The Complexity of Symmetry Breaking Beyond Lex-Leader

Markus Anders
TU Darmstadt, Germany

Sofia Brenner
TU Darmstadt, Germany

Gaurav Rattan
University of Twente, Enschede, The Netherlands

Abstract

Symmetry breaking is a widely popular approach to enhance solvers in constraint programming, such as those for SAT or MIP. Symmetry breaking predicates (SBPs) typically impose an order on variables and single out the lexicographic leader (lex-leader) in each orbit of assignments. Although it is NP-hard to find complete lex-leader SBPs, incomplete lex-leader SBPs are widely used in practice.

In this paper, we investigate the complexity of computing complete SBPs, lex-leader or otherwise, for SAT. Our main result proves a natural barrier for efficiently computing SBPs: efficient certification of graph non-isomorphism. Our results explain the difficulty of obtaining short SBPs for important CP problems, such as matrix-models with row-column symmetries and graph generation problems. Our results hold even when SBPs are allowed to introduce additional variables. We show polynomial upper bounds for breaking certain symmetry groups, namely automorphism groups of trees and wreath products of groups with efficient SBPs.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases symmetry breaking, boolean satisfiability, matrix models, graph isomorphism

Digital Object Identifier 10.4230/LIPIcs.CP.2024.3

Related Version *Preprint*: https://arxiv.org/abs/2407.04419

Funding The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (EngageS: grant agreement No. 820148).
Sofia Brenner: Received funding from the German Research Foundation DFG (SFB-TRR 195 "Symbolic Tools in Mathematics and their Application").

1 Introduction

The search space of a constraint program can exhibit a large amount of symmetry. This simple yet far-reaching observation forms the core principle behind the use of *symmetry based* approaches in the realm of constraint programming [23, 44]. Such methods prune the symmetric parts of the search space to save computational costs. Ideally, they ensure that at most one solution exists per equivalence class of candidate solutions. Over the last two decades, numerous methods have been proposed to exploit symmetries of constraint programs. In particular, many approaches have been developed for Boolean satisfiability solvers [14, 1, 13, 17, 27, 30, 16, 43, 38] as well as mixed integer programming [35, 39, 40]. Symmetry-based solving remains an active and fruitful area of interest, especially from a practical perspective: for example, the defining feature of arguably one of the most successful entries in the SAT competition 2023 was symmetry breaking [11, 10].

30th International Conference on Principles and Practice of Constraint Programming (CP 2024).
Editor: Paul Shaw; Article No. 3; pp. 3:1–3:24
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

How symmetries should be used best remains unclear. Approaches can be roughly divided into two different categories: in *dynamic* and *static* approaches. In a *dynamic* approach, symmetries are used during the execution of a solver [43, 18, 16, 38]. A typical example is that the solver incorporates a branching rule that makes use of the symmetries directly [43].

The second approach is the *static* use of symmetries, which is the main focus of this paper. Here, additional constraints, so-called *symmetry breaking predicates* (SBPs), are added to a given problem instance. The notion of SBPs was first introduced in the seminal paper of Crawford, Ginsberg, Luks and Roy [14]. Their goal was to generate polynomial-sized SBPs for SAT formulas in conjunctive normal form (CNF). However, since their framework is rooted in group theory, many results neatly generalize to other constraint languages.

The framework of Crawford et al., as well as the majority of the subsequent work in this area, uses so-called *lex-leader* predicates to achieve complete symmetry breaking. Using *incomplete* lex-leader predicates is arguably one of the most successful approaches to symmetry breaking in practice [1, 17]. On a complexity-theoretic level, however, Crawford et al. proved that computing a predicate true of only the lex-leader in each equivalence class of Boolean assignments is NP-hard. Subsequent results showed that this even holds true for restricted classes of groups [34], as well as orders similar to lex-leader [29, 46].

One may wonder whether there are *other kinds* of SBPs that are efficiently computable. Here, *other kinds* of SBPs simply means that they do *not* make use of a lexicographic ordering of the assignments. In principle, choosing any canonical representative among symmetric assignments is permissible, lex-leader or otherwise. This question is motivated, for instance, by the realm of graph isomorphism (GI). There, choosing the lex-leader is also known to be NP-hard [7], and the best theoretical and practical approaches make use of other mechanisms.

Concerning practical symmetry breaking, only a few, though surprisingly different, approaches of generating non-lex-leader SBPs have been explored. In [22], the global cardinality constraint [42] is used in conjunction with lex-leader constraints to efficiently handle (particular) wreath symmetry. In [13], SAT symmetry breaking constraints for graph problems are produced similarly to the canonical labeling algorithm NAUTY [37]. In [25], minimal SAT symmetry breaking constraints are generated for small groups.

In general, however, the complexity of SBPs remains largely unexplored, even for fairly restricted kinds of symmetries. Perhaps the most glaring example is the problem of breaking *row-column symmetries*, which arise in the so-called *matrix models* [20]. These models allow the decision variables to be arranged in a matrix such that interchanging any two rows or any two columns is a symmetry of the model. Matrix models arise in multiple areas of constraint programming such as scheduling, combinatorial problems, and design [21]. Perhaps the most well-known matrix model is the pigeonhole principle problem, for which it is NP-hard to compute the lex-leader or similar assignments [14]. While the problem of devising SBPs for such models has received much attention [20, 29, 23], the known results do not explain the lack of compact SBPs for matrix models.

Our Results

The objective of this paper is to further investigate the exact complexity of computing static symmetry breaking predicates. *Given a group of symmetries on the variables of a formula, how hard is it to generate a complete symmetry breaking predicate?* The ultimate goal of our work is to obtain a classification of symmetry groups, in terms of the complexity of computing SBPs. Such a classification could help inform practitioners as to which cases can be handled easily, and which ones are more challenging.

In order to simplify the exposition, our setting of choice is that of Boolean satisfiability testing (SAT). However, in the same vein as [14], our results are founded in a general group-theoretic setting, so they should easily transfer to many branches of constraint programming: we consider computing symmetry breaking predicates for a given *permutation group*, instead of a particular SAT formula exhibiting such symmetry.

Our results can be divided into *hardness results* and *upper bounds.* The high-level idea for the hardness results can be summarized as follows: We show that if symmetry breaking is feasible for certain expressive groups, such as matrix groups or Johnson actions, then graph isomorphism is in coNP. The containment GI $\in$ co-NP is a major unresolved problem [31], even for the restricted case of group isomorphism [5]. While GI $\in$ co-NP seems to have no other major complexity theoretic consequences and is seemingly not "implausible", it still poses a barrier to compact SBPs.

The idea of our reductions is to encode the input graphs as binary strings in a suitable way, then guess a canonizing permutation, and use the symmetry-breaking constraint to verify that the result is indeed the canonical form: By definition, the symmetry-breaking constraint is true of precisely the canonical forms. The graphs are non-isomorphic exactly when the canonical forms are different. As a strengthening, we show that this holds even when the symmetry breaking constraint uses additional variables as their values can be guessed as well.

We now explain our hardness results in greater detail.

Matrix Models. Our first result tackles the difficulty of breaking row-column symmetries in matrix models. As mentioned above, this problem has received much attention in symmetry breaking literature.

▶ **Theorem 1.** *Suppose there exists a polynomial time algorithm for generating complete symmetry breaking predicates for row-column symmetries. Then* GI $\in$ co-NP *holds, i.e., graph non-isomorphism admits a non-deterministic polynomial time algorithm.*

Our theorem explains the difficulty of obtaining compact symmetry breaking predicates for matrix models, in the sense that it would imply polynomial time algorithms for certifying graph non-isomorphism. Section 3.1 contains a detailed description of our result.

Johnson Actions. We identify yet another class of groups for which symmetry breaking is hard, namely the (k, t)-*Johnson groups.* These are symmetric groups $\mathrm{Sym}(k)$ acting on t-subsets of $[k]$ for fixed $t < k$. It is well-known that these actions form an important sub-case of Babai's quasi-polynomial algorithm for graph isomorphism [5].

▶ **Theorem 2.** *Let $t > 1$ be a fixed positive integer. Suppose that we can generate complete symmetry breaking predicates for all (k, t)-Johnson groups in polynomial time (in terms of the domain size). Then* GI $\in$ coNP *holds.*

Section 4 contains a formal description of this result. In fact, it follows from Theorem 16, which proves a stronger statement.

Certificates for Canonization. We strengthen our hardness results of Theorem 1 and Theorem 2 as follows. We allow an algorithm to produce more expressive SBPs:

1. The SBP can be given as a Boolean circuit.
2. The SBP is allowed to *introduce additional variables.* Essentially, this gives the predicate access to additional non-determinism. The SBP may introduce an arbitrary number of additional variables, as long as the overall size is polynomial.

Despite allowing more powerful SBPs, we conclude a stronger hardness implication: an efficient algorithm for such predicates implies an *efficiently verifiable graph canonical form* (see Theorem 13 and Theorem 15). Note that an efficient verifier for graph canonical forms implies an efficient verifier for graph non-isomorphism (Lemma 12), but the converse is unknown. For this result, we observe that SBPs for a permutation group G on the domain $[n]$ essentially solve a particular *decision version* of the *string canonization* problem w.r.t. G on strings of length n. String canonization is a fundamental problem of interest in the graph isomorphism community [7, 5, 6]. Section 3.2 contains a detailed description. Moreover, we prove that the hardness results also hold for all subgroups of polynomial index (see Lemma 18).

Quasi-Polynomial Upper Bound. Realizing that symmetry breaking reduces to string canonization allows us to express an upper-bound on the size of *circuit* SBPs for general permutation groups. The result is mainly of theoretical interest, but we believe that this could have useful consequences in SBP heuristic design. The theorem immediately follows from the quasi-polynomial time algorithm of Babai [6], see Section 3.2 for more details.

▶ **Theorem 3.** *Given a permutation group $G \leq \mathrm{Sym}(n)$, there is a quasi-polynomial time (in n) algorithm producing a complete symmetry breaking circuit of quasi-polynomial size.*

We complement these results concerning *hard* families by focusing on polynomial upper bounds, i.e., the question for which families of groups symmetry breaking is *easy*:

Polynomial Upper Bounds. In Section 5, we examine how group-theoretic structure can help to design SBPs. Our results show how we can assemble SBPs for a group from the SBPs of its constituents, in context of natural operations such as disjoint direct products and wreath products. This extends the results of [24], where the existence of *lex-leader* constraints for constituents is assumed to assemble constraints for direct products and wreath products. (The paper also treats cyclic, dihedral and alternating groups.)

The following theorem is the main consequence of our results in this section.

