Letters, 114 (2014), pp. 462–468.

[24] V. Lagerkvist and M. Wahlström, *Polynomially closed co-clones*, in Proceedings of the 2014 IEEE 44th International Symposium on Multiple-Valued Logic, ISMVL 2014, Washington, DC, USA, 2014, IEEE Computer Society, pp. 85–90.

[25] D. Lau, *Function Algebras on Finite Sets*, Springer, Berlin, 2006.

[26] D. Lokshtanov, D. Marx, and S. Saurabh, *Lower bounds based on the exponential time hypothesis*, Bulletin of the EATCS, 105 (2011), pp. 41–72.

[27] D. Marx, *Can you beat treewidth?*, Theory of Computing, 6 (2010), pp. 85–112.

[28] C. Moore and J. M. Robson, *Hard tiling problems with simple tiles*, Discrete & Computational Geometry, 26 (2001), pp. 573–590.

[29] R. A. Moser and D. Scheder, *A full derandomization of Schoening's k-SAT algorithm*, in Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing, STOC 2011, New York, NY, USA, 2011, ACM, pp. 245–252.

[30] G. Nordh and B. Zanuttini, *Frozen Boolean partial co-clones*, in Proceedings of the 39th International Symposium on Multiple-Valued Logic, ISMVL 2009, May 2009, pp. 120 –125.

[31] E. Post, *The two-valued iterative systems of mathematical logic*, Annals of Mathematical Studies, 5 (1941), pp. 1–122.

[32] B. A. Romov, *The algebras of partial functions and their invariants*, Cybernetics and Systems Analysis, 17 (1981), pp. 157–167.

[33] R. Santhanam and S. Srinivasan, *On the limits of sparsification*, in Automata, Languages, and Programming, A. Czumaj, K. Mehlhorn, A. Pitts, and R. Wattenhofer, eds., vol. 7391 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 774–785.

[34] T. J. Schaefer, *The complexity of satisfiability problems*, in Proceedings of the tenth annual ACM symposium on Theory of computing, STOC 1978, New York, NY, USA, 1978, ACM, pp. 216–226.

[35] H. Schnoor and I. Schnoor, *New algebraic tools for constraint satisfaction*, in Complexity of Constraints, Nadia Creignou, Phokion Kolaitis, and Heribert Vollmer, eds., no. 06401 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[36] I. Schnoor, *The weak base method for constraint satisfaction*, PhD thesis, Gottfried Wilhelm Leibniz Universität, Hannover, Germany, 2008.

[37] K. Schölzel, *Galois theory for partial clones and some relational clones*, in Proceedings of the 41st IEEE International Symposium on Multiple-Valued Logic, ISMVL 2011, 2011, pp. 187–192.

[38] K. Schölzel, *Dichotomy on intervals of strong partial Boolean clones*, To appear in Algebra Universalis, (2014).

[39] U. Schöning, *A low and a high hierarchy within NP*, Journal of Computer and System Sciences, 27 (1983), pp. 14 – 28.

[40] Á. Szendrei, *Clones in Universal Algebra*, vol. 99 of *Seminaires de Mathématiques Supérieures*, University of Montreal, 1986.

[41] J. van Rooij, M. van Kooten Niekerk, and H. Bodlaender, *Partition into triangles on bounded degree graphs*, in SOFSEM 2011: Theory

and Practice of Computer Science, Ivana Cerná, Tibor Gyimóthy, Juraj Hromkovic, Keith Jefferey, Rastislav Královic, Marko Vukolic, and Stefan Wolf, eds., vol. 6543 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2011, pp. 558–569.

[42] M. Wahlström, *Algorithms, measures and upper bounds for satisfiability and related problems*, PhD thesis, Linköping University, TCSLAB - Theoretical Computer Science Laboratory, The Institute of Technology, 2007.

[43] G. J. Woeginger, *Exact algorithms for NP-hard problems: A survey*, in Combinatorial Optimization - Eureka, You Shrink!, vol. 2570 of Lecture Notes in Computer Science, 2003, pp. 185–208.