Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)

On Computational Complexity of Automorphism Groups in Classical Planning

Alexander Shleyfman
Technion, Haifa, Israel
alesh@campus.technion.ac.il

Abstract

Symmetry-based pruning is a family of powerful methods for reducing search effort in planning as heuristic search. Applying these methods requires first establishing an automorphism group that is then used for pruning within the search process. Despite the growing popularity of state-space symmetries in planning techniques, the computational complexity of finding the automorphism group of a compactly represented planning task has not been formally established. In a series of reductions, we show that computing the automorphism group of a grounded planning task is GI-hard. Furthermore, we discuss the presentations of these symmetry groups and list some of their drawbacks.

Introduction

Symmetry breaking is a method for search-space reduction that has been well explored across several areas in computer science, and in particular in classical planning (Starke 1991; Emerson and Sistla 1996; Fox and Long 1999; Rintanen 2003; Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012; Gnad et al. 2017). Symmetry-based pruning divides the states in the search space into orbit-based equivalence classes, which in turn allows for exploring only one representative state per such class. Application of this technique to forward search partially curtails the exponential growth of the search space in the presence of objects with symmetric behavior. Beyond the state-space pruning, symmetries have been also successfully used in classical planning to enhance performance of heuristics (Domshlak, Katz, and Shleyfman 2013; Sievers et al. 2015b), prune redundant operators (Wehrle et al. 2015; Fišer, Álvaro Torralba, and Shleyfman 2019), and even decompose planning tasks (Abdulaziz, Norrish, and Gretton 2015).

In the foundations of all these techniques lies computing a subgroup of automorphisms of the state-transition graph of the problem. Considering the complexity of finding automorphism groups of state-transition graphs, Juntilla (2003) showed that the problem of finding symmetry groups in Petri nets is equivalent to the graph automorphism problem. Given the established connections between reachability problem in Petri nets and classical planning (Bonet et al. 2008), a rather direct corollary of that result is that computing the automorphism group of explicitly given state-transition graphs of classical planning problems is reducible to the Graph Isomorphism problem (GI-hard). However, since the state-transition graph in classical planning is worst-case exponential in the size of its compact representation, and since the work on symmetries in planning is focused on symmetries derived from such compact representations, the relevance of this complexity result is rather limited.

The first notion of grounded symmetries for classical planing was proposed by Pochter *et al.* (2011), and then refined by Domshlak *et al.* (2012). The definitions presented in these works, practical however they are, were based on the notion of colored graphs, and thus are quite cumbersome to reason about. Later on, Shleyfman *et al.* (2015) came up with the notion of structural symmetries that captures previously proposed concepts, and which can be derived from the syntax of a planning task in a simple declarative manner. Still, to our knowledge, the complexity of computing the automorphism group of a grounded planning task remained open.

In this work, we present reductions to graphs that establish a "negative" result that computing automorphism groups for grounded planning tasks (in both FDR and STRIPS formalisms) is as hard as for general undirected graphs. Along this route, we also show a nontrivial connection between the automorphism groups of a planning task and its causal graph, and discuss the presentations of the groups of classical planning tasks, as well as the suitableness of group generators for representation of the group's structure and size.

Background

To define a *planning task* we use the finite-domain representation formalism (FDR) (Bäckström and Nebel 1995; Helmert 2006). Each planning task is given by a tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$, where $\mathcal{V}$ is a set of multivalued *variables*, each associated with a finite domain $\mathcal{D}(v)$. The sets of variable/value pairs are written as $\langle var, val \rangle$, and sometimes referred as *facts*. A *state* s is a full variable assignments which maps each variable $v \in \mathcal{V}$ to some value in its domain, i.e. $s(v) \in \mathcal{D}(v)$. For $V \subseteq \mathcal{V}$, $s[V]$ denotes the partial assignment (also referred as a *partial state*) of s over V. *Initial state* I is a state. The *goal* G is a partial assignment. Let p be a partial assignment. We denote

by $vars(p) \subseteq \mathcal{V}$ the subset of variables on which p is defined. For two partial assignments p and q, we say that p *satisfies* q, if $vars(q) \subseteq vars(p)$, and $p[v] = q[v]$ for all $v \in vars(q)$, this is denoted by $p \models q$. $\mathcal{A}$ is a finite set of *actions*, where each action is represented by a triplet $\langle \mathsf{pre}(a), \mathsf{eff}(a), \mathsf{cost}(a) \rangle$ of *precondition*, *effect*, and *cost*, where $\mathsf{pre}(a)$ and $\mathsf{eff}(a)$ are partial assignments to $\mathcal{V}$, and $\mathsf{cost}(a) \in \mathbb{R}^{0+}$. In this work we assume all actions are of unit-cost, unless stated otherwise. An action a is *applicable* in a state s if $s \models \mathsf{pre}(a)$. Applying a in s changes the value of all $v \in vars(\mathsf{eff}(a))$ to $\mathsf{eff}(a)[v]$, and leaves s unchanged elsewhere. The outcome state s' is denoted by $s[\![a]\!]$.

S denotes the set of all states of Π. We say that action sequence π is a *plan*, if it begins in I, ends in s_G s.t. $s_G \models G$, and each action in π is iteratively applicable, i.e. for each $a_i \in \pi$ holds that $s_{i-1} \models \mathsf{pre}(a_i)$ and $s_{i-1}[\![a_i]\!] = s_i$. The cost of a plan is defined as $\mathsf{cost}(\pi) = \sum_{a_i \in \pi} \mathsf{cost}(a_i)$. An *optimal* plan, is a plan of a minimal cost. Here, since we assumed unit-cost domains, an optimal plan is a plan of the shortest length. The *state space* of Π is denoted $\mathcal{T}_\Pi$.

A *directed graph* is a pair $\langle N, E \rangle$ where N is the finite set of *vertices*, and $E \subseteq N^2$ is the set of *edges*, where each edge is an ordered pair of vertices. A*loop* (sometimes referred as self-loop) is a directed edge from a vertex to itself. In what follows, we will consider only *simple graphs*, i.e. graphs with no loops and no parallel edges. A directed graph with no cycles will be called *directed acyclic graph* (DAG).

An *undirected graph* is a pair $\langle N, E \rangle$ where N, once again, is the set of *vertices*, and $E \subseteq \{e \subseteq N \mid |e| = 2\}$ is the set of edges.

Let $\mathcal{G} = \langle N, E \rangle$ be a (un-)directed graph, and let σ be a permutation over the vertices N. We say that σ is a graph *automorphism* (or just an automorphism, if this is clear from the context) when $(n, n') \in E$ iff $(\sigma(n), \sigma(n')) \in E$. The definition of a *graph automorphism* for an undirected graph is almost the same, where $\{n, n'\} \in E$ iff $\{\sigma(n), \sigma(n')\} \in E$. The automorphisms of a graph $\mathcal{G}$ are closed under composition, and for every automorphism there exists an inverse permutation which is also an automorphism. Thus, automorphisms of a graph form a group. We call this group an *automorphism group*, and denote it by $Aut(\mathcal{G})$. The identity element e in this group will be denoted by $id_\mathcal{G}$.

