Journal of Artificial Intelligence Research 70 (2021) 1183-1221 Submitted 07/2020; published 03/2021

Computational Complexity of Computing Symmetries in Finite-Domain Planning

Alexander Shleyfman SHLEYFMAN.ALEXANDER@GMAIL.COM
Department of Mechanical and Industrial Engineering
University of Toronto
Canada

Peter Jonsson PETER.JONSSON@LIU.SE
Department of Computer and Informaton Science
Linköping University
Sweden

Abstract

Symmetry-based pruning is a powerful method for reducing the search effort in finite-domain planning. This method is based on exploiting an automorphism group connected to the ground description of the planning task – these automorphisms are known as *structural symmetries*. In particular, we are interested in the StructSym problem where the generators of this group are to be computed. It has been observed in practice that the StructSym problem is surprisingly easy to solve. We explain this phenomenon by showing that StructSym is **GI**-complete, i.e., the graph isomorphism problem is polynomial-time equivalent to it and, consequently, solvable in quasi-polynomial time. This implies that it is solvable substantially faster than most computationally hard problems encountered in AI. We accompany this result by identifying natural restrictions of the planning task and its causal graph that ensure that StructSym can be solved in polynomial time. Given that the StructSym problem is **GI**-complete and thus solvable quite efficiently, it is interesting to analyse if other symmetries (than those that are encompassed by the StructSym problem) can be computed and/or analysed efficiently, too. To this end, we present a highly negative result: checking whether there exists an automorphism of the state transition graph that maps one state s into another state t is a **PSPACE**-hard problem and, consequently, at least as hard as the planning problem itself.

1. Introduction

Symmetries in a (natural or artificial) system can, mathematically speaking, be viewed as the invariance of the system under certain transformations. When systems exhibit symmetric structure, it is often possible to exploit this algebraic property to various ends. One of the main usages of symmetries in computer science is *symmetry breaking* – it is a prominent method for search-space reduction. The main idea behind symmetry breaking is to divide the search space into clusters of similar states which allows the exploration of only one representative state per cluster. The exploitation of this technique together with some suitable search strategy may significantly reduce the total search effort.

We concentrate on symmetry breaking in finite-domain representation (FDR) planning in this article. FDR planning is an interesting testbed for evaluating search-space reduction methods since there is an exponentially large number of states in the search-space. Therefore, the search problem has high complexity (it is known to be **PSPACE**-complete,

Bäckström & Nebel, 1995), and there are a multitude of problems that can be formulated within this framework. Thus, task simplification becomes highly relevant and many such methods have been proposed. Examples include partial order reduction (Wehrle & Helmert, 2012), relevance analysis (Haslum, Helmert, & Jonsson, 2013), symmetry based reductions (Röger, Sievers, & Katz, 2018), symmetry breaking, *etc.* In this work we concentrate on symmetry breaking in FDR planning. Research on this topic has been intensive and has led to numerous results (Starke, 1991; Emerson & Sistla, 1996; Fox & Long, 1999; Rintanen, 2003; Pochter, Zohar, & Rosenschein, 2011; Domshlak, Katz, & Shleyfman, 2012; Gnad, Torralba, Shleyfman, & Hoffmann, 2017; Fišer, Torralba, & Shleyfman, 2019). These results rely on the computation of a certain kind of symmetries of the state transition graph known as *automorphisms*. An automorphism of a graph is a bijective function from vertices to vertices that maps edges into edges and non-edges into non-edges.

One of the problems with the approach is that symmetry detection is a complex problem. One may illustrate this by the symmetries of the state transition graph of an FDR planning task. Assume we want to compute all automorphisms of this graph. Given an instance with n variables and domain size d, the state space contains d^n vertices and up to $(d^n)! \approx (d^n/\mathrm{e})^{d^n}$ automorphisms. In many cases, it is sufficient to work with generators of the automorphism group; however, up to $d^n - 1$ generators are still required in the worst case. As a response to this, research has focused on studying symmetries defined on compact representations of the state space. The first notion of this kind of symmetries for FDR planing was proposed by Pochter et al. (2011) and later refined by Domshlak et al. (2012). Shleyfman, Katz, Helmert, Sievers, and Wehrle (2015) introduced the notion of *structural symmetries* that captures the previously proposed concepts, and additionally can be derived from a given planning task in a straightforward declarative manner.