▶ **Theorem 4.** *Assume that $G \leq \mathrm{Sym}(n)$ is the automorphism group of a tree T. Then G admits a complete symmetry breaking predicate of linear size. Given the tree T, it can be computed in polynomial time.*

Automorphism groups of trees are special cases of so-called wreath products. Such groups naturally occur, for example, whenever models exhibit hierarchical structure. Intuitively, the structure can be split into parts with the same symmetry group (the *base group*), which are permuted by the so-called *top group.* Essentially, we combine symmetry breaking constraints for the base group and the top group to a symmetry breaking constraint for the wreath product by using the predicate for base group to make every part canonical, and the constraint of the top group to fix an ordering of the parts. For the general case of wreath products the problem is far more technical, but we obtain the following result (see Section 5 for details).

▶ **Theorem 5.** *Let $G \leq \mathrm{Sym}(n)$ and $H \leq \mathrm{Sym}(m)$ be permutation groups. Assume that a complete symmetry breaking circuit for G can be computed in polynomial time. Moreover, suppose that for every partition P of $[m]$, the partition stabilizer S of P in H and a complete symmetry breaking circuit for S can be computed in polynomial time. Then there is a complete symmetry breaking circuit for the wreath product $W := G \wr H$ that can be computed in polynomial time.*

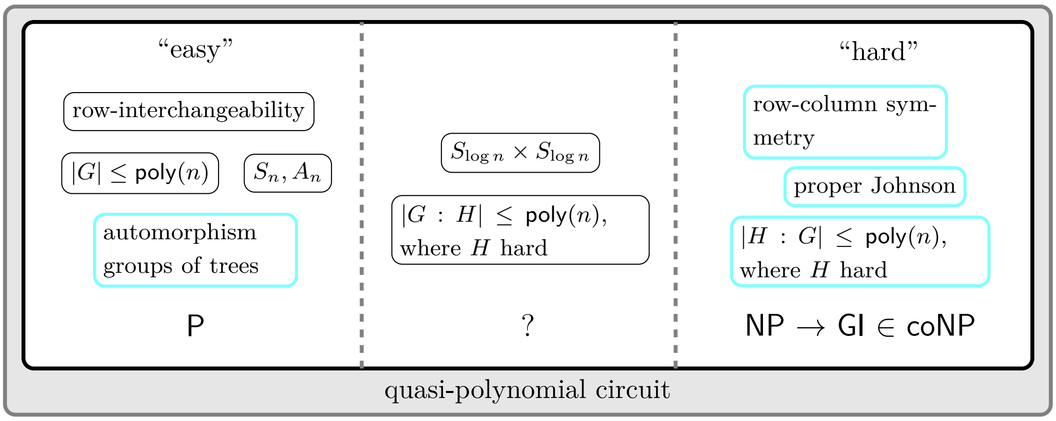
"easy"
row-interchangeability
$|G| \leq \mathsf{poly}(n)$
S_n, A_n
automorphism groups of trees
P
$S_{\log n} \times S_{\log n}$
$|G : H| \leq \mathsf{poly}(n)$, where H hard
?
"hard"
row-column symmetry
proper Johnson
$|H : G| \leq \mathsf{poly}(n)$, where H hard
NP → GI ∈ coNP
quasi-polynomial circuit

Figure 1 Complexity of computing symmetry breaking predicates for the stated families of groups in SAT. All groups can be handled in quasi-polynomial time using a circuit. The symbol G refers to the permutation group of consideration. The parameter n refers to the domain size of the permutation group, or, the number of variables of the formula. For "easy" families of groups, a CNF predicate can be computed in polynomial time. For "hard" families of groups, the existence of polynomial time symmetry breaking, even allowing the use of additional variables, implies that GI is in coNP. Blue outlines indicate novel results proven in this paper.

In summary, Figure 1 provides a concise description of our progress towards a complexity classification for the problem of generating SBPs for permutation groups.

2 Preliminaries

2.1 Boolean Circuits and Satisfiability

Boolean Circuits. A Boolean circuit ψ is a circuit consisting of input gates, one output gate, and $\{\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}\}$-gates connecting them in the usual way. We refer to the input gates as the *variables* $\mathrm{Var}(\psi)$. The *size* of a circuit refers to the number of gates.

An *assignment* of ψ is a function $\theta\colon V \to \{0,1\}$ where $V \subseteq \mathrm{Var}(\psi)$. The assignment is *complete* whenever $V = \mathrm{Var}(\psi)$ and *partial* otherwise. A circuit is evaluated using an assignment $\theta\colon V \to \{0,1\}$, by replacing each input gate $v \in V$ with $\theta(v)$, with the usual meaning. The resulting circuit is $\psi[\theta]$. Whenever θ is complete, the value of the output gate can be determined in linear time, and hence either $\psi[\theta] = 0$ or $\psi[\theta] = 1$ holds.

If $\psi[\theta] = 1$ we call θ a *satisfying assignment*, whereas if $\psi[\theta] = 0$ we call θ a *conflicting assignment*. A circuit ψ is *satisfiable* if and only if there exists a satisfying assignment to ψ.

Conjunctive Normal Form. In practice, a SAT instance ψ is typically given in *conjunctive normal form* (CNF), which we denote with $\psi = \{\{l_{1,1} \vee \dots \vee l_{1,k_1}\} \wedge \dots \wedge \{l_{m,1} \vee \dots \vee l_{m,k_m}\}\}$. Each element $C \in \psi$ is called a *clause*, whereas a clause itself consists of a set of *literals*. A literal is either a variable v or its negation $\overline{v}$.

A symmetry, or *automorphism*, of ψ is a permutation of the variables $\varphi\colon \mathrm{Var}(\psi) \to \mathrm{Var}(\psi)$ which maps ψ back to itself, i.e., $\psi^\varphi \equiv \psi$, where φ is applied element-wise to the variables in each clause. The permutation group of all symmetries of ψ is $\mathrm{Aut}(\psi) \leq \mathrm{Sym}(\mathrm{Var}(\psi))$.

Another common way to define symmetries is to define them on the *literals* of the formula, allowing the use of so-called *negation symmetries* (see [44]). In any case, symmetries can be efficiently computed in practice using state-of-the-art symmetry detection tools [37, 28, 15, 2].

2.2 Permutation Groups

We briefly introduce some notation and results for permutation groups. For further background material on permutation groups, we refer to [19]. Throughout, we use the notation $[n] := \{1, \ldots, n\}$ for $n \in \mathbb{Z}_{>0}$ and set $[0] := \emptyset$.

Let Ω be a nonempty finite set. Let $\mathrm{Sym}(\Omega)$ denote the *symmetric group* on Ω, i.e., the group of permutations of Ω. A *permutation group* is a subgroup G of $\mathrm{Sym}(\Omega)$, denoted by $G \leq \mathrm{Sym}(\Omega)$. We also say that G *acts on* Ω. A permutation group is always specified by the abstract isomorphism type of G (for instance, G could be cyclic of order 10), together with the action of G on Ω. For $g \in G$ and $\omega \in \Omega$, we write ω^g for the image of ω under g and $\omega^G = \{\omega^g \colon g \in G\}$ for the *orbit* of ω under G. The *support* of G consists of those elements in Ω that are moved (i.e., not fixed) by some element of G. For a partition $P = (\Omega_1, \ldots, \Omega_r)$ of Ω (i.e., $\Omega = \Omega_1 \dot\cup \cdots \dot\cup \Omega_r$), the *partition stabilizer* of P in G consists of all elements $g \in G$ that setwise stabilize $\Omega_1, \ldots, \Omega_r$, i.e. for all $i \in [r]$, $\{\omega^g : \omega \in \Omega_i\} = \Omega_i$. The *index* of a subgroup H of G is $|G : H| := |G|/|H|$.

Two permutation groups $G \leq \mathrm{Sym}(\Omega)$ and $H \leq \mathrm{Sym}(\Delta)$ are *permutation isomorphic* if there exists a bijection $\lambda\colon \Omega \to \Delta$ and a group isomorphism $\alpha\colon G \to H$ such that $\lambda(\omega^g) = \lambda(\omega)^{\alpha(g)}$ for all $\omega \in \Omega$ and $g \in G$. Note that this notion is stronger than G and H being isomorphic (as abstract groups) as the same abstract group can give rise to different group actions. For instance, $\mathrm{Sym}(k)$ admits so-called Johnson actions on different domains:

Johnson Groups. Let k be a positive integer and $t \in [k-1]$. A permutation $\pi \in \mathrm{Sym}(k)$ induces a permutation $\hat\pi$ on the domain $\binom{[k]}{t}$ of t-subsets of $[k]$, mapping a t-subset A to $A^{\hat\pi} = \{a^\pi \colon a \in A\}$. This way, $\mathrm{Sym}(k)$ becomes a permutation group $S_k^{(t)}$ on a domain of size $\binom{k}{t}$. The groups $S_k^{(t)}$ are called *Johnson groups* and the action is called a *Johnson action*. We call a Johnson group *proper* if $t \notin \{1, k-1\}$ holds.

Usually, the analogous action of the so-called alternating groups is also called a Johnson action. Due to our results in Section 4.3, it suffices to only consider the symmetric groups.

Wreath products. Let $G \leq \mathrm{Sym}(\Omega)$ and $H \leq \mathrm{Sym}(\Delta)$ be permutation groups. The *wreath product* $G \wr H$ consists of the set $G^\Delta \times H$, endowed with the multiplication rule $\big((g_\delta)_{\delta\in\Delta}, h\big)\big((g'_\delta)_{\delta\in\Delta}, h'\big) = \big((g_\delta g'_{\delta^{h^{-1}}})_{\delta\in\Delta}, hh'\big)$. We call G the *base group* and H the *top group*. The group $G \wr H$ acts on $\Omega \times \Delta$ by $(\omega, \delta)^{((g_\delta)_{\delta\in\Delta}, h)} = (\omega^{g_{\delta^h}}, \delta^h)$. This action is called the *imprimitive action* of the wreath product.

2.3 Graph Isomorphism and String Canonization

Graphs. A finite, undirected graph $\Gamma = (V, E)$ consists of a set of vertices $V \subseteq \mathbb{N}$ and an edge relation $E \subseteq \binom{V}{2}$. Unless stated otherwise, the set of vertices V is $\{1, \ldots, n\}$ and $m := |E|$ denotes the number of edges. We may refer to the set of vertices of Γ with $V(\Gamma)$, and to the set of edges with $E(\Gamma)$. The *adjacency matrix* of Γ is the $n \times n$-matrix $A = (a_{ij})$ with $a_{ij} = 1$ if $\{i, j\} \in E(\Gamma)$, and $a_{ij} = 0$ otherwise. Unless stated otherwise, we assume our graphs are given as adjacency matrices.

A graph Γ is *bipartite* if $V(\Gamma) = A \dot\cup B$ can be partitioned into two independent sets $A = \{a_1, \ldots, a_k\}$ and $B = \{b_1, \ldots, b_\ell\}$. In this case, we may obtain an *bipartite adjacency matrix* $M = (m_{ij})$ by setting $m_{ij} = 1$ if a_i and b_j are adjacent, and $m_{ij} = 0$ otherwise.