Causal Graph

A planning task may often be structurally complex. One way of capturing said complexity is via causal graphs. This idea is mentioned in numerous papers, e.g. Knoblock (1994), Bacchus and Yang (1994), Domshlak and Brafman (2002)), however here we will follow the definition given by Helmert (2004).

The *causal graph* of a planning task $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$ is a directed graph $CG(\Pi) = \langle \mathcal{V}, \mathcal{E} \rangle$, where $(u, v) \in \mathcal{E}$ if $u \neq v$ and there exists $a \in \mathcal{A}$, s.t. $u \in vars(\mathsf{pre}(a)) \cup vars(\mathsf{eff}(a))$ and $v \in vars(\mathsf{eff}(a))$.

In a nutshell, the causal graph contains an edge from a source variable to a target variable, if changing the value of the target variable may depend on the value of the source variable, even if it is only a co-depending effect.

Structural Symmetries

The second ingredient we need was introduced by Shleyfman *et al.* (2015). This subsection defines the notion of *structural symmetries*, which captures previously proposed concepts of symmetries in classical planning. In short, structural symmetries are relabelling of the FDR of a given planning task Π. Variables are mapped to variables, values to values (preserving the $\langle var, val \rangle$ structure), and actions are mapped to actions. In this work, we follow the definition of structural symmetries for FDR planning tasks as defined by Wehrle *et al.* (2015). For a planning task $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$, let P be the set of Π's facts, and let $P_\mathcal{V} := \{\{\langle v, d \rangle \mid d \in \mathcal{D}(v)\} \mid v \in \mathcal{V}\}$ be the set of sets of facts attributed to each variable in $\mathcal{V}$. We say that a permutation $\sigma : P \cup \mathcal{A} \to P \cup \mathcal{A}$ is a *structural symmetry* if the following holds:

1. $\sigma(P_\mathcal{V}) = P_\mathcal{V}$,
2. $\sigma(\mathcal{A}) = \mathcal{A}$, and, for all $a \in \mathcal{A}$, $\sigma(\mathsf{pre}(a)) = \mathsf{pre}(\sigma(a))$, $\sigma(\mathsf{eff}(a)) = \mathsf{eff}(\sigma(a))$, and $\mathsf{cost}(\sigma(a)) = \mathsf{cost}(a)$.
3. $\sigma(G) = G$.

We define the application of σ to a set X by $\sigma(X) := \{\sigma(x) \mid x \in X\}$, where σ is applied recursively up to the level of action labels and facts. For example, let s be a partial state, since s can be represented a set of facts. Applying σ to s will result in a partial state s', s.t. for all facts $\langle v, d \rangle \in s$ it holds that $\sigma(\langle v, d \rangle) = \langle v', d' \rangle \in s'$ and $s'[v'] = d'$.

A set of structural symmetries Σ for a planning task Π induces a subgroup Γ of the automorphism group $Aut(\mathcal{T}_\Pi)$, which in turn defines an equivalence relation over the states S of Π. Namely, we say that s is *symmetric* to s' iff there exists an automorphism $\sigma \in \Gamma$ such that $\sigma(s) = s'$. The group of all structural symmetries of Π will be denoted by $Aut(\Pi)$.

Symmetries and Problem Description Graphs

The last mathematical object we would like to mention in this section is the problem description graph (PDG) that was introduced by Pochter *et al.* (2011), and later on reformulated for different purposes by Domshlak *et al.* (2012), and Shleyfman *et al.* (2015). In this work we will use the definition of PDG for the FDR planning tasks. It is important to point out that this structure has no direct use in the proofs in the work below. However, since we want to illustrate some of our claims by graphic examples, PDGs become quite helpful, since in contrast to structural symmetries PDGs are graphs, and hence can be presented in a picture.

Definition 1. *Let Π be a* FDR *planning task. The* ***problem description graph*** *(PDG) of Π is the colored directed graph $\langle N, E \rangle$ with nodes*

$$N = N_\mathcal{V} \cup \bigcup_{v \in \mathcal{V}} N_{\mathcal{D}(v)} \cup N_\mathcal{A}$$

where $N_\mathcal{V} = \{n_v \mid v \in \mathcal{V}\}$, $N_{\mathcal{D}(v)} = \{n_{\langle v,d \rangle} \mid d \in \mathcal{D}v\}$,

and $N_{\mathcal{A}} = \{n_a \mid a \in \mathcal{A}\}$*; node colors*

$$col(n) = \begin{cases} 0 & \text{if } n \in N_{\mathcal{V}} \\ 1 & \text{if } n \in \bigcup_{v \in \mathcal{V}} N_{\mathcal{D}(v)} \text{ and } \langle v, d \rangle \in G \\ 2 & \text{if } n \in \bigcup_{v \in \mathcal{V}} N_{\mathcal{D}(v)} \text{ and } \langle v, d \rangle \notin G \\ 3 + \mathsf{cost}(a) & \text{if } n_a \in N_{\mathcal{A}} \end{cases}$$

and edges

$$E = \bigcup_{v \in \mathcal{V}} E^v \cup \bigcup_{a \in \mathcal{A}} E_a^{\mathsf{pre}} \cup E_a^{\mathsf{eff}},$$

where $E^v = \{(n_v, n_{\langle v,d \rangle}) \mid d \in \mathcal{D}(v)\}$, $E_a^{\mathsf{pre}} = \{(n_a, n_{\langle v,d \rangle}) \mid \langle v, d \rangle \in \mathsf{pre}(a)\}$, *and* $E_a^{\mathsf{eff}} = \{(n_{\langle v,d \rangle}, n_a) \mid \langle v, d \rangle \in \mathsf{eff}(a)\}$.

In their work, Pochter *et al.* (2011) observed that *PDG symmetry* is a symmetry of $\mathcal{T}_\Pi$ that is induced by a graph automorphism of the PDG of Π. In what follows, we will denote by $Aut(PDG(\Pi))$ the automorphism group of the PDG of the task Π. Shleyfman *et al.* (2015), in turn showed that every structural symmetry of Π corresponds to a PDG symmetry of Π in the sense that they induce the same transition graph symmetry, thus we will assume $Aut(PDG(\Pi)) = Aut(\Pi)$. The illustrative examples of planning tasks will also be presented via PDGs.

Complexity

In this section we aim to prove that for each undirected graph one may construct a planning task with a similar automorphism group. We show a simple reduction that will prove that the computation of this automorphism group is at least GI-hard.