Let StructSym denote the problem of computing the generators of the structural symmetries of FDR planning tasks. The main result of this article shows that StructSym is a **GI**-complete problem, i.e., the *graph isomorphism* problem is polynomial-time equivalent to it. The graph isomorphism problem has been intensively studied in complexity theory due to its important applications, but it has resisted all attempts both at identifying polynomial-time algorithms for it and at proving it being **NP**-hard. This may be viewed as "bad news", since there is a fair chance that the computation of structural symmetries cannot be performed in polynomial-time. However, StructSym is **GI**-complete and, due to recent algorithmic results by Babai (2015)[1], solvable in *quasi-polynomial* time. A problem X is solvable in quasi-polynomial time if there exists a fixed constant c and an algorithm for X running in $O(2^{(\log n)^c})$ time. Just as the name suggests, problems solvable in quasi-polynomial time may be viewed as problems that are very close to being solvable in polynomial time, and it is highly conceivable that no **NP**-hard problem can be solved in quasi-polynomial time. We note that many computational problems that appear in AI have much higher time complexity under plausible assumptions. The *strong exponential time hypothesis* (SETH) was introduced by Impagliazzo and Paturi (2001); it asserts that the Boolean satisfiability problem cannot be solved in $2^{(1-\epsilon)n}$ time for any $\epsilon > 0$. This hypothesis implies that many computational problems cannot be solved in $2^{o(n)}$ time and,

1. The history behind this result is a bit complicated since an error in the original proof was found by Helfgott and it was later corrected by Babai. More details can be found in the reports by Helfgott, Bajpai, and Dona (2017) and Babai (2016).

consequently, not in quasi-polynomial time. One may, for example, note that STRIPS planning can be solved in $O(4^n \cdot \text{poly}(||\Pi||))$ time (where n is the number of variables and $||\Pi||$ denotes the size of the input instance), yet not in $O(2^{cn})$ time for any $c < 1$ under SETH (Bäckström & Jonsson, 2017). In light of this, our result should be viewed as "good news" and the result is in line with empirical observations: the computation of structural symmetries has proven to be very efficient in practice (Domshlak et al., 2012; Sievers, Röger, Wehrle, & Katz, 2019).

Given that StructSym is solvable in quasi-polynomial time, it is still interesting to identify subclasses that allow for polynomial-time computation. It may, for instance, be the case that StructSym contains so large polynomial-time solvable fragments that the need for the quasi-polynomial algorithm becomes rare in practice. This would imply that simpler and/or more efficient algorithms can frequently be used. One idea for identifying tractable fragments has its roots in the work by Shleyfman (2019). He addresses certain connections between planning tasks, their corresponding causal graphs, and structural symmetries. This work mostly concentrate on algebraic connections and not so much on computational implications. We complement these results by showing that there are interesting connections also from the computational side. In brief, we use the causal graph for identifying natural restrictions on planning tasks so that the computation of structural symmetries can be performed in polynomial time.

We have earlier discussed that computing generators for the automorphism group of the full state transition graph is a very time-consuming operation since the number of generators is huge. At the same time, knowledge about the automorphisms of this graph may be highly valuable for search-space reduction. In the final part of this article we study the complexity of obtaining partial information about the automorphisms of the state transition graph. We consider the following natural problem: given an FDR planning task and two states s, t, is there an automorphism of the state transition graph that maps s to t? We prove that this problem is **PSPACE**-hard and, thus, at least as hard as the planning problem itself. This indicates that structural symmetries exhibit a good trade-off between complexity and search-space reduction.

This article is structured as follows. We present some basic information about FDR planning, the mathematical background, and symmetries in Section 2 and an introduction to the Graph Isomorphism problem in Section 3. The **GI**-completeness proof can be found in Section 4, we treat polynomial-time solvable special cases in Section 5, and we study the complexity of computing more general symmetries in Section 6. We conclude the paper with a brief discussion in Section 7. Some results presented in Section 4 are based on a previous conference publication (Shleyfman, 2019).