Lexicographic ordering. For $\{0,1\}$-strings θ, θ' of the same length, we write $\theta \preceq_{\text{lex}} \theta'$ if θ is smaller or equal to θ' with respect to the lexicographic ordering. Likewise, we define a lexicographic ordering of matrices with entries in $\{0,1\}$ of a fixed size by interpreting them as strings, reading them row by row.

Relational Structures. As a generalization of graphs, we define a *t-ary relational structure* $R = (U, A)$, where U is a universe and A is a t-ary relation on U. A t-ary relational structure is *symmetric* if for every t-tuple $(u_1, \dots, u_t) \in A$ and for every $\sigma \in \mathrm{Sym}(t)$, it holds that $(u_{\sigma(1)}, \dots, u_{\sigma(t)}) \in A$.

Graph Isomorphism. Two graphs $\Gamma_1 = (V_1, E_1), \Gamma_2 = (V_2, E_2)$ are said to be *isomorphic*, whenever there exists a bijection $\varphi\colon V_1 \to V_2$ such that $\varphi(\Gamma_1) = (V_1^{\varphi}, E_1^{\varphi}) = (V_2, E_2) = \Gamma_2$ holds. Here, V_1^{φ} and E_1^{φ} means applying φ element-wise to each element in V_1, and each element of each tuple in E_1, respectively. We call φ an *isomorphism* between Γ_1 and Γ_2. We may write $\Gamma_1 \cong \Gamma_2$ to denote isomorphism. A corresponding computational problem follows:

▶ **Problem 6** (GI)**.** *Given two graphs Γ_1, Γ_2, does $\Gamma_1 \cong \Gamma_2$ hold?*

Regarding certification, it is easy to see that GI is in NP. On the other hand, graph isomorphism is known to be in coAM, i.e., there are efficient randomized proofs for non-isomorphism [9]. As mentioned in the introduction, whether graph isomorphism is in coNP is a long-standing open problem [31].

Analogously, we may define isomorphism for t-ary relational structures $R_1 = (U_1, A_1)$ and $R_2 = (U_2, A_2)$: R_1 and R_2 are *isomorphic* if there exists a bijection $\pi\colon U_1 \to U_2$ such that for every $(u_1, \dots, u_t) \in A_1$, it holds that $(u_1^{\pi}, \dots, u_t^{\pi}) \in A_2$ and vice-versa.

String Canonization. We next define the *string canonization* problem [7, 32]. The string canonization problem asks, given a permutation group $G \le \mathrm{Sym}(\Omega)$ and a string $\sigma\colon \Omega \to \Sigma$ on a finite alphabet Σ, for a canonical representative of σ^G. In particular, it computes a function $F\colon \mathcal{G} \times \Sigma^{\Omega} \to \Sigma^{\Omega}$ where $\mathcal{G}$ denotes the set of all permutation groups $G \le \mathrm{Sym}(\Omega)$, and for all $\sigma_1, \sigma_2 \in \Sigma^{\Omega}$ it holds that (1) $F(G, \sigma_1) \cong_G \sigma_1$ and (2) if $\sigma_1 \cong_G \sigma_2$ then $F(G, \sigma_1) = F(G, \sigma_2)$. Here, $\cong_G$ means that σ_1 can be permuted to σ_2 using an element of G. A corresponding computational problem follows:

▶ **Problem 7** (s-SCANON$_F$)**.** *Given a permutation group $G \le \mathrm{Sym}(\Omega)$, a finite alphabet Σ and a string $\sigma \in \Sigma^{\Omega}$, compute the canonical representative $F(G, \sigma)$.*

The graph isomorphism problem polynomial time reduces to s-SCANON, but the converse is unknown. However, there is an F such that there is a quasi-polynomial time algorithm which solves the string canonization problem [6]. It turns out that the string canonization problem is intimately related to symmetry breaking, which we discuss thoroughly in Section 3.2.

A crucial special case of string canonization is *graph canonization*. As the name suggests, it computes canonical forms for graphs. Let f be a graph canonization function, i.e., for graphs Γ, Δ, it holds that (1) $\Gamma \cong \Delta$ iff $f(\Gamma) = f(\Delta)$, and, (2) $f(\Gamma) \cong \Gamma$. Here, the symbol $\cong$ denotes the graph isomorphism relation. The corresponding computational problem follows:

▶ **Problem 8** (s-GCANON$_f$)**.** *Given a graph Γ, compute the canonical representative $f(\Gamma)$ within the isomorphism class of Γ.*

Indeed, this problem is a special case of string canonization: G can be chosen as a Johnson group of appropriate order and the strings encode the given graphs (see [6]).

2.4 Notions of Symmetry Breaking

Next, we define our notions of symmetry breaking. Let ψ be a CNF formula. Typically, symmetry breaking is defined specifically for the automorphism group $\mathrm{Aut}(\psi)$ of ψ. However, it turns out that often, our symmetry breaking predicates only depend on the structure of $\mathrm{Aut}(\psi)$ and its action on the set of variables $\mathrm{Var}(\psi)$. In particular, they do not depend on the specific shape of the formula ψ. Hence, we define symmetry breaking only using an arbitrary permutation group $G \leq \mathrm{Sym}(\Omega)$ and without referring to a precise formula ψ.

Symmetry Breaking Constraints. We begin with a discussion of complete symmetry breaking. Indeed, we find that in the literature two different notions are in use.

The first of these notions is what we will refer to simply as *complete symmetry breaking*. The idea is that a complete symmetry breaking constraint must ensure that in each orbit of *complete* assignments, all but one canonical representative is conflicting [14].

Formally, we let $\theta_{\text{full}}(\Omega) := \{\theta \mid \theta \colon \Omega \to \{0,1\}\}$ denote the set of all complete assignments to Ω. We let $G \leq \mathrm{Sym}(\Omega)$ act on $\theta_{\text{full}}(\Omega)$ in the natural way. A Boolean circuit ψ with $\mathrm{Var}(\psi) \subseteq \Omega$ is called a *complete* symmetry breaking circuit for G, whenever for each orbit $O \subseteq \theta_{\text{full}}(\Omega)$ under G, there is

- a $\tau \in O$ such that $\psi[\tau]$ is satisfying,
- for all $\tau' \in O$ with $\tau \neq \tau'$ the formula $\psi[\tau']$ is conflicting.

If ψ is restricted to be a CNF formula, we refer to ψ as a symmetry breaking *predicate*. This notion is typically used in the context of general-purpose symmetry breaking, such as for example in [14, 1, 17, 25]. We remark that in [25], this notion is referred to as an *isolator*.

The second notion in use in the literature is *isomorph-free generation*. It is usually considered in the realm of *dynamic* symmetry breaking. However, a notion for predicates can be defined: a predicate is supposed to ensure that in each orbit of *partial* assignments, all but one canonical representative is conflicting. Intuitively, isomorph-free generation ensures that no isomorphic branches are *ever* considered in the search. Isomorph-free generation immediately also ensures complete symmetry breaking. It is typically used in the context of generation tasks, such as in [36, 30], but it has also been considered for general-purpose symmetry breaking [27].

The focus of this paper is on complete symmetry breaking and not on isomorph-free generation.

Symmetry Breaking as a Computational Problem. We define a corresponding computational problem for symmetry breaking.

▶ **Problem 9** (Symmetry Breaking). *Given a permutation group $G \leq \mathrm{Sym}(\Omega)$, compute a complete symmetry breaking circuit for G.*

There are two variations of this problem that we discuss throughout the paper: the first of which concerns the group G. Usually, G is the automorphism group of a given CNF formula ψ, i.e., $G = \mathrm{Aut}(\psi)$. In this case, the problem might become easier, since automorphism groups and a given formula may admit further structural arguments. However, considering symmetry breaking for arbitrary permutation groups G opens up the possibility of using symmetries *beyond* syntactic ones, even though it might be unclear how they could be obtained. Furthermore, results are independent of the specific structure of SAT instances.

The second variation concerns the output: we may expect a CNF predicate, or a Boolean circuit. Computing a CNF predicate may be harder, since circuits are more expressive. We believe that all variations of the problem are of potential interest. Therefore, it seems best to attempt to use the problem definition which yields the strongest possible statement.

3 Row-Column Symmetries

In this section, we analyze the complexity of computing symmetry breaking predicates for row-column symmetry. Section 3.1 describes the hardness of obtaining SBPs for breaking row-column symmetries. In particular, we provide a proof of Theorem 1. Section 3.2 establishes the connection between symmetry breaking and decision string canonization. Lastly, in Section 3.3, we strengthen our results to work for circuit SBPs and SBPs with extra variables.

3.1 Hardness of Breaking Row-Column Symmetries

We begin with a formal definition of row-column symmetry.

Row-Column Symmetry. Let m, n be two positive integers, and $\Omega := [n] \times [m]$. The *row-column symmetry* group G is defined to be the group $\mathrm{Sym}([n]) \times \mathrm{Sym}([m])$, where $\mathrm{Sym}([n])$ naturally acts on the first component of Ω, and $\mathrm{Sym}([m])$ on the second component. Informally, we can view Ω as a *matrix* with n rows and m columns. The group $G \leq \mathrm{Sym}(\Omega)$ then consists of all the possible row transpositions and all possible column transpositions, along with their arbitrary compositions.

A *matrix model* is a constraint program whose decision variables can be arranged as a matrix above such that its automorphism group is the row-column symmetry group for this matrix arrangement. For such programs, it is typical to index their variable set by $\Omega = \{x_{ij} \mid i \in [n], j \in [m]\}$.

The following lemma states a one-to-one correspondence between assignments to a matrix model and bipartite graphs. Let $\Gamma(U, V)$ denote a bipartite graph with a designated left-partition U and a right-partition V, where U and V are non-interchangeable.

▶ **Lemma 10.** *There exists a one-to-one correspondence between the set of all Boolean assignments to the variables $\{x_{11}, \dots, x_{nm}\}$ of a matrix model and the set of all bipartite graphs $\Gamma([n], [m])$ with designated left and right partitions.*

Proof. Interpret the truth-value of x_{ij} as the indicator for whether there exists an edge between $i \in [n]$ and $j \in [m]$. ◀

We proceed with the proof of Theorem 1.

Proof of Theorem 1. We devise a polynomial time verifier for checking purported certificates for non-isomorphism, assuming that we can compute a row-column symmetry breaking predicate in polynomial time.