The Graph Isomorphism problem (GI) is a well-known problem that gave its name to a whole complexity class. This problem is a decision problem of determining whether two finite graphs are isomorphic. Another well-known problem is the graph automorphism problem, that is a problem of testing whether a graph has a nontrivial automorphism. This is at least as hard as solving the decision problem of whether automorphism group of a given graph is trivial or not. The graph automorphism problem is polynomial-time many-one reducible to the graph isomorphism problem (Mathon 1979) (the converse reduction is unknown). Thus, given the reduction below, we can say that computing the automorphism group of a given planning task is at least GI-hard. The latest result by Babai (2015) claims (most probably rightfully) that GI can be solved in quasi-polynomial time, i.e. in $\exp((\log n)^{O(1)})$. The best previous bound stood on $O(\exp(\sqrt{n \log n}))$ (Babai and Luks 1983).

Morphisms

While discussing relations between the automorphism groups of different structures, we should first introduce the notation of structure preserving mappings and comparison between these groups. In this section we will rely mostly on the basic definitions of the group theory taken from "Topics in algebra" by Herstein (1975). Let us start with some useful mappings:

Definition 2. *Let* G *and* G' *be groups:*

1. *and let* $\phi : G \mapsto G'$ *be a mapping that satisfies* $\phi(ab) = \phi(a)\phi(b)$ *for all* $a, b \in G$*. Then,* ϕ *is a* ***homomorphism*** *of* G *to* G'*.*
2. *If, in addition to 1.,* ϕ *is also a bijection, it is called an* ***isomorphism****, this is denoted by* $G \cong G'$*, or simply* $G = G'$*.*
3. *If, in addition to 1.,* ϕ *is also an injection, then there exist a subgroup* $H \leq G'$*, s.t.* $H \cong G$*. In this case we will write simply* $G \leq G'$*.*

Since we still would like to embed groups into groups, we will need the following Definitions and Theorem (once again taken from the book "Topics in algebra").

Definition 3. *Let* H *be a subgroup of* G*, and let* x *be an element in* G*.*

1. *The set of elements* $Hx = \{hx \mid h \in H\}$ *is called a* ***right coset*** *of* H*. A* ***left coset*** *defined similarly.*
2. *If for every* $x \in G$ *holds that* $Hx = xH$*,* H *is called a* ***normal subgroup****.*
3. *Let* H *be a normal subgroup of a* G*. The set* ${}^{G}/_{H} := \{xH \mid x \in G\}$ *of all left cosets forms a* ***quotient group*** *of* G *modulo* H*.*

To establish additional relationships between homomorphisms, quotients, and subgroups we will need the following theorem by Noether (1927).

Definition 4. *Let* H *and* G *be two groups, and let* $\phi : G \to H$ *be a group homomorphism. The* ***kernel*** *of the map* ϕ*, denoted by* $ker(\phi)$*, is the set* $\phi^{-1}(id_H)$*.*

In some sense, the kernel of a homomorphism measures how much "non-injective" the homomorphism is. This measure is especially important if we want to form a connection between two subgroups that are not subgroups of each other. The next Theorem shows how one can construct such a subgroup (note that there is always at least one, trivial, homomorphism between two groups):

Theorem 1 (First Isomorphism Theorem)**.** *Let* G *and* H *be groups, and let* $\phi : G \to H$ *be a homomorphism. Then:*

1. *The kernel of* ϕ *is a normal subgroup of* G*,*
2. *The image of* ϕ *is a subgroup of* H*, and*
3. *The image of* ϕ *is isomorphic to the quotient group* ${}^{G}/_{ker(\phi)}$*.*

In particular, if ϕ *is surjective then* H *is isomorphic to* ${}^{G}/_{ker(\phi)}$*.*

Note that the proofs that both $ker(\phi) \leq G$ and $\phi(G) \leq H$ subgroups are immediate. Now, fully equipped with these abstract algebra tools, we can proceed to slaying our little mathematical dragon.

Reduction to a Single Variable

As we mentioned before, Pochter *et al.* (2011) introduced a method for deduction of the automorphism group for an FDR representation of a planning task using PDGs. While this representation is easy to visualize and understand, it is a bit inconvenient as an algebraic model of representation. On

the other hand, structural symmetries, while having a much simpler definition, lack the graphical appeal. Thus, while the proofs in this section will be formulated in the language of structural symmetries, the examples will be drawn as PDGs.

The first case we would like to examine is the case of a planning task Π with an unbounded variable domain. Under this condition only one variable per task is sufficient, to show the reduction to undirected graphs, since Zemlyachenko *et al.* (1985) showed that finding an isomorphism of a connected graph is a GI-complete problem. We will use this result to prevent the task from being reducible via standard reachability preprocessing. It is important to note that this complexity result holds for a clearly polynomial-time solvable task.

Proposition 1. *Let $\mathcal{G}$ be a connected undirected graph. Then, there exists planning task $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$, s.t. $Aut(\mathcal{G}) = Aut(\Pi)$ and $|\mathcal{V}| = 1$.*

Proof. Let $\langle N, E \rangle$ be the vertices and edges of $\mathcal{G}$, correspondingly, and let $\mathcal{V} = \{v\}$ be the single variable in the task Π. We will define the domain of v to be $\mathcal{D}(v) := \{v_x \mid x \in N\} \cup \{v_g\}$, where v_g is the goal value of the variable v ($G := \{\langle v, v_g \rangle\}$). Now, since the structural symmetries ignore the initial state, all is left to do is to define the actions of this task. Since we have only one variable, we will use the following notation $a_{v_x \to v_y} := \langle \{\langle v, v_x \rangle\}, \{\langle v, v_y \rangle\} \rangle$. The actions $\mathcal{A}$ of our task will be divided into two sets:

1. $\mathcal{A}_E := \{a_{v_x \to v_y}, a_{v_y \to v_x} \mid e = \{x, y\} \in E\}$, and
2. $\mathcal{A}_g := \{a_{v_x \to v_g} \mid x \in N\}$.

For a graphical example see Figure 1. Now, let us look at the map $\psi : N \to \{\langle v, d \rangle \mid d \in \mathcal{D}(v)\}$, by construction ψ is injective. Therefore the map $\phi : Aut(\mathcal{G}) \to Aut(\Pi)$:

$$\phi(\sigma)(\langle v, v_x \rangle) = \begin{cases} \psi(\sigma(\psi^{-1}(\langle v, v_x \rangle))) & \text{if } v_x \neq v_g, \\ \langle v, v_g \rangle & \text{otherwise} \end{cases}$$

is also injective, since it is easy to see that ψ preserves the relation on the edges, $\psi : E \to \mathcal{A}_E$ set-wise. The injection follows from the observation that $\langle v, v_g \rangle$ is a unique goal fact which can be mapped by σ only upon itself, and $\psi : N \to \mathcal{A}_g$ is a bijection. Thus we get the desired $Aut(\mathcal{G}) = Aut(\Pi)$. □

The immediate corollary of this proof is that computing the symmetry group of a planning task formulated in propositional STRIPS language (Fikes and Nilsson 1971) is also GI-hard.