2. Preliminaries

This section describes some background material that is used throughout the article: we cover planning, group theory, and structural symmetries in the forthcoming three sections. Before we begin, we need to introduce some basic mathematical concepts. A *directed graph* (*digraph* for short) 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. 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. A *labeled graph* assumes that each edge has the form $\langle n_1, n_2; l\rangle$ where n_1, n_2 are vertices, $l \in L$ is a label, and L is a label set. In what follows, each graph that is not labeled is considered to be a *simple graph*, that is, graph with no loops and no parallel edges. A *colored graph* assumes a color function on the graph's vertices, $\mathsf{col} : N \to \mathbb{N}$. A *path* in a graph is a finite sequence of distinct edges which joins a sequence of vertices. A *simple cycle* is a non-empty path where the only coinciding vertices are the first and last one. For a vertex n in the undirected graph $\mathcal{G} = \langle N, E\rangle$ let $\deg(n)$ denote the *degree* of n, i.e. $|\{\{n', n\} \in E \mid n' \in N\}|$, the degree of the graph $\mathcal{G}$ is $\deg(\mathcal{G}) = \max\{\deg(n) \mid n \in N\}$. To avoid unnecessary reminders, the notion of deg of directed graphs is introduced in the beginning of Section 5. Finally, we write $i \in [k]$ as a shortcut for $i \in \{1, 2, \ldots, k\}$, $k \in \mathbb{N}$.

2.1 Deterministic Planning Tasks

A *planning task* consists of a description of states (via variables and their associated domains), a set of state modifying actions, an initial state, and a goal. We adopt the widely used FDR language for describing such tasks (e.g., Bäckström & Klein, 1991; Bäckström & Nebel, 1995; Helmert, 2009). Notation and terminology mostly follow Helmert and Domshlak (2009).

Definition 1. *An* FDR planning task *is a 4-tuple* $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G\rangle$*, where:*

- $\mathcal{V}$ *is a finite set of* state variables*, with each variable* $v \in \mathcal{V}$ *being associated with its finite domain* $\mathcal{D}(v)$*.*
  - *Each complete assignment to* $\mathcal{V}$ *is called a* state*, and* $S = \bigtimes_{v\in\mathcal{V}} \mathcal{D}(v)$ *is called the* state space *of* Π*.*
  - *A* fact *is a pair* $\langle v, d\rangle$*, where* $v \in \mathcal{V}$ *and* $d \in \mathcal{D}(v)$*, thus each variable* $v \in \mathcal{V}$ *can be represented as a set of mutually exclusive facts* $\mathcal{F}_v = \{\langle v, d\rangle \mid d \in \mathcal{D}(v)\}$*. The set of all facts will be denoted by* $\mathcal{F}$*. A* partial variable assignment *is a set of facts, each corresponding to a different variable. For a partial assignment (or a state)* p *to a set of variables* $\mathcal{V}$*,* $vars(p) \subseteq \mathcal{V}$ *denotes the subset of variables instantiated by* p*, and, for* $v \in vars(p)$*,* $p[v]$ *denotes the value provided by* p *to the variable* v*.*
- s_0 *is an* initial state*.*
- *The* goal G *is a partial assignment to* $\mathcal{V}$*: A state* s *is a* goal state *iff* $G \subseteq s$*.*
- $\mathcal{O}$ *is a finite set of* actions*, each* action o *is given by a tuple* $\langle \mathsf{pre}(o), \mathsf{eff}(o), \mathsf{cost}(o)\rangle$*, where*
  - $\mathsf{pre}(o)$ *and* $\mathsf{eff}(o)$ *are partial states and denote the* precondition *and the* effect *of* o*, respectively, and*
  - $\mathsf{cost} : \mathcal{O} \to \mathbb{R}^{0+}$ *a function that assigns non-negative costs to actions.*

The semantics of FDR actions is as follows: An action o is applicable in a state s iff $s[v] = \mathsf{pre}(o)[v]$ for all $v \in vars(\mathsf{pre}(o))$. If o is applicable in state s, then the application of o changes the value of each $v \in vars(\mathsf{eff}(o))$ to $\mathsf{eff}(o)[v]$, and the resulting state is denoted

by $s[\![o]\!]$. Otherwise, the result is undefined. By $\mathcal{O}(s) \subseteq \mathcal{O}$ we denote the set of actions applicable in s.