Bipartite Graphs Suffice. It will be more convenient for us to work with bipartite graphs instead of general graphs, in the spirit of standard reductions in isomorphism literature [47]. To every graph Γ, we can always associate a bipartite graph $\mathrm{bip}(\Gamma)$, namely the vertex-edge incidence graph as follows. The graph $\mathrm{bip}(\Gamma)$ has a designated *left* partition consisting of $V(\Gamma)$, a designated *right* partition consisting of $E(\Gamma)$, and the edges of $\mathrm{bip}(\Gamma)$ are defined by vertex-edge incidence. Moreover, $\mathrm{bip}(\Gamma)$ is a vertex-ordered graph: the left partition inherits the ordering from the graph Γ, and the right partition $E(\Gamma)$ is ordered according to the ordering induced by the ordering of $V(\Gamma)$. Observe that the mapping $\Gamma \mapsto \mathrm{bip}(\Gamma)$ is injective. Moreover, it is easy to verify that two graphs Γ and Δ are isomorphic if and only if the bipartite graphs $\mathrm{bip}(\Gamma)$ and $\mathrm{bip}(\Delta)$ are isomorphic via a bijection which maps the left-partition (right-partition) of $\mathrm{bip}(\Gamma)$ to the left-partition (right-partition) of $\mathrm{bip}(\Delta)$.

Therefore, it suffices to verify non-isomorphism certificates for bipartite graphs.

Certificate. Given two bipartite graphs Γ and Δ, our chosen certificate of non-isomorphism is a pair of bijections (σ, π), where $\sigma\colon V(\Gamma) \to V(\Gamma)$ and $\pi\colon V(\Delta) \to V(\Delta)$.

Verifier. Given such a certificate (σ, π), our polynomial time verifier proceeds as follows:

1. Compute a symmetry breaking predicate $\delta_{n,m}(x_{11}, \ldots, x_{nm})$ in time $\mathsf{poly}(n, m)$.
2. Check if both Γ^σ and Δ^π satisfy $\delta_{n,m}(x_{11}, \cdots, x_{nm})$, when viewed as Boolean assignments. If both of them satisfy $\delta_{n,m}$, continue; otherwise reject.
3. Check whether $\Gamma^\sigma \neq \Delta^\pi$, otherwise reject.
4. Declare Γ and Δ to be non-isomorphic.

It is easy to verify that all of the steps above are polynomial time computations.

Correctness of Verifier. It remains to be shown that (1) for every pair of non-isomorphic graphs, there exists a polynomial sized certificate accepted by the verifier above, and (2) for every pair of isomorphic graphs, the verifier always rejects any certificate.

For (1), let Γ and Δ be two non-isomorphic graphs. Let Γ^* be the unique satisfying assignment of $\delta_{n,m}$ in the orbit of Γ (similarly define Δ^*) under row-column symmetries. Let σ be an isomorphism from Γ to Γ^* (similarly define π). Since $\Gamma \ncong \Delta$, it must hold that Γ^* and Δ^* lie in different orbits, and hence $\Gamma^* \neq \Delta^*$. Since the certificate satisfies all conditions of the verifier, the verifier correctly certifies Γ and Δ to be non-isomorphic.

For (2), suppose Γ and Δ are isomorphic. Then they lie in the same orbit under the action of row-column symmetry on the Boolean assignments to the matrix model. Given any certificate (σ, π), the requirement of Γ^σ and Δ^π having to satisfy $\delta_{n,m}$ implies that $\Gamma^\sigma = \Delta^\pi$. But then such a certificate is rejected by the verifier in the third step. Hence, the verifier correctly refuses to certify that Γ and Δ are non-isomorphic. ◀

It is not clear whether the converse of Theorem 1 holds. In fact, even a P-time algorithm for graph isomorphism may not be sufficient to yield symmetry breaking algorithms for row-column symmetries. In what follows, we address this situation with a closer examination of the complexity of symmetry breaking.

3.2 A Decision Version of String Canonization

We now introduce a *decision variant* of the string canonization problem, which only decides whether a given string *is* the canonical string:

▶ **Problem 11** (d-SCANON$_F$). *Given a group $G \leq \mathrm{Sym}(\Omega)$, a finite alphabet Σ and a string $\sigma \in \Sigma^\Omega$, decide whether $\sigma = F(G, \sigma)$ holds, i.e., whether σ is the canonical representative within its isomorphism class σ^G.*

Let us consider a CNF formula ψ. We consider the case of the string canonization problem where $\Sigma = \{0, 1\}$ and the group $G \leq \mathrm{Sym}(\mathrm{Var}(\psi))$ consists of symmetries of ψ. Note that any two given assignments σ_1 and σ_2 of ψ can be interpreted as strings, and $\sigma_1 \cong_G \sigma_2$ holds if and only if they are in the same orbit of G.

We observe that an algorithm for d-SCANON accepts precisely one assignment per orbit of G. But this just means that if we translate such an algorithm into a Boolean circuit, the resulting circuit *is* a symmetry breaking circuit.

Clearly, d-SCANON$_F$ polynomial time reduces to s-SCANON$_F$. Since s-SCANON can be solved using a quasi-polynomial time algorithm [6], Theorem 3 follows immediately.

Analogously, we may define a decision version of the graph canonization problem, denoted as d-GCANON$_f$. Recall that graph canonization is a special case of string canonization. In the following, we prove that the decision canonization problem is tightly related to graph isomorphism in terms of its non-deterministic complexity.

▶ **Lemma 12.** *Let f be a canonical form such that d-GCANON$_f$ is in* NP*. Then,* $\mathsf{GI} \in \mathsf{coNP}$.

Proof. Assuming d-GCANON is in NP gives us access to a class of polynomial-sized certificates and a polynomial time verifier for these certificates, such that the following hold. If a given graph Γ is the canonical representative of its isomorphism class, then there must be a certificate σ such that the verifier accepts (Γ, σ). If Γ is not the canonical representative, then for all certificates σ the verifier rejects (Γ, σ).

Based on this, we provide a non-deterministic polynomial time algorithm for graph non-isomorphism of two graphs Γ and Δ.

Certificate. The certificate consists of two permutations $\varphi_1 \in \mathrm{Sym}(V(\Gamma)), \varphi_2 \in \mathrm{Sym}(V(\Delta))$, a certificate σ_1 for decision canonization of Γ, as well as σ_2 for decision canonization of Δ.

Verifier. Given two graphs Γ, Δ and certificate $(\varphi_1, \varphi_2, \sigma_1, \sigma_2)$, the verifier proceeds as follows. (Step 1) Run the decision canonization verifier for $(\Gamma^{\varphi_1}, \sigma_1)$ and $(\Delta^{\varphi_2}, \sigma_2)$. If both are accepted, proceed, otherwise reject. (Step 2) Accept if $\Gamma^{\varphi_1} \neq \Delta^{\varphi_2}$, otherwise reject.

Correctness of Verifier. Note that whenever we reach Step 2 of the verifier, the procedure guarantees that Γ^{φ_1} is a canonical form of Γ and Δ^{φ_2} of Δ. Hence, $\Gamma \cong \Delta$ holds if and only if $\Gamma^{\varphi_1} = \Delta^{\varphi_2}$ holds. It immediately follows that the algorithm accepts if and only if Γ and Δ are non-isomorphic. ◀

3.3 Hardness of Symmetry Breaking with Additional Variables

Consider the situation where one is allowed to use additional variables from a set Ω' to write down symmetry breaking constraints. In principle, this expands our domain Ω of variables used to $\Omega \,\dot\cup\, \Omega'$. Since the introduction of new variables Ω' changes the set of assignments, we need to adjust our definition of complete symmetry breaking.

Symmetry Breaking with Additional Variables. A Boolean circuit ψ is called a complete symmetry breaking circuit *with additional variables* for $G \leq \mathrm{Sym}(\Omega)$, whenever for each orbit $\tau \subseteq \sigma_{\mathrm{full}}(\Omega)$ under G, there is

- a $\tau' \in \tau$ such that $\psi[\tau']$ is satisfiable,
- for all $\tau'' \in \tau$ with $\tau' \neq \tau''$ the circuit $\psi[\tau'']$ is unsatisfiable.

A point of contention in the above definition might be whether $\psi[\tau']$ should actually have *exactly one* satisfying assignment. This would ensure that there is precisely one satisfying assignment per orbit, while our definition only suffices to ensure a unique satisfying assignment *when restricted* to the variables of ψ. In this paper, we stick to the above definition.

Using additional variables is typically not considered in the literature, most likely because this might substantially alter the difficulty of the underlying instance. Introducing additional variables is however intriguing: it gives the symmetry breaking predicates access to non-determinism, and hence might enable substantially more powerful constraints.

Hardness with Additional Variables. We show hardness results for symmetry breaking *even if* we are allowed to introduce new variables.

▶ **Theorem 13.** *Suppose there exists a polynomial time algorithm for generating complete symmetry breaking circuits* with additional variables *for row-column symmetries. Then, it holds that d-GCANON* $\in$ NP *and hence* $\mathsf{GI} \in \mathsf{coNP}$.

Proof sketch. It suffices to show that there exists a canonical form f such that d-GCANON$_f$ $\in$ NP (see Lemma 12). Again, we encode the input graph as a bipartite graph as in the proof of Theorem 1. The main argument follows by an inspection of the proof of Theorem 1: we observe that a certificate can also guess an assignment to the additional variables introduced by the SBP. We then simply verify that the adjacency matrix of the input graph *and* the assignment to the additional variables is accepted by the symmetry breaking circuit. ◀

The formal details of the proof can be found in Appendix A.

4 Johnson Actions

Next, we consider the so-called Johnson groups. Johnson groups are groups which naturally occur in problems encoding graph generation tasks [30]. We begin this section by describing a correspondence between Johnson groups and symmetric relational structures. Then, we provide a formal proof of Theorem 2. Lastly, we show how to derive SBPs for a group G, given SBPs for a small index subgroup $H \leq G$ (Lemma 18).

4.1 Johnson Groups and Relational Structures

Johnson Families. Let k be a positive integer. For $t \in [k-1]$, let X_k^t be the set of variables indexed by t-element subsets of $[k]$. In particular, we have $|X_k^t| = \binom{k}{t}$. For fixed $t \geq 1$, we call the group family $S_k^{(t)} \leq \mathrm{Sym}(X_k^t)$ the *Johnson family of arity t*.

Johnson groups form a subclass of the so-called groups of Cameron type. These groups as well as their natural action can be recognized in polynomial time (see [8]).

Relational Structures and Johnson Groups. To a symmetric t-ary relational structure R, we associate an assignment $f_R\colon X_k^t \to \{0,1\}$ with $f(x_S) = 1$ for a t-subset S of $[k]$ if S is a hyperedge in R, and $f(x_S) = 0$ otherwise. Conversely, given $f\colon X_k^t \to \{0,1\}$, we define a symmetric t-ary relational structure R_f on the universe $[k]$ whose relation is the set of all tuples $(a_1, \dots, a_t)$ with $f(\{a_1, \dots, a_t\}) = 1$.

This defines a one-to-one correspondence between assignments of X_k^t and symmetric t-ary relational structures. The following result formalizes the correspondence (see also [33]).