Corollary 1. *Let $\mathcal{G}$ be a connected undirected graph. Then, there exists a delete-relaxed planning task Π^+ stated in the* STRIPS *formalism, s.t. $Aut(\Pi^+) = Aut(\mathcal{G})$.*

Proof sketch. Given that the paper, as it is, already overburdened with definitions and notations, we skip the formal definition of the STRIPS formalism.

Informally, a PDG for STRIPS will have a vertex of color 1 for each atom p, and two vertices for each action a, where the first one corresponds to the precondition of this action, and the second to both of the effects, i.e. $\mathsf{del}(a)$, and $\mathsf{add}(a)$. We will also add an edge $(\mathsf{pre}(a), \mathsf{eff}(a))$ for each action,

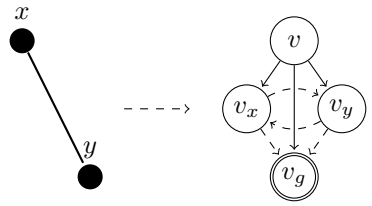

Figure 1: Illustration of a mapping of a single edge in a graph for Proposition 1: vertices $x, y \in N$,and an edge $e_{x,y} = \{x, y\} \in E$ are mapped to value vertices v_x, v_y and two dashed action edges $(v_x, v_y), (v_y, v_x)$, correspondingly. in addition, to preserve the PDG structure, each PDG graph will have a single variable vertex v, a single goal value v_g, and an edge (v_x, v_g) for each $x \in N$.

and set $\mathsf{pre}(a)$ to be of color 2 , and $\mathsf{eff}(a)$ of color 3 (assuming unit-cost). Now, for an atom $p \in \mathsf{eff}(a)$ we have a directed edge $(p, \mathsf{eff}(a)$, if $p \in \mathsf{add}(a)$ a directed edge $(p, \mathsf{eff}(a))$, and if $p \in \mathsf{del}(a)$ the directed edge will be $(\mathsf{eff}(a), p)$. Goal atoms will be of a color 4.

In the delete-relaxed case the eff vertex is redundant, since we can represent each action with a single vertex a, s.t. if $p \in \mathsf{pre}(a)$ we have an edge (p, a) and if $p \in \mathsf{add}(a)$ an edge (a, p).

Thus, to convert the FDR PDG from Proposition 1 to a PDG of a delete relaxed planning task, formulated in STRIPS, all we need to do is to remove the variable vertex v, by this removing the mutex condition on all values\atoms. Since, by construction, v is a vertex fixed by every automorphism (this vertex has a unique color), $Aut(\Pi^+)$ will remain the same as in Proposition 1. □

Since both a precondition and an effect of each action in the proof are single-valued, mapping each value to an atom will produce a delete-relaxed STRIPS planning task, which is a special case of STRIPS planning tasks. Thus, by the fact that STRIPS$^+ \subseteq$ STRIPS we have:

Corollary 2. *Computing the automorphism group of a planning task formulated in* STRIPS *is GI-hard.*

It is also important to note that all the reductions presented in this section have a linear time complexity in the size of the task at hand.

Bounded Variable Domain

In their work Bäckström and Klein (1991) defined D-BOUNDED FDR PLANNING to be all tasks where $|\mathcal{D}(v)| \leq$ D for all $v \in \mathcal{V}$. They also showed that this class of subproblems lie in PSPACE-complete, even for D $= 2$. We are interested in this class since it is the minimal condition that bounds the variable domain, and we would like to establish a connection between the vertices of a given undirected graph and the variables of the planning task constructed by the reduction. This relation is not necessarily unique, and may be interpreted in numerous ways. As the middle ground, however, we chose to conduct our proof using causal graphs, since they are already well-explored, and had been shown

to have direct connections to the complexity of planning tasks (e.g. Giménez and Jonsson (2008), Katz and Domshlak (2008)). Unfortunately, the next statement asserts that there is no straightforward subgroup relation between the automorphism group of the planning task and the automorphism group of its causal graph.

Observation 1. *Exists a planning task* Π*, s.t.* $Aut(\Pi) \not\leq Aut(CG(\Pi))$ *and* $Aut(CG(\Pi)) \not\leq Aut(\Pi)$.

Proof. Let Π be a planning task with variables $\mathcal{V}$ and actions $\mathcal{A}$. Consider a set of variables $\mathcal{V} = \{v, u\}$, where $\mathcal{D}(v) = \{v_1, v_2, v_3, v_4\}$ and $\mathcal{D}(u) = \{u_1, u_2\}$. Let $\mathcal{A}$ be a set of actions, each with a single precondition and a single effect. To improve the readability of this example, we will use the following notation $a_{x_i \to y_j} := \langle\{\langle x, x_i\rangle\}, \{\langle y, y_j\rangle\}\rangle$, for example the action $a_{v_1 \to v_2} = \langle\{\langle v, v_1\rangle\}, \{\langle v, v_2\rangle\}\rangle$. Now, let us define a set of actions of Π,

$$\mathcal{A} = \{a_{v_i \to v_{(i \bmod 3)+1}}, a_{v_i \to v_4} \mid i \in [3]\} \cup \{a_{v_4 \to u_2}, a_{u_2 \to v_4}, a_{u_1 \to u_2}, \}.$$

Since we don't want our planning task to be redundant, we will set $G = \{\langle v, v_4\rangle\}$. It is easy to check that $Aut(\Pi)$ is generated by the cycle $(\langle v, v_1\rangle, \langle v, v_2\rangle, \langle v, v_3\rangle)$, i.e. for some $\sigma \in Aut(\Pi)$ holds that $\sigma(\langle v, v_1\rangle) = \langle v, v_2\rangle$, $\sigma(\langle v, v_2\rangle) = \langle v, v_3\rangle$, and $\sigma^3 = id_\Pi$ to complete the cycle, and that σ is fixed on all other facts. Thus, $Aut(\Pi) \cong \mathbb{Z}_3$, a cyclic group[1] of order 3. The causal graph and PDG of Π are depicted in Figure 2.

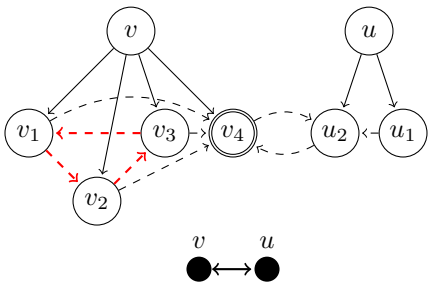

Figure 2: Illustration for Observation 1: The graph with the white nodes represents the PDG of the task described in the Observation in question. Since precondition and effect of each action are single-valued, we annotated them via dashed arrows. The goal fact is denoted by a double circle. Here it is easy to see that the automorphism group of the PGD is generated by the cycle (v_1, v_2, v_3) (red, dashed arrows). The filled dots represent the causal graph of the same task.