A sequence of actions $\pi = \langle o_1, \ldots, o_n \rangle$ is a *path* that starts in s if, for $i \in [n]$, o_i is applicable in state $s[\![o_1]\!] \ldots [\![o_{i-1}]\!]$. The state resulting from applying such a sequence of actions π in s is denoted by $s[\![\pi]\!]$, and the cost of π is the cumulative cost of actions in the sequence, i.e., $\mathsf{cost}(\pi) = \sum_{i=1}^{n} \mathsf{cost}(o_i)$. A path that starts in s_0 and achieves $G \subseteq s[\![\pi]\!]$ is called a *plan* for Π. A plan π is *optimal* if the sum of action costs in π is minimal among all plans. (Optimal) planning aims at finding an (optimal) plan for Π.

A planning task Π implicitly induces a *state transition graph* $\mathcal{T}_\Pi$. This is a labeled digraph $\mathcal{T}_\Pi = \langle S, E \rangle$, whose vertexes S are the states of Π, the set of labeled arcs $E = \{\langle s, s[\![o]\!]; o \rangle \mid s \in S, o \in \mathcal{O}(s)\}$[2] is induced by the actions of Π, and the cost function of the labeled arcs $\mathsf{cost} : \mathcal{O} \to \mathbb{R}^{0+}$ being induced by the action cost function as $\mathsf{cost}(s, s[\![o]\!]; o) = \mathsf{cost}(o)$. The problem of finding a plan for Π is equivalent to finding a directed path in $\mathcal{T}_\Pi$ from the vertex s_0 to a vertex that corresponds to a goal state.

Planning tasks are often structurally complex and difficult to comprehend. One simplified way of visualizing them is *causal graphs* (e.g., Knoblock, 1994; Bacchus & Yang, 1994; Jonsson & Bäckström, 1998; Domshlak & Brafman, 2002). We follow the definition given by Helmert (2004): the *causal graph* of a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$ is a directed graph $CG_\Pi = \langle N, E \rangle$, where $N = \{n_v \mid v \in \mathcal{V}\}$ and $(n_u, n_v) \in E$ if $u \neq v$ and there exists $o \in \mathcal{O}$, such that $u \in \mathit{vars}(\mathsf{pre}(o)) \cup \mathit{vars}(\mathsf{eff}(o))$ and $v \in \mathit{vars}(\mathsf{eff}(o))$. 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.

2.2 Groups

Groups are frequently used in mathematics and elsewhere for describing symmetries of various kinds. We refer the reader to Herstein (1975) for a more detailed exposition. A *group* Γ is a pair $(X_\Gamma, \circ)$ where X_Γ is a set and $\circ$ is a total function from $X_\Gamma \times X_\Gamma$ to X_Γ that has the following properties.

1. For all $a, b, c \in X_\Gamma$, $(a \circ b) \circ c = a \circ (b \circ c)$.
2. There exists an element $e \in X_\Gamma$ (referred to as the *identity* element) such that, for every element a in X_Γ, $e \circ a = a \circ e = a$. We denote this element id_Γ.
3. For each a in X_Γ, there exists an element b in X_Γ such that $a \circ b = b \circ a = e$, where e is the identity element. The element b is often denoted a^{-1}.

For increased readability, we sometimes write $a \in \Gamma$ instead of $a \in X_\Gamma$.

A concrete example (that we will encounter several times in the sequel) is the *cyclic group* of order n. Consider a group $\mathbb{Z}_n$ whose set of elements is the integers $\{0, \ldots, n-1\}$ and the operation is addition modulo n. The identity element of $\mathbb{Z}_n$ is trivial, i.e., $e = 0$. The inverse of element a is $a^{-1} = (n - a) \mod n$, since $(a + a^{-1}) \mod n = (a + n - a) \mod n = 0$. A cyclic