▶ **Lemma 14.** *Let R and R' be two symmetric t-ary relational structures on the universe $[k]$. Then R and R' are isomorphic if and only if the assignments f_R and $f_{R'}$ of the set X_k^t lie in the same orbit under the action of the Johnson group $S_k^{(t)} \leq \mathrm{Sym}(X_k^t)$.*

Proof. Suppose that R and R' are isomorphic via a bijection $\pi\colon [k] \to [k]$. Then, the induced action $\hat{\pi}$ on t-subsets of $[k]$ defines an element of $S_k^{(t)}$ with $f_{R'} = f_R^{\hat{\pi}}$. Conversely, suppose that $f_{R'} = f_R^{\hat{\pi}}$ for some $\hat{\pi}$ corresponding to the induced action of $\pi\colon [k] \to [k]$. It is easy to check that π is an isomorphism between R and R'. ◀

4.2 Johnson Families of Fixed Arity are Hard

In this section, we show that polynomial time symmetry breaking for Johnson families of fixed arity $t \geq 2$ implies GI $\in$ coNP.

▶ **Theorem 15.** *Suppose there exists a polynomial time algorithm for generating complete symmetry breaking circuits with additional variables for the Johnson family of arity 2. Then, d-GCANON $\in$ NP and hence GI $\in$ coNP.*

Proof. We again make use of Lemma 12, proving that polynomial time symmetry breaking circuits with additional variables for Johnson groups give rise to a non-deterministic polynomial time algorithm for decision graph canonization. Using similar arguments to Theorem 13, this follows from Lemma 14 and the fact that for two relational structures R, R' it holds that $R = R'$ if and only if $f_R = f_{R'}$. ◀

We generalize Theorem 15 to arbitrary arity.

▶ **Theorem 16.** *Let $t \geq 2$ be a fixed arity. Suppose there exists a polynomial time algorithm for generating complete symmetry breaking circuits with additional variables for the Johnson family of arity t. Then, d-GCANON $\in$ NP and hence GI $\in$ coNP.*

Proof sketch. Lemma 12 ensures that it suffices to prove d-GCANON $\in$ NP. By Lemma 14, it suffices to solve d-GCANON in non-deterministic polynomial time using a non-deterministic polynomial time oracle for decision canonization for uniform, symmetric t-ary relational structures. This is achieved by defining an isomorphism-invariant encoding of graphs into t-ary symmetric relational structures, essentially extending every graph edge to a t-ary relation by adding $t - 2$ bogus vertices. ◀

The remaining reduction is standard and can be found in Appendix C.

▶ Remark 17. In contrast, observe that the Johnson family for $t = 1$ consists of the symmetric groups $\mathrm{Sym}(n)$ in their natural action on n points. For these groups, complete symmetry breaking can be achieved with a CNF predicate of linear size (see Section 5).

4.3 Subgroups of Small Index and Large Primitive Groups

In this section, we consider symmetry breaking for a permutation group $G \leq \mathrm{Sym}(n)$ and a subgroup H of G. Mostly, we are interested in the case that H has polynomial index in G. We first show that a symmetry breaking constraint for H gives rise to symmetry breaking constraint for G:

▶ **Lemma 18.** *There exists a polynomial p such that the following holds: if there is a complete symmetry breaking circuit for a group $H \leq \mathrm{Sym}(n)$ which can be computed in time t, then complete symmetry breaking circuit with additional variables for $G \leq \mathrm{Sym}(n)$ with $G \geq H$ can be computed in time $t \cdot p(n|G : H|)$.*

Proof. Let ψ be a symmetry breaking circuit for H. We now devise a symmetry breaking circuit for G. For simplicity, we fix a system of representatives R of the right cosets of H in G, which can be computed in time polynomial in $|G : H|$ (see [26]).

Certificate. The certificate $\sigma = \{(\theta_r, h_r) : r \in R\}$ consists of assignments $\theta_r \colon \mathrm{Var}(\psi) \to \{0,1\}$ and an element $h_r \in H$ for every $r \in R$.

Verifier. Given an assignment $\theta \colon \mathrm{Var}(\psi) \to \{0,1\}$ and a certificate $\sigma = \{(\theta_r, h_r) : r \in R\}$, we proceed as follows:

1. For all $r \in R$, verify that $\theta_r^{h_r} = \theta^r$ holds.
2. For all $r \in R$, verify that $\psi[\theta_r]$ is satisfying. Verify that $\psi[\theta]$ is satisfying.
3. For all $r \in R$, check whether $\theta \preceq_{\mathrm{lex}} \theta_r$ holds. If this is the case, accept θ, otherwise reject.

Clearly, the runtime of this procedure is polynomial in t and $|G : H|$.

Correctness of Verifier. Let Δ be a G-orbit of assignments. Note that Δ is a disjoint union of H-orbits $\Delta_1, \ldots, \Delta_k$. In each Δ_i, there exists a unique assignment α_i such that $\psi[\alpha_i]$ is satisfying. Let θ be the lexicographically minimal element in $\{\alpha_1, \ldots, \alpha_k\}$. Note that we have $\theta^G = \bigcup_{r \in R} (r\theta)^H$ as every element of G can be decomposed as hr for $h \in H$ and $r \in R$. For $r \in R$, there exists $i_r \in [k]$ with $(r\theta)^H = \Delta_{i_r}$. Hence, there exists $h_r \in H$ with $\alpha_{i_r}^{h_r} = \theta^r$. By construction, θ together with the certificate $\sigma = \{(\alpha_{i_r}, h_r) \colon r \in R\}$ is accepted by the verifier.

Now suppose that $\theta, \theta' \in \Delta$ are accepted by the verifier, and let $\sigma_\theta = \{(\theta_r, h_r) \colon r \in R\}$ and $\sigma_{\theta'} = \{(\theta'_r, h'_r) \colon r \in R\}$ denote corresponding certificates. Due to the decomposition of θ^G and since $\psi[\theta_r]$ and $\psi[\theta'_r]$ are satisfying for all $r \in R$, we have $\{\alpha_1, \ldots, \alpha_k\} = \{\theta_r \colon r \in R\} = \{\theta'_r \colon r \in R\}$. Since the verifier accepts both θ and θ', they coincide with the lexicographically minimal element in $\{\alpha_1, \ldots, \alpha_k\}$, so $\theta = \theta'$ follows. ◀

It should be noted that while the above lemma gives a valid upper bound, the resulting SBP is not practical: The SBP simply uses the additional variables to determine the representative for all cosets, and then determines a minimal one among them. This requires trying out all the symmetric choices, defeating the purpose of the SBP. However, the result can also be read as a hardness result. For example, for the matrix models studied in Section 3, we can restrict the group on each axis of the model as follows, while still being able to retrieve our hardness result (see Theorem 13):

▶ **Corollary 19.** *Consider a family of permutation groups $G_{m,n} = X_m \times Y_n$ with $X_m \leq \mathrm{Sym}(m)$ and $Y_n \leq \mathrm{Sym}(n)$, acting component-wise on $[m] \times [n]$. Assume that $|\mathrm{Sym}(m) : X_m| < \mathsf{poly}(m)$ and $|\mathrm{Sym}(n) : Y_n| < \mathsf{poly}(n)$ holds. Then, efficient complete symmetry breaking with additional variables for $G_{m,n}$ implies $\mathsf{GI} \in \mathsf{coNP}$.*

Our main interest in studying subgroups of small index is sparked by a result on the structure of so-called large primitive groups, which forms an important building block of the quasi-polynomial isomorphism test for general graphs [5]. Roughly speaking, every primitive group $G \leq \mathrm{Sym}(n)$ with $|G| \geq n^{1+\log_2 n}$ contains a normal subgroup N with $|G : N| \leq n$ exhibiting a natural Johnson action. If the converse of Lemma 18 holds, we can thus employ our results on Johnson groups to study the complexity of symmetry breaking for large primitive groups.

5 Upper Bounds

Complementing the results from the previous sections, we show that certain families of groups can be efficiently handled. We begin by recalling three simple cases.

Groups of Polynomial Order. The first case pertains to groups where the order is polynomial in the size of the original formula. For these groups, we can explicitly write a constraint that breaks each element of the group [14]. The resulting constraint is complete and of polynomial size in the formula.

Symmetric Groups. Symmetric groups in their natural action can be handled by imposing an ordering on the assignments. For $\mathrm{Sym}(n)$, this can be achieved by the predicate $\psi_n = x_1 \leq x_2 \leq \cdots \leq x_n$.

A slight extension of symmetric groups are known and used in practice, namely row-interchangeability subgroups [17, 40]. We say that a permutation group $G \leq \mathrm{Sym}(\Omega)$ exhibits *row-interchangeability* if Ω can be arranged in an $n \times m$-matrix $X = (x_{ij})$ such that G consists precisely of the permutations of the *rows* of X. This symmetry can be broken by lexicographically ordering the rows in any assignment $\theta\colon X \to \{0,1\}$ (viewed as an $n \times m$-matrix). Formally, for $i \in [n-1]$, let $\lambda_i^k := (\bigwedge_{r\in[k-1]}(x_{ir} = x_{(i+1)r})) \to (x_{ik} \leq x_{(i+1)k})$. Then $\lambda_{n,m} := \bigwedge_{i=1}^{n-1} \bigwedge_{k=1}^{m} \lambda_i^k$ is a symmetry breaking predicate for G.

Disjoint direct decomposition. A direct product $G = G_1 \times \cdots \times G_r$ of permutation groups is called a *disjoint direct decomposition* if the subgroups $G_1, \ldots, G_r$ have pairwise disjoint supports. Disjoint direct products naturally arise and have been successfully used in practice [17]. The finest disjoint direct decomposition can be computed in polynomial time for general permutation groups [12], and in quasi-linear time for automorphism groups of graphs [3]. For the sake of completeness, we argue that disjoint direct decompositions can be exploited without giving up on complete symmetry breaking.

▶ **Lemma 20.** *Let $G \leq \mathrm{Sym}(\Omega)$ be a permutation group with a disjoint direct product decomposition $G = G_1 \times \cdots \times G_r$. For $i \in [r]$, let Ω_i denote the support of G_i and assume that a complete symmetry breaking predicate γ_i for G_i, viewed as a permutation group on Ω_i, is given. In particular, we require $\mathrm{Var}(\gamma_i) \subseteq \Omega_i$. Then $\gamma := \gamma_1 \wedge \cdots \wedge \gamma_r$ is a complete symmetry breaking predicate for G.*

Proof. Let $F \subseteq \Omega$ be the set of points fixed by G. Then $\Omega = \Omega_1 \,\dot\cup \ldots \dot\cup\, \Omega_r \,\dot\cup\, F$. An assignment $\theta\colon \Omega \to \{0,1\}$ can be viewed as a tuple $(\theta_1, \ldots, \theta_r, \theta_F)$ of assignments defined on $\Omega_1, \ldots, \Omega_r, F$, respectively, and we have $\theta^G = {\theta_1}^{G_1} \times \cdots \times {\theta_r}^{G_r} \times \{\theta_F\}$. Hence θ satisfies γ if and only if θ_i satisfies γ_i for every $i \in [r]$. Thus, γ is a complete symmetry breaking predicate for G. ◀

The size of the constraint γ is linear in the size of the constraints $\gamma_1, \ldots, \gamma_r$.