On the other hand, the causal graph of task Π is $\langle N = \{v, u\}, \mathcal{E} = \{(v, u), (u, v)\}\rangle$, and has the automorphism group that is isomorphic to $\mathbb{Z}_2$. Hence, since both $\mathbb{Z}_3, \mathbb{Z}_2$ have no nontrivial subgroups, the claim holds. □

Since the following sequence of proofs may seem a bit cumbersome we provide this cheat sheet that follows the skeleton proof for what is presented below. Let Π be an FDR planning task, with $PDG(\Pi)$ and $CG(\Pi)$, as its problem description and causal graphs, correspondingly.

0. Given a simple preprocessing $Aut(PDG(\Pi)) = Aut(\Pi)$ (Shleyfman *et al.* (2015)).
1. As we have just seen, $Aut(\Pi)$ is not a subgroup of $Aut(CG(\Pi))$, and vice versa (Observation 1).
2. However, and this is the main part of the proof, $Aut(\Pi)$ modulo the automorphisms of the values is a subgroup of $Aut(CG(\Pi))$ (Lemma 1). Intuitively, we remove the symmetries of the values, but keep the symmetry relations between the variables in the sense that if the variables v and u were not symmetric before the removal, they will not be symmetric afterwards.
3. In the special case, where all variables are "the same" using Lemma 1 we prove that $Aut(CG(\Pi)) = Aut(\Pi)$ (Theorem 2).
4. Note that $CG(\Pi)$ is a directed simple graph.
5. For any undirected graph $\mathcal{G}$, we have a special case of $CG(\Pi)$ (that is a DAG) s.t. $Aut(\mathcal{G}) = Aut(CG(\Pi))$ (Proposition 2).
6. For every undirected graph $\mathcal{G}$ there is a planning task Π s.t. $Aut(\Pi) = Aut(PDG(\Pi)) = CG(\Pi) = Aut(\mathcal{G})$ (Corollary 3).

All the reductions in this chart have a linear time complexity.

Since we still would like to establish a connection between the automorphism groups of a planning task and its causal graph, we will embed $Aut(CG(\Pi))$ into $Aut(\Pi)$ while removing the "undesirable" automorphisms. Intuitively, the next Lemma shows that if we strip all the automorphisms from $Aut(\Pi)$, so that they do not affect the variables of the task, the resultant subgroup can be embedded into the automorphism group of $CG(\Pi)$.

Lemma 1. *Let* $\phi : Aut(\Pi) \mapsto Aut(CG(\Pi))$ *be a map s.t. for each* $\sigma \in Aut(\Pi) : \phi(\sigma) = \sigma_{\mathcal{V}}$*, where* $\sigma_{\mathcal{V}}$ *is* σ *restricted to* $\mathcal{V}$*. Then,* ϕ *is a homomorphism, and* $^{Aut(\Pi)}/_{ker(\phi)} \leq Aut(CG(\Pi))$.

Proof. Once again, let Π be a planning task with variables $\mathcal{V}$ and actions $\mathcal{A}$. First, let us prove that $\sigma_{\mathcal{V}}$ is an automorphism. Let $(u, v) \in \mathcal{E}$ be an edge in $CG(\Pi)$. Hence exists $a \in \mathcal{A}$, s.t. $u \in vars(\mathsf{pre}(a)) \cup vars(\mathsf{eff}(a))$ and $v \in vars(\mathsf{eff}(a))$. Therefore, for each $\sigma \in Aut(\Pi)$ it holds that $\sigma(u) \in vars(\mathsf{pre}(\sigma(a))) \cup vars(\mathsf{eff}(\sigma(a)))$ and $\sigma(v) \in vars(\mathsf{eff}(\sigma(a)))$, from which follows that $(\sigma_{\mathcal{V}}(u), \sigma_{\mathcal{V}}(v)) \in \mathcal{E}$. The converse is true, since each σ^{-1} is also an automorphism.

Second, ϕ is a homomorphism, since for each $\sigma, \sigma' \in Aut(\Pi)$, it holds that $\phi(\sigma)\phi(\sigma') = \sigma_{\mathcal{V}}\sigma'_{\mathcal{V}} = (\sigma\sigma')_{\mathcal{V}} = \phi(\sigma\sigma')$, given that ϕ is a restriction to variables.

Now, $ker(\phi) = \{\sigma \in Aut(\Pi) \mid \sigma_{\mathcal{V}} = id_{\mathcal{V}}\}$, and by the first isomorphism theorem it holds that $^{Aut(\Pi)}/_{ker(\phi)} = \phi(Aut(\Pi)) \leq Aut(CG(\Pi))$. □

Following the intuition of Lemma 1, in the Theorem below we construct a planning task that has no "inner" automorphism. The automorphism group of such task should be

[1]The definition of a cyclic group is given below, in the Group Presentation section.

isomorphic to the automorphism groups of its causal graph. This theorem is the main result of this section.

Theorem 2. *Let $\mathcal{G}$ be a directed graph. Then, there exists a planning task Π, s.t. $\mathcal{G} = CG(\Pi)$, $Aut(\mathcal{G}) = Aut(\Pi)$.*

Proof. In this proof, given a directed graph $\mathcal{G} = \langle N, E\rangle$, we should construct a planning task Π that satisfies the conditions of the Theorem. First, it is clear that vertex $x \in N$ should correspond a variable $v \in \mathcal{V}$. Now, since we would like to use Lemma 1, the kernel of the homomorphism ϕ should be trivial. Thus, we set $\mathcal{D}v = \{T, F\}$, and add an action $a_{v:F\to v:T} := \langle\{\langle v, F\rangle\}, \{\langle v, T\rangle\}\rangle$, s.t. for each $\sigma \in Aut(\Pi)$ holds $\sigma(\langle v, F\rangle) \neq \langle v, T\rangle$. For each $(x, y) \in E$, let v and u be the corresponding variables in $\mathcal{V}$. To ensure that $\mathcal{G} = CG(\Pi)$, we add a unique action $a_{u:F\to v:F} := \langle\{\langle u, F\rangle\}, \{\langle v, F\rangle\}\rangle$, which, in turn, assures that if $\sigma(a_{u:F\to v:F}) \neq a_{u:F\to v:F}$, then either $\sigma(v) \neq v$ or $\sigma(u) \neq u$. In addition, we need to specify an initial state, and a goal description. Let those two be full assignments $I := \{\langle v, F\rangle \mid v \in \mathcal{V}\}$ and $G := \{\langle v, T\rangle \mid v \in \mathcal{V}\}$. Since, by construction of Π, σ never maps T to F, this leaves the automorphism group $Aut(\Pi)$ unchanged. To summarize, the constructed planning task $\Pi = \langle\mathcal{V}, \mathcal{A}, I, G\rangle$ looks as follows:

1. $\mathcal{V} = \{v \mid v \in N\}$, with $\mathcal{D}(v) = \{T, F\}$ for each v,
2. $\mathcal{A} = \{a_{v:F\to v:T} \mid v \in V\} \cup \{a_{v:F\to u:F} \mid (v, u) \in E\}$,
3. $I = \{\langle v, F\rangle \mid v \in \mathcal{V}\}$, and
4. $G = \{\langle v, T\rangle \mid v \in \mathcal{V}\}$.