Wreath Products. We now turn our attention to so-called *wreath products*. They naturally occur as the automorphism groups of tree-like structures and can be detected as the induced action on a block system [45]. Tree-like appendages are already detected and exploited by practical symmetry detection algorithms [4], and thus these wreath products seem readily available.

Indeed, certain cases of wreath products can be efficiently handled in symmetry breaking. Specifically, we show that automorphism groups of trees can be taken care of (see Theorem 4).

Intuitively, a wreath symmetry occurs if the domain can be partitioned into equally-sized parts with identical symmetries that can be permuted among each other. The corresponding symmetry group is made of a group describing the possible permutations of the points within a part, and a group describing the permutation of the parts. Formally, let $G \leq \mathrm{Sym}(n)$ and $H \leq \mathrm{Sym}(m)$, and consider the imprimitive action of $G \wr H$ on $X := \{x_{ij} \colon i \in [n],\, j \in [m]\}$. Explicitly, it is given by $x_{ij}^{((g_1,\ldots,g_m),h)} = x_{g_{h(j)}(i)\,h(j)}$. For $\theta\colon X \to \{0,1\}$ and $j \in [m]$, let $\theta_j := \theta|_{\{x_{1j},\ldots,x_{nj}\}}$ and write $\theta = (\theta_1, \ldots, \theta_m)$.

Wreath Products with CNF. Let us first focus on CNF predicates. Recall the predicate $\lambda_{m,n}$ from the beginning of this section. The following result shows that a symmetry breaking predicate for a permutation group G can be "lifted" to a predicate for $G \wr \mathrm{Sym}(m)$:

▶ **Lemma 21.** *Assume that γ is a complete symmetry breaking predicate for $G \leq \mathrm{Sym}(n)$ and set $\gamma_j := \gamma(x_{1j}, \ldots, x_{nj})$ for all $j \in [m]$. Then $\nu := \bigwedge_{j\in[m]} \gamma_j \wedge \lambda_{m,n}$ is a complete symmetry breaking predicate for $W := G \wr \mathrm{Sym}(m)$.*

Proof. Let $\theta\colon X \to \{0,1\}$ be an assignment. For every $j \in [m]$, there exists $g_j \in G$ such that $\theta_j^{g_j}$ satisfies γ. Write $\theta' := \theta^{((g_1,\dots,g_m),1)}$. There exists $h \in \mathrm{Sym}(m)$ with $\theta'_{h^{-1}(1)} \preceq_{\mathrm{lex}} \cdots \preceq_{\mathrm{lex}} \theta'_{h^{-1}(m)}$. Hence, the assignment $\theta'^{(1,h)} \in \theta^W$ satisfies ν.

On the other hand, consider assignments $\theta, \theta'\colon X \to \{0,1\}$ satisfying ν, and assume $\theta' = \theta^{((g_1,\dots,g_m),h)}$ for $\big((g_1,\dots,g_m),h\big) \in W$. For all $j \in [m]$, this implies $\theta'_j = \theta_{h^{-1}(j)}^{g_{h^{-1}(j)}} \in \theta_{h^{-1}(j)}{}^G$. As θ'_j and $\theta_{h^{-1}(j)}$ satisfy γ, they coincide, so we may choose $g_1 = \cdots = g_m = 1$. Since θ and θ' satisfy $\lambda_{m,n}$, we have $\theta_1 \preceq_{\mathrm{lex}} \cdots \preceq_{\mathrm{lex}} \theta_m$ and $\theta'_1 \preceq_{\mathrm{lex}} \cdots \preceq_{\mathrm{lex}} \theta'_m$. This yields $\theta'_j = \theta_j$ for all $j \in [m]$, so $\theta = \theta'$ follows. ◀

▶ Remark 22. The size of the predicate ν given in Lemma 21 is in $\mathcal{O}(s(\gamma)m + nm)$, where $s(\gamma)$ denotes the size of γ. Note that if $s(\gamma) \in \mathcal{O}(n)$ holds, then the size of ν is linear in the domain size nm of the wreath product.

▶ **Corollary 23.** *The predicate $\nu = \bigwedge_{j\in[m]}(x_{1j} \leq \cdots \leq x_{nj}) \wedge \lambda_{m,n}$ is a complete symmetry breaking predicate for $\mathrm{Sym}(n) \wr \mathrm{Sym}(m)$.*

Combining the results for direct disjoint decompositions and wreath products, it readily follows that automorphism groups of trees can be handled efficiently (see Theorem 4).

Proof of Theorem 4. The group G can be constructed by iterated disjoint direct decompositions and wreath products in which the top group is a full symmetric group [41]. Combining Lemma 20 and Proposition 21 thus yields a symmetry breaking predicate for G. Inductively, it follows from Remark 22 that the size of this predicate is linear. ◀

Wreath Products with Circuits. Using circuits and a few further assumptions, we treat general wreath products $W := G \wr H$.

▶ **Theorem 5.** *Let $G \leq \mathrm{Sym}(n)$ and $H \leq \mathrm{Sym}(m)$ be permutation groups. Assume that a complete symmetry breaking circuit for G can be computed in polynomial time. Moreover, suppose that for every partition P of $[m]$, the partition stabilizer S of P in H and a complete symmetry breaking circuit for S can be computed in polynomial time. Then there is a complete symmetry breaking circuit for the wreath product $W := G \wr H$ that can be computed in polynomial time.*

Proof sketch. It suffices to give a polynomial-time algorithm for symmetry breaking for W. Let $\theta\colon X \to \{0,1\}$ be an assignment and write $\theta = (\theta_1,\dots,\theta_m)$ as before. We obtain the elements of $\theta^{G\wr H}$ by applying elements of G to $\theta_1,\dots,\theta_m$ and permuting the resulting strings using an element of H.

Based on this, the representative $\theta = (\theta_1,\dots,\theta_m)$ of an orbit is chosen as follows: We may require that all of $\theta_1,\dots,\theta_m$ are accepted by the symmetry breaking circuit for G. It remains to fix a unique permutation of $\theta_1,\dots,\theta_m$ under an element of H. There, we proceed in a way resembling the lexicographic ordering used in the proof of Lemma 21: We require that the symmetry breaking circuit for H accepts the string s_1 consisting of the first entries of $\theta_1,\dots,\theta_m$. There might still be multiple permutations of $\theta_1,\dots,\theta_m$ with the same string of first entries. We thus proceed with the string of second entries s_2, but only taking those permutations in H into account that fix the string of first entries. Thus we require that the symmetry breaking circuit for $H \cap S_1$ accepts s_2, where S_1 is the stabilizer of s_1. The latter can be expressed as a partition stabilizer. We continue this way up to the last position and accept θ as the orbit representative if none of the intermediate verifications fails. ◀

The full proof of this theorem is a mere formalization of this idea. A detailed description as well as a discussion of correctness can be found in Appendix B. In general, it is unknown whether partition stabilizers can be efficiently computed (see [26]). However, for $H = \mathrm{Sym}(m)$, the stabilizer of $P = (\Omega_1, \ldots, \Omega_r)$ is simply given by $\mathrm{Sym}(\Omega_1) \times \cdots \times \mathrm{Sym}(\Omega_r) \leq \mathrm{Sym}(n)$, and hence readily computable. This also holds if the order of H is small. There, we obtain the following consequence of the preceding result:

▶ **Corollary 24.** *Let $G \leq \mathrm{Sym}(n)$ and $H \leq \mathrm{Sym}(m)$ be permutation groups. Assume that a complete symmetry breaking circuit for G can be computed in polynomial time and that $|H| \in \mathsf{poly}(n, m)$ holds. Then a complete symmetry breaking circuit for $G \wr H$ can be computed in polynomial time.*

6 Conclusion and Future Work

We laid the foundation for a systematic study of the complexity of symmetry breaking. A central tool in our investigation was the relation to the string canonization problem (see Section 3). In particular, we showed that polynomial time symmetry breaking for the row-column symmetry group, even with circuits and additional variables, implies $\mathsf{GI} \in \mathsf{coNP}$ (see Theorem 1). The same applies to proper Johnson actions (see Theorem 2). On the other hand, we showed that symmetry breaking in polynomial time is possible for several families of groups, including certain classes of wreath products (see Section 5).

Clearly, the dividing line between permutation groups that are "hard" instances for symmetry breaking, and those which admit efficient symmetry breaking, needs to be further explored. Among others, the following questions immediately arise:

1. Given a permutation group G and a subgroup H of polynomial index, does H admit efficient symmetry breaking if G does (i.e., does the converse of Lemma 18 hold)?
2. How difficult are permutation groups of intermediate size, in particular groups of quasi-polynomial order in the size of the domain?

A positive answer to Question 1 would not only contribute to further decomposition results, but it is particularly relevant as large primitive permutation groups are known to contain normal subgroups of small index exhibiting a Johnson action.

References

1 Fadi A. Aloul, Igor L. Markov, and Karem A. Sakallah. Shatter: efficient symmetry-breaking for boolean satisfiability. In *Proceedings of the 40th Design Automation Conference, DAC 2003, Anaheim, CA, USA, June 2-6, 2003*, pages 836–839. ACM, 2003. doi:10.1145/775832.776042.

2 Markus Anders and Pascal Schweitzer. Parallel computation of combinatorial symmetries. In *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 6:1–6:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.6.

3 Markus Anders, Pascal Schweitzer, and Mate Soos. Algorithms transcending the sat-symmetry interface. In *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy*, volume 271 of *LIPIcs*, pages 1:1–1:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SAT.2023.1.

4 Markus Anders, Pascal Schweitzer, and Julian Stieß. Engineering a preprocessor for symmetry detection. In *21st International Symposium on Experimental Algorithms, SEA 2023, July 24-26, 2023, Barcelona, Spain*, volume 265 of *LIPIcs*, pages 1:1–1:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SEA.2023.1.

5 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.

6 László Babai. Canonical form for graphs in quasipolynomial time: preliminary report. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1237–1246. ACM, 2019. doi:10.1145/3313276.3316356.

7 László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 171–183, New York, NY, USA, 1983. Association for Computing Machinery.

8 László Babai, Eugene M. Luks, and Ákos Seress. Permutation groups in NC. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 409–420. ACM, 1987. doi:10.1145/28395.28439.

9 László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988. doi:10.1016/0022-0000(88)90028-1.

10 Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Certified dominance and symmetry breaking for combinatorial optimisation. *J. Artif. Intell. Res.*, 77:1539–1589, 2023. doi:10.1613/JAIR.1.14296.

11 Bart Bogaerts, Jakob Nordström, Andy Oertel, and Çağrı Uluç Yıldırımoğlu. BreakID-kissat in SAT competition 2023 (system description). In *Proceedings of SAT Competition 2023: Solver, Benchmark and Proof Checker Descriptions*, Department of Computer Science Series of Publications B, Finland, 2023. Department of Computer Science, University of Helsinki.

12 Mun See Chang and Christopher Jefferson. Disjoint direct product decompositions of permutation groups. *J. Symb. Comput.*, 108:1–16, 2022. doi:10.1016/j.jsc.2021.04.003.

13 Michael Codish, Graeme Gange, Avraham Itzhakov, and Peter J. Stuckey. Breaking symmetries in graphs: The nauty way. In *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, volume 9892 of *Lecture Notes in Computer Science*, pages 157–172. Springer, 2016. doi:10.1007/978-3-319-44953-1_11.

14 James M. Crawford, Matthew L. Ginsberg, Eugene M. Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96), Cambridge, Massachusetts, USA, November 5-8, 1996*, pages 148–159. Morgan Kaufmann, 1996.

15 Paul T. Darga, Mark H. Liffiton, Karem A. Sakallah, and Igor L. Markov. Exploiting structure in symmetry detection for CNF. In *Proceedings of the 41th Design Automation Conference, DAC 2004, San Diego, CA, USA, June 7-11, 2004*, pages 530–534. ACM, 2004. doi:10.1145/996566.996712.

16 Jo Devriendt, Bart Bogaerts, and Maurice Bruynooghe. Symmetric explanation learning: Effective dynamic symmetry handling for SAT. In *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10491 of *Lecture Notes in Computer Science*, pages 83–100. Springer, 2017. doi:10.1007/978-3-319-66263-3_6.

17 Jo Devriendt, Bart Bogaerts, Maurice Bruynooghe, and Marc Denecker. Improved static symmetry breaking for SAT. In *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 104–122. Springer, 2016. doi:10.1007/978-3-319-40970-2_8.

18 Jo Devriendt, Bart Bogaerts, Broes De Cat, Marc Denecker, and Christopher Mears. Symmetry propagation: Improved dynamic symmetry breaking in SAT. In *IEEE 24th International*

Conference on Tools with Artificial Intelligence, ICTAI 2012, Athens, Greece, November 7-9, 2012, pages 49–56. IEEE Computer Society, 2012. doi:10.1109/ICTAI.2012.16.

19 John D. Dixon and Brian Mortimer. *Permutation Groups*. Graduate Texts in Mathematics. Springer New York, 1996. URL: https://books.google.de/books?id=4QDpFN6k61EC.

20 Pierre Flener, Alan M. Frisch, Brahim Hnich, Zeynep Kiziltan, Ian Miguel, Justin Pearson, and Toby Walsh. Breaking row and column symmetries in matrix models. In *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, volume 2470 of *Lecture Notes in Computer Science*, pages 462–476. Springer, 2002. doi:10.1007/3-540-46135-3_31.

21 Pierre Flener, Alan M. Frisch, Brahim Hnich, Zeynep Kızıltan, Ian Miguel, and Toby Walsh. Matrix modelling. Technical Report APES-36-2001, APES group (2001), 2001.

22 Pierre Flener, Justin Pearson, and Meinolf Sellmann. Static and dynamic structural symmetry breaking. *Ann. Math. Artif. Intell.*, 57(1):37–57, 2009. doi:10.1007/S10472-009-9172-3.

23 Ian P. Gent, Karen E. Petrie, and Jean-François Puget. Symmetry in constraint programming. In *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 329–376. Elsevier, 2006. doi:10.1016/S1574-6526(06)80014-3.

24 Andrew Grayland, Chris Jefferson, Ian Miguel, and Colva M. Roney-Dougal. Minimal ordering constraints for some families of variable symmetries. *Annals of Mathematics and Artificial Intelligence*, 57:75–102, 2009.

25 Marijn J. H. Heule. Optimal symmetry breaking for graph problems. *Math. Comput. Sci.*, 13(4):533–548, 2019. doi:10.1007/S11786-019-00397-5.

26 D.F. Holt, B. Eick, and E.A. O'Brien. *Handbook of Computational Group Theory*. Discrete Mathematics and Its Applications. CRC Press, 2005. URL: https://books.google.de/books?id=i2UjAASZ33YC.

27 Tommi A. Junttila, Matti Karppa, Petteri Kaski, and Jukka Kohonen. An adaptive prefix-assignment technique for symmetry reduction. *J. Symb. Comput.*, 99:21–49, 2020. doi:10.1016/J.JSC.2019.03.002.

28 Tommi A. Junttila and Petteri Kaski. Conflict propagation and component recursion for canonical labeling. In *Theory and Practice of Algorithms in (Computer) Systems - First International ICST Conference, TAPAS 2011, Rome, Italy, April 18-20, 2011. Proceedings*, volume 6595 of *Lecture Notes in Computer Science*, pages 151–162. Springer, 2011. doi:10.1007/978-3-642-19754-3_16.

29 George Katsirelos, Nina Narodytska, and Toby Walsh. On the complexity and completeness of static constraints for breaking row and column symmetry. In *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, volume 6308 of *Lecture Notes in Computer Science*, pages 305–320. Springer, 2010. doi:10.1007/978-3-642-15396-9_26.

30 Markus Kirchweger and Stefan Szeider. SAT modulo symmetries for graph generation. In *27th International Conference on Principles and Practice of Constraint Programming, CP*, volume 210 of *LIPIcs*, pages 34:1–34:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CP.2021.34.

31 Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Progress in Theoretical Computer Science. Birkhäuser/Springer, 1993. doi:10.1007/978-1-4612-0333-9.

32 Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982. doi:10.1016/0022-0000(82)90009-5.

33 Eugene M. Luks. Hypergraph isomorphism and structural equivalence of boolean functions. In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 652–658. ACM, 1999. doi:10.1145/301250.301427.

34 Eugene M. Luks and Amitabha Roy. The complexity of symmetry-breaking formulas. *Ann. Math. Artif. Intell.*, 41(1):19–45, 2004. doi:10.1023/B:AMAI.0000018578.92398.10.

35 François Margot. Pruning by isomorphism in branch-and-cut. *Math. Program.*, 94(1):71–90, 2002. doi:10.1007/S10107-002-0358-2.

36 Brendan D. McKay. Isomorph-free exhaustive generation. *J. Algorithms*, 26(2):306–324, 1998. doi:10.1006/JAGM.1997.0898.

37 Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014. doi:10.1016/j.jsc.2013.09.003.

38 Hakan Metin, Souheib Baarir, Maximilien Colange, and Fabrice Kordon. Cdclsym: Introducing effective symmetry breaking in SAT solving. In *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I*, volume 10805 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2018. doi:10.1007/978-3-319-89960-2_6.

39 James Ostrowski, Jeff T. Linderoth, Fabrizio Rossi, and Stefano Smriglio. Constraint orbital branching. In *Integer Programming and Combinatorial Optimization, 13th International Conference, IPCO 2008, Bertinoro, Italy, May 26-28, 2008, Proceedings*, volume 5035 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2008. doi:10.1007/978-3-540-68891-4_16.

40 Marc E. Pfetsch and Thomas Rehn. A computational comparison of symmetry handling methods for mixed integer programs. *Math. Program. Comput.*, 11(1):37–93, 2019. doi:10.1007/s12532-018-0140-y.

41 G. Pólya. Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen. *Acta Mathematica*, 68(none):145 – 254, 1937. doi:10.1007/BF02546665.

42 Jean-Charles Régin. Generalized arc consistency for global cardinality constraint. In William J. Clancey and Daniel S. Weld, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 1*, pages 209–215. AAAI Press / The MIT Press, 1996. URL: http://www.aaai.org/Library/AAAI/1996/aaai96-031.php.

43 Ashish Sabharwal. Symchaff: exploiting symmetry in a structure-aware satisfiability solver. *Constraints An Int. J.*, 14(4):478–505, 2009. doi:10.1007/S10601-008-9060-1.

44 Karem A. Sakallah. Symmetry and satisfiability. In *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 509–570. IOS Press, 2021. doi:10.3233/FAIA200996.

45 Ákos Seress. *Permutation Group Algorithms.* Cambridge Tracts in Mathematics. Cambridge University Press, 2003. doi:10.1017/CBO9780511546549.

46 Toby Walsh. On the complexity of breaking symmetry. *CoRR*, abs/2005.08954, 2020. arXiv:2005.08954.

47 Viktor N Zemlyachenko, Nickolay M Korneenko, and Regina I Tyshkevich. Graph isomorphism problem. *Journal of Soviet Mathematics*, 29:1426–1481, 1985.

Appendix

A Proof of Theorem 13

Proof. It suffices to show that there exists a canonical form f such that d-GCANON$_f \in \mathsf{NP}$ (see Lemma 12).

Certificate. Given a graph $\Gamma = (V, E)$, we consider the row-column symmetry group with $n = |V|$ rows and $m = |E|$ columns. More precisely, we assume $V = [n]$. We let $\delta_{n,m}(x_{11}, \ldots, x_{nm}, y_1, \ldots, y_p)$ denote the symmetry breaking circuit as computed by the polynomial time algorithm for row-column symmetry. Obviously, $p \in \text{poly}(m, n)$. Our chosen certificate for decision canonization is σ, where σ is an assignment to the variables $V = \text{Var}(\delta_{n,m}(x_{11}, \ldots, x_{nm}, y_1, \ldots, y_p))$.

Bipartite Reordering. We observe a technicality of our reduction: d-GCANON expects as input a graph, whereas our row-column symmetry breaking circuits essentially solve the decision canonization for bipartite graphs. We proceed using the same encoding for graphs into bipartite graphs as discussed previously (see proof of Theorem 1). The order of a graph Γ is fully determined by the order of its vertices. However, observe that when only restricting the order of the left partition, the corresponding bipartite graph bip(Γ) may still differ in the order of the edges, i.e., in the order of the *right* partition. Indeed, the symmetry breaking circuit may choose to accept any of these orderings of the right partition. We define the set bip$'(\Gamma)$ of bipartite graphs, where the left partition is ordered according to $V(\Gamma)$, and all potential reorderings of the right partition are contained. Note that $\bigcup_{\Delta \in \Gamma^{\text{Sym}(V(\Gamma))}} \text{bip}'(\Delta)$ covers all reorderings of the corresponding bipartite graphs.

Verifier. Given a certificate σ, our polynomial time verifier proceeds as follows:

1. For each column c in the matrix of x_{ij} variables, verify that the assignment has precisely two true variables in the column c. Formally, there exist $i, j \in [n]$ with $i \neq j$ such that $\sigma(x_{ic}) = 1$ and $\sigma(x_{jc}) = 1$, and for all $k \in [n]$ with $j \neq k \neq i$ it holds that $\sigma(x_{ic}) = 0$.
2. Check that Γ corresponds to the bipartite graph as given by the assignment σ: for each edge $\{v_1, v_2\} \in E$, we verify that there exists a column c such that $x_{v_1c} = 1$ and $x_{v_2c} = 1$.
3. Check if σ satisfies $\delta_{n,m}(x_{11}, \cdots, x_{nm}, y_1, \ldots, y_p)$. If it satisfies $\delta_{n,m}$, accept, otherwise reject.