Since the algebraic mapping can be hard to imagine, the PDG structure of edge (u, v) is depicted in Figure 3.

Now, let ϕ be a homomorphism as defined is Lemma 1. By construction of Π, ϕ is surjective and $ker(\phi) = id_{\mathcal{G}}$, thus $Aut(\mathcal{G}) = Aut(\Pi)$. □

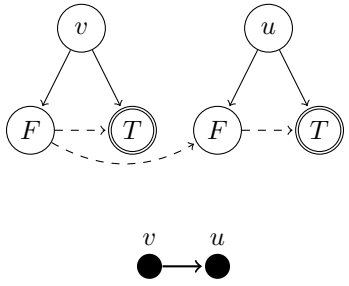

Figure 3: Illustration of a mapping of a single edge in a graph for Theorem 2: Once again, the graph with the white nodes represents the PDG of and edge (v, u) (depicted by filled nodes). Here it easy to see that there no "inner" symmetries, and the planning variable $(v, \mathcal{D}(v))$ can be mapped into a planning variable $(u, \mathcal{D}(u))$ exactly in one way.

Now all is left to show, that there is an automorphism group preserving reduction from undirected graphs to directed graphs.

Proposition 2. *Let $\mathcal{G}$ be a undirected graph. Then, there exists a directed simple graph $\mathbb{G}$, s.t. $Aut(\mathcal{G}) = Aut(\mathbb{G})$.*

The proof of this statement is not new, but we will use it later on to show that even special cases of planning tasks are difficult to solve, in the sense of finding the automorphism group. But first, let us introduce yet another notation. Let $\langle\mathcal{V}, \mathcal{E}\rangle$ be a directed graph. We denote the *indegree* and the *outdegree* (correspondingly) of a vertex $v \in \mathcal{V}$ by

$$\deg_{out}(v) := |\{u \in \mathcal{V} \mid (v, u) \in \mathcal{E}\}| \text{ and}$$
$$\deg_{in}(v) := |\{u \in \mathcal{V} \mid (u, v) \in \mathcal{E}\}|.$$

Proof. Let $\mathcal{G} = \langle N, E\rangle$ be an undirected graph. Let us define a directed graph $\mathbb{G} = \langle\mathcal{V}, \mathcal{E}\rangle$ as follows:

1. $\mathcal{V} := N \cup E$, and
2. $\mathcal{E} := \{(e, x), (e, y) \mid e = \{x, y\} \in E\}$.

Note that for the vertices in $\mathcal{V}$ for $x \in N$ and $e \in E$, $\deg_{out}(x) = \deg_{in}(e) = 0$. The graphic example of this construction can be seen if Figure 4. Hence, for each $\sigma \in Aut(\mathbb{G})$ holds that $\sigma(N) = N$ and $\sigma(E) = E$, where N and E are both sets of vertices in $\mathbb{G}$. Moreover, for each edge $e = \{x, y\}$ in E correspond to edges $(e, x), (e, y)$ in $\mathbb{G}$. Thus, for $\sigma \in Aut(\mathcal{G})$, $e = \{x, y\} \in E$ iff $\sigma(e) = \{\sigma(x), \sigma(y)\} \in E$ which corresponds to $(e, x), (e, y) \in \mathcal{E}$ iff $(\sigma(e), \sigma(x)), (\sigma(e), \sigma(y)) \in \mathcal{E}$. Using this, we will define $\phi : Aut(\mathcal{G}) \to Aut(\mathbb{G})$:

$$\phi(\sigma)(v) = \begin{cases} \sigma(v) & \text{if } v \in N, \\ e_{\sigma(x),\sigma(y)} & \text{if } v = e_{x,y} \\ & \text{for } e = \{x, y\} \in E. \end{cases}$$

Now, to prove that ϕ is an isomorphism we need to prove that ϕ is a surjection, and that $ker(\phi) = \{id_{\mathcal{G}}\}$. First, for each $\tau \in Aut(\mathbb{G})$, $\phi^{-1}(\tau) = \tau|_N \in Aut(\mathcal{G})$. Second, $\phi^{-1}(id_{\mathbb{G}}) = id_{\mathbb{G}}|_N = id_{\mathcal{G}}$. Thus, by the first isomorphism theorem it holds that ${}^{Aut(\mathcal{G})}/_{\{id_{\mathcal{G}}\}} = Aut(\mathcal{G}) = Aut(\mathbb{G})$. □

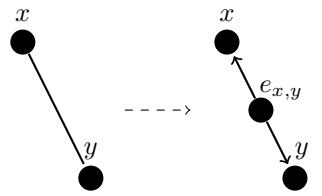

Figure 4: Illustration of a mapping of a single edge in a graph for Proposition 2: edge $e_{x,y} = \{x, y\} \in E$ is mapped to a vertex $e_{x,y} \in \mathcal{V}$ and two edges $(e, x), (e, y) \in \mathcal{E}$.

The next Corollary is the immediate consequence of Proposition 2 and Theorem 2.

Corollary 3. *Given a planing task Π, computing $Aut(\Pi)$ is equivalent to computing $Aut(\mathcal{G})$ for some undirected graph $\mathcal{G}$.*

Proof of Proposition 2 also shows that even planning tasks that have a bipartite one-way directed causal graphs (fork decomposition by Katz and Domshlak (2008)) may have an arbitrary finite automorphism group, which follows from the next theorem proven by Frucht (1949)

Theorem 3 (Frucht's theorem)**.** *Every finite group is the automorphism group of a finite undirected graph.*

To summarize, in this section we proved that for each connected undirected graph one may construct a planning task with the same automorphism group. This planning task may be formalized using FDR or STRIPS, in the former case, the task may have a single variable, or a variable domain bounded of a size 2, but not both.

Group Presentation

In this section we discuss the presentation of symmetry groups, show some trivial upper and lower bounds on the size of these groups, and present some drawbacks of this presentation.

Most of the tools for computing automorphism groups, such as *Bliss* (Junttila and Kaski 2007), *nauty* (McKay and Piperno 2014), and *saucy* (Darga, Sakallah, and Markov 2008), report the set of generators required to produce the $Aut(\mathcal{G})$ automorphism group of a given graph $\mathcal{G}$. In some works (Sievers et al. 2015a; 2017), the authors chose to report these numbers for each group, or even for each planning domain in the experimental benchmarks. This may lead to a false impression that given the number of generators of the group one may asses the size of the group, or even decide that given two groups with a different number of generators, these two groups must differ in structure (Sievers et al. 2015a). In this section we will show some faults in this approach. One indeed may calculate a simple lower bound on the size of the group. However, given the size of the generating set, one can neither approximate the upper bound on the size of the group (even with the order of each generator provided), nor can one decide whether two groups are not isomorphic given that their generating sets differ in size.