By assumption, $\delta_{n,m}(x_{11}, \ldots, x_{nm}, y_1, \ldots, y_p)$ can be computed in polynomial time. Clearly, all the other steps can be computed in polynomial time as well.

Correctness of Verifier. We need to argue that for each isomorphism class of graphs $\Gamma^{\text{Sym}(V(\Gamma))}$, there is precisely one ordered graph G accepted by the verifier.

First, assume towards a contradiction that there is a Γ such that for all $\varphi \in \text{Sym}(V(\Gamma))$, and all certificates, Γ^φ is rejected by the verifier. Consider the corresponding bipartite graph bip(Γ). By assumption, we know that for all orderings of the left partition, all orderings of the right partition are rejected by the verifier. Hence, all orderings of bip(Γ) under $\text{Sym}([n]) \times \text{Sym}([m])$ are rejected by the symmetry breaking circuit. Hence, $\delta_{n,m}$ can not be a correct symmetry breaking circuit, which is a contradiction.

Next, assume towards a contradiction that there are two distinct isomorphic graphs $\Gamma \cong \Delta$ which are both accepted by the verifier. Since $\delta_{n,m}$ is a correct symmetry breaking circuit, this may only occur if there are corresponding bipartite graphs $\Gamma^* \in \text{bip}'(\Gamma), \Delta^* \in \text{bip}'(\Delta)$ such that $\Gamma^* = \Delta^*$. However, this would immediately imply $\Gamma = \Delta$. ◀

B Proof of Theorem 5

▶ **Theorem 5.** *Let $G \leq \mathrm{Sym}(n)$ and $H \leq \mathrm{Sym}(m)$ be permutation groups. Assume that a complete symmetry breaking circuit for G can be computed in polynomial time. Moreover, suppose that for every partition P of $[m]$, the partition stabilizer S of P in H and a complete symmetry breaking circuit for S can be computed in polynomial time. Then there is a complete symmetry breaking circuit for the wreath product $W := G \wr H$ that can be computed in polynomial time.*

Proof. Since we may turn a polynomial time algorithm into a polynomial-sized circuit, it suffices to give a polynomial-time algorithm for symmetry breaking for W.

Let ψ_G denote the symmetry breaking circuit for G, and for any partition stabilizer S in H, write ψ_S for the corresponding symmetry breaking circuit. For an assignment $\theta\colon X \to \{0,1\}$, write $\theta = (\theta_1, \dots, \theta_m)$ as before. For $i \in [n]$, let $c_i(\theta)$ be the string of length m consisting of the i-th entries of $\theta_1, \dots, \theta_m$. We define partitions $P_1, \dots, P_n$ of $[m]$ as follows: let P_1 denote the partition into the index sets of zero and one entries in $c_1(\theta)$. For $i \geq 2$, P_i is the refinement of P_{i-1} according to the zero-one-partition of $c_i(\theta)$. For $i \in [n]$, let S_i denote the partition stabilizer of P_i in H, and set $S_0 := H$.

Description of Algorithm. Given an assignment $\theta = (\theta_1, \dots, \theta_m)$, we define our algorithm as follows:

1. If $\psi_G[\theta_i]$ is non-satisfying for some $i \in [m]$, return false.
2. For $i \in [n]$, compute the vectors $c_i(\theta)$ as well as the partitions P_i and their stabilizers S_i.
3. For $i \in [n]$, check if $\psi_{S_{i-1}}[c_i(\theta)]$ is satisfying. If this fails for some $i \in [n]$, return false. Otherwise, return true.

Correctness of Algorithm. By assumption, partition stabilizers in H as well as all the necessary symmetry breaking circuits can be computed in polynomial time. The remaining steps of the algorithm can clearly be computed in polynomial time.

Replacing the input assignment $\theta = (\theta_1, \dots, \theta_m)$ by some element $\theta^{((g_1,\dots,g_m),1)} \in \theta^W$, we may assume that $\psi_G[\theta_1], \dots, \psi_G[\theta_m]$ are satisfying. By assumption, there exists $h_1 \in H$ such that $\psi_H[c_1(\theta^{(1,h_1)})]$ is satisfying. Moreover, there exists $h_2 \in S_1$ such that $\psi_{S_1}[c_2(\theta^{(1,h_2h_1)})]$ is satisfying. Note that $c_1(\theta^{(1,h_2h_1)}) = c_1(\theta^{(1,h_1)})$ holds due to $h_2 \in S_1$. Continuing, we obtain an element $\theta' := \theta^{(1,h_{n-1}\cdots h_1)} \in \theta^W$ for which the algorithm returns true.

On the other hand, suppose that $\theta = (\theta_1, \dots, \theta_m)$ and $\theta' = (\theta'_1, \dots, \theta'_m)$ are assignments in the same W-orbit accepted by the algorithm. Then $\psi_G[\theta_i]$ and $\psi_G[\theta'_i]$ are satisfying for all $i \in [m]$. Since θ and θ' lie in the same W-orbit, the strings $\theta_1, \dots, \theta_m$ and $\theta'_1, \dots, \theta'_m$ coincide up to reordering. The ordering of the substrings is lexicographic with respect to a successive application of H. This yields $\theta = \theta'$. ◀

C Proof of Theorem 16

▶ **Theorem 16.** *Let $t \geq 2$ be a fixed arity. Suppose there exists a polynomial time algorithm for generating complete symmetry breaking circuits with additional variables for the Johnson family of arity t. Then, d-GCANON $\in$ NP and hence* GI $\in$ coNP.

Proof. Again, Lemma 12 ensures that it suffices to prove d-GCANON $\in$ NP. From Lemma 14, it follows that it suffices to solve d-GCANON in non-deterministic polynomial time using a non-deterministic polynomial time oracle for decision canonization for uniform, symmetric t-ary relational structures.

Graph to t-ary Structure. Given a graph $\Gamma = (V, E)$, we define a t-uniform relational structure R_Γ as follows. Let $I \subseteq V$ be the set of isolated vertices. We have $V(R_\Gamma) = \{r_u : u \in V\} \cup \{v_1, \dots, v_{t-2}, a, b\}$. Observe that we added t vertices, namely $v_1, \dots, v_{t-2}, a, b$. These vertices will be called *bogus vertices*. We presume the order $r_{v_1} \prec \dots \prec r_{v_n} \prec v_1 \prec \dots \prec v_{t-2} \prec a \prec b$ for the symbols used in the construction. The hyperedges in R_Γ are given by

$$\Big\{\{r_u, r_w, v_1, \dots, v_{t-2}\} : \{u, w\} \in E\Big\} \cup \Big\{\{r_u, v_1, \dots, v_{t-2}, a\} : u \in V \setminus I\Big\} \\ \cup \Big\{\{v_1, \dots, v_{t-2}, a, b\}\Big\}.$$

Observe that

$$\begin{cases} \deg_{R_\Gamma}(r_u) = 0, & u \in I \\ \deg_{R_\Gamma}(r_u) = \deg_\Gamma(u) + 1, & u \in V \setminus I \\ \deg_{R_\Gamma}(v_i) = |E| + |V \setminus I| + 1, & i \in [t-2] \\ \deg_{R_\Gamma}(a) = |V \setminus I| + 1, & \\ \deg_{R_\Gamma}(b) = 1. & \end{cases}$$

In particular, for $u \in V \setminus I$, we have $1 < \deg_{R_\Gamma}(r_u) \leq |V \setminus I|$.

Now let Γ and Δ be graphs on n vertices. Without loss of generality, we may assume that Γ and Δ contain edges. We claim that Γ and Δ are isomorphic precisely if R_Γ and R_Δ are isomorphic. Assume that there exists an isomorphism $\varphi\colon R_\Gamma \to R_\Delta$. Denote the vertices in R_Γ and R_Δ with an exponent Γ and Δ, respectively. By the above degree conditions, we have $\varphi(b^\Gamma) = b^\Delta$ (here, the notation b^Γ refers to node b of graph Γ). As b^Γ is adjacent to $v_1^\Gamma, \dots, v_{t-2}^\Gamma, a^\Gamma$ (similarly in Δ), the degree conditions then imply $\varphi(a^\Gamma) = a^\Delta$. Now the vertices $v_1^\Gamma, \dots, v_{t-2}^\Gamma$ are mapped bijectively to $v_1^\Delta, \dots, v_{t-2}^\Delta$. In particular, φ induces a bijection between $\{r_u^\Gamma : u \in V(\Gamma)\}$ and $\{r_u^\Delta : u \in V(\Delta)\}$. It is then easy to see that φ induces an isomorphism between Γ and Δ.

On the other hand, if Γ and Δ are isomorphic, $R_\Gamma \cong R_\Delta$ follows from the fact that the above construction is isomorphism-invariant: all additional bogus vertices universally appear with all edges, as well as with all non-isolated vertices.

Furthermore, it is easy to see that if $\Gamma \neq \Delta$, then $R_\Gamma \neq R_\Delta$ follows.

Certificate. Our certificate will consist of a permutation $\varphi \in \mathrm{Sym}(V(R_\Gamma))$, as well as a certificate for decision canonization of t-ary structures σ.

Verifier. Our verifier proceeds as follows:

1. Using the decision canonization oracle for t-ary structures, continue if σ is a valid certificate for R_Γ^φ, and reject otherwise.
2. If for all pairs of vertices $v, v' \in V(\Gamma)$ with $v \prec v'$ it holds that $\varphi(r_v) \prec \varphi(r_{v'})$, accept, otherwise reject.

Correctness of Verifier. From the arguments above, we know that for all graphs Δ in the isomorphism class of Γ it holds that $R_\Gamma \cong R_\Delta$. The oracle in Step 1 will accept precisely one canonical t-ary structure R_Γ^φ in the isomorphism class of R_Γ. In turn, the verifier accepts a graph Γ, if and only if the order of the vertices is preserved in the canon R_Γ^φ (see Step 2). We remark that there may also be different φ' which map R_Γ to the canon, which may not preserve the order of $V(\Gamma)$. Clearly, there is at least one graph Γ in each isomorphism class that is accepted by the verifier.

Assume there is another graph $\Delta \neq \Gamma$ with $\Delta \cong \Gamma$ which is also accepted by the verifier. Since we know that $R_\Delta \cong R_\Gamma$ holds, this means there is a $\varphi' \in \mathrm{Sym}(V(R_\Delta))$ such that $R_\Delta^{\varphi'} = R_\Gamma^{\varphi}$ holds. In particular, φ' preserves the order of vertices in Δ. Recall that bogus vertices can only ever be mapped to bogus vertices. Therefore, $R_\Delta^{\varphi'} = R_\Gamma^{\varphi}$ immediately implies that the vertices of Γ and Δ can be mapped, in order, onto each other, while preserving the edge relation of the original graphs. In other words, $\Delta = \Gamma$ holds, which is a contradiction to the assumption that the verifier accepts Δ. ◀