To this aim, we need some standard definitions:

Definition 5. *Let G be a group. We say that G has a **presentation** $\langle S \mid R \rangle$, where S is a set of **generators** so that every element of the group can be written as a product of powers of some of these generators, and R is a set of **relations** among those generators.*

*Let $F(S)$ be a **free group** on S, that is all finite words of S with the relation $vuu^{-1}w = sw$, where $v, u, w \in S$. The set of relations R is a subset of $F(S)$.*

The group G is said to have the above presentation if it is isomorphic to the quotient of $F(S)$ by the minimal normal subgroup that contains the set R.

*We say that presentation $\langle S \mid R \rangle$ of group G is **irreducible** if for no $S' \subset S$ holds that G isomorphic to $\langle S' \mid R|_{S'} \rangle$.*

From the First Isomorphism Theorem follows that every finite group has a presentation. As an easily obtained corollary of this statement we have that every finite group is finitely generated, since S can be taken to be G itself. To get a better grip on this definition we will give a couple of examples, that will be used further in this section.

Example 1. *The **cyclic group** is a group generated by a single element. The group $\mathcal{C}_k$ can be presented as $\langle S \mid R \rangle$ where:*

- $S := \{\sigma\}$;
- $R := \{\sigma^k\}$.

It is easy to see that for a given $k \in \mathbb{N}$, it holds that $|\mathcal{C}_k| = k$, and the group has exactly one generator. Example 1 yields the fact that the number of generators does not provide the upper bound on the size of the group. To calculate the lower bound we will prove the following lemma[2]:

Lemma 2. *Let G be a group with presentation $\langle S \mid R \rangle$, and let $S = \{g_1, \ldots, g_t\}$ be a set of t irreducible generators. Then, $|G| \geq 2^t$.*

Proof. Let G^m be a subgroup of G that has a set of generators $\{g_1, \ldots g_m\}$, for $1 \leq m < t$. Since the set S is irreducible with respect to G, every subset of S is also irreducible with respect to the subgroup of G it generates, thus $g_{m+1} \notin G^m$. Therefore, G^{m+1} has at least two cosets $e\,G^m = G^m$ and $g_{m+1}G^m$. By definition of cosets it holds that $G^m \cap g_{m+1}G^m = \emptyset$. Which leads to $|G^{m+1}| \geq 2|G^m|$. Thus, by induction on m we have that $G^t \geq 2^t$. □

The equality for the Lemma above is achieved on the group $\mathcal{C}_2^t \cong \mathbb{Z}_2^t = \prod_{i=1}^{t} \mathbb{Z}_2$. This means, the amount of elements (*order*) of a finite group is at least exponential in the size of group generators.

To show that group structure is not defined by the number of generators we will need at least one other group that is not cyclic. To this end we will define the symmetric group. Note that in the literature, *symmetry group* is often used as a synonym to the automorphism group of some mathematical object, where the *symmetric group* (used here) is the group of all permutations of some n identical objects.

Example 2. *The **symmetric group** $\mathcal{S}_n$ on a finite set of n symbols is the group whose elements are all the permutations on n distinct symbols. The group $\mathcal{S}_n$ can be written as $\langle S \mid R \rangle$ where:*

- $S := \{\sigma_i \mid i \in [n-1]\}$;
- $R := \{\sigma_i^2 \mid i \in [n-1]\} \cup \{\sigma_i\sigma_j\sigma_i^{-1}\sigma_j^{-1} \mid i \neq j \pm 1\} \cup \{(\sigma_i\sigma_j)^3 \mid i, j \in [n-1]\}$

Note that the cyclic notation is the natural representation of the structural symmetries group of a planning task, where each number $i \in [n]$ represents a fact used for state representation. It is also important to point out (and easy to check) that $\mathcal{S}_n$ has $n!$ elements. Using the cyclic notation, each element in the presentation can be written as $\sigma_i = (i, i+1)$, which means that σ_i maps the element i to element $i+1$, element $i+1$ is mapped to i, and other elements are mapped to themselves. This presentation is irreducible (Alperin and Bell 1995), meaning that none of the permutations can be excluded from the set S, where $|S| = n - 1$.

[2]This result is well known in group theory, but unfortunately we haven not found any citing source.

As we showed a bit earlier in this section the group $\mathcal{C}_2^n$ also has n generators, each of order two[3]. At the same time, we have $\mathcal{C}_2^n \ncong \mathcal{S}_{n+1}$, since $2^n \neq (n+1)!$, for $n > 1$.

By this example, reporting the number of generators per domain, or even for a specific group (even including the order of these generators) is not very informative, since these numbers basically tell us nothing about the size and/or structure of the group. For example, the group $\langle a, b \mid a^2, b^2 \rangle$ with only two generators both of order 2 is of an infinite size: $\{a, ab, aba, abab, \dots\}$.

Another way to write the cyclic group $\mathcal{S}_n$, may be given with only two cyclic generators $(1, 2)$ and $(2, \dots, n)$, which are also irreducible (Alperin and Bell 1995). Note that in the first cyclic presentation of $\mathcal{S}_n$ each generator has an order of 2, and in the second presentation the first cycle is of order 2, and the second cycle is of order $n - 1$. As one can see, each group may have more than one presentation, while calculating the minimal number of these generators per group is known to be at most $O(\log^2 n)$ space (Arvind and Torán 2006). The size of the group, however, can be calculated in polynomial time. Recall the famous Cayley's theorem (Herstein 1975).

Theorem 4 (Cayley's theorem)**.** *Every group G is isomorphic to a subgroup of the symmetric group acting on G.*

A direct corollary of this theorem is that every finite group is isomorphic to a subgroup of the symmetric group $\mathcal{S}_n$. Let Π be a planning task given either in FDR or STRIPS formalism. Given that structural symmetries provide us with a natural embedding of $Aut(\Pi)$ into $\mathcal{S}_n$, where n is the number of facts (in FDR) or atoms (in STRIPS) of the planning task, one can deduce that the immediate bound on the size of the group $Aut(\Pi)$ is $n!$. This bound does not depend on the size of the generating set, and it is almost sharp, since for K_n a clique graph on n vertices, we have that $Aut(K_n) = \mathcal{S}_n$. Hence, by the proof of Proposition 1, there is an FDR planning task s.t. $Aut(\Pi) \subseteq \mathcal{S}_n$ and Π has exactly $n + 1$ facts, if this planning task is not trivial (i.e., it must have exactly one goal fact, which is not symmetric to all others, by definition). Given the formalism STRIPS one may transform the clique K_n into a non-trivial planning task with exactly n atoms (where each atom of the task is a goal atom).

To obtain the exact size of the group $G \leq \mathcal{S}_n$ with a generating set S here given in a cyclic notation, one may chose to use the Schreier-Sims algorithm, developed by Sims (1970), with time-complexity $O(n^6 + |S|n^2)$, and later on improved by Jerrum (1986) and Knuth (1991) – $O(n^5 + |S|n^2)$. Currently, the state-of the-art in the terms of time-complexity is the randomized algorithm developed by Babai *et al.* (1995). The Monte Carlo time-complexity of the later is $O^{\sim}(n^3)$, which is considered near to optimal for general groups. Given all that, it remains unclear how the size of the automorphism group can be employed in the setting of classical planning.

[3]Order of an element g is the minimal number m s.t. $g^m = e$.

Acknowledgements

The work was supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities. The author would like to thank Carmel Domshlak, Ilya Shitov, and the anonymous reviewers for their helpful and constructive comments that greatly contributed to the improvement of the final version of the paper.

References

Abdulaziz, M.; Norrish, M.; and Gretton, C. 2015. Exploiting symmetries by planning for a descriptive quotient. In *IJCAI 2015*, 1479–1486.

Alperin, J. L., and Bell, R. B. 1995. *Groups and representations*. Graduate texts in mathematics. Springer.

Arvind, V., and Torán, J. 2006. The complexity of quasigroup isomorphism and the minimum generating set problem. In *ISAAC*.

Babai, L., and Luks, E. M. 1983. Canonical labeling of graphs. In *STOC 1983*, 171–183.

Babai, L.; Cooperman, G.; Finkelstein, L.; Luks, E.; and Seress, A. 1995. Fast monte carlo algorithms for permutation groups. In *Selected Papers of the 23rd Annual ACM Symposium on Theory of Computing*, 296–308. Orlando, FL, USA: Academic Press, Inc.

Babai, L. 2015. Graph isomorphism in quasipolynomial time. *CoRR* abs/1512.03547.

Bacchus, F., and Yang, Q. 1994. Downward refinement and the efficiency of hierarchical problem solving. *AIJ* 71(1):43–100.

Bäckström, C., and Klein, I. 1991. Planning in polynomial time: the SAS-PUBS class. *Computational Intelligence* 7(3):181–197.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS^+ planning. *Computational Intelligence* 11(4):625–655.

Bonet, B.; Haslum, P.; Hickmott, S. L.; and Thiébaux, S. 2008. Directed unfolding of petri nets. *Trans. Petri Nets and Other Models of Concurrency* 1:172–198.

C. Sims, C. 1970. Computational methods in the study of permutation groups. In *Computational Problems in Abstract Algebra*, 169–183.

Darga, P. T.; Sakallah, K. A.; and Markov, I. L. 2008. Faster symmetry discovery using sparsity of symmetries. In *DAC '08*, 149–154. New York, NY, USA: ACM.

Domshlak, C., and Brafman, R. I. 2002. Structure and complexity in planning with unary operators. In *AIPS 2002*, 34–43.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced symmetry breaking in cost-optimal planning as forward search. In *ICAPS 2012*.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2013. Symmetry breaking: Satisficing planning and landmark heuristics. In *ICAPS 2013*.

Emerson, E. A., and Sistla, A. P. 1996. Symmetry and model-checking. *Formal Methods in System Design* 9(1/2):105–131.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *AIJ* 2:189–208.

Fišer, D.; Álvaro Torralba; and Shleyfman, A. 2019. Operator mutexes and symmetries for simplifying planning tasks. In *AAAI 2019*.

Fox, M., and Long, D. 1999. The detection and exploitation of symmetry in planning problems. In *IJCAI 1999*, 956–961.

Frucht, R. 1949. Graphs of degree three with a given abstract group. *Canadian Journal of Mathematics* 1:365–378.

Giménez, O., and Jonsson, A. 2008. The complexity of planning problems with simple causal graphs. *JAIR* 31:319–351.

Gnad, D.; Torralba, Á.; Shleyfman, A.; and Hoffmann, J. 2017. Symmetry breaking in star-topology decoupled search. In *ICAPS 2017.*, 125–134.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *ICAPS 2004*, 161–170.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Herstein, I. N. 1975. *Topics in algebra*. Xerox College Pub.

Jerrum, M. 1986. A compact representation for permutation groups. *J. Algorithms* 7(1):60–78.

Junttila, T., and Kaski, P. 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. In Applegate, D.; Brodal, G. S.; Panario, D.; and Sedgewick, R., eds., *Workshop on Algorithm Engineering and Experiments*, ALENEX07, 135–149. SIAM.

Junttila, T. 2003. *On the symmetry reduction method for Petri nets and similar formalisms*. Helsinki University of Technology; Teknillinen korkeakoulu.

Katz, M., and Domshlak, C. 2008. Structural patterns heuristics via fork decomposition. In *ICAPS 2008*, 182–189.

Knoblock, C. A. 1994. Automatically generating abstractions for planning. *AIJ* 68(2):243–302.

Knuth, D. E. 1991. Efficient representation of perm groups. *Combinatorica* 11(1):33–43.

Mathon, R. 1979. A note on the graph isomorphism counting problem. *Information Processing Letters* 8:131–132.

McKay, B. D., and Piperno, A. 2014. Practical graph isomorphism, {II}. *Journal of Symbolic Computation* 60(0):94 – 112.

Noether, E. 1927. Abstrakter aufbau der idealtheorie in algebraischen zahl- und funktionenkörpern. *Mathematische Annalen* 96:26–61.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting problem symmetries in state-based planners. In *AAAI 2011*.

Rintanen, J. 2003. Symmetry reduction for SAT representations of transition systems. In *ICAPS 2003*, 32–41.

Shleyfman, A.; Katz, M.; Helmert, M.; Sievers, S.; and Wehrle, M. 2015. Heuristics and symmetries in classical planning. In *AAAI 2015*, 3371–3377.

Sievers, S.; Wehrle, M.; Helmert, M.; and Katz, M. 2015a. An empirical case study on symmetry handling in cost-optimal planning as heuristic search. In *KI 2015*, 166–180.

Sievers, S.; Wehrle, M.; Helmert, M.; Shleyfman, A.; and Katz, M. 2015b. Factored symmetries for merge-and-shrink abstractions. In *AAAI 2015*, 3378–3385.

Sievers, S.; Röger, G.; Wehrle, M.; and Katz, M. 2017. Structural symmetries of the lifted representation of classical planning tasks. In *HSDIP 2017*.

Starke, P. 1991. Reachability analysis of petri nets using symmetries. *Journal of Mathematical Modelling and Simulation in Systems Analysis* 8(4/5):293–304.

Wehrle, M.; Helmert, M.; Shleyfman, A.; and Katz, M. 2015. Integrating partial order reduction and symmetry elimination for cost-optimal classical planning. In *IJCAI 2015*, 1712–1718.

Zemlyachenko, V. N.; Korneenko, N. M.; and Tyshkevich, R. I. 1985. Graph isomorphism problem. *Journal of Soviet Mathematics* 29(4):1426–1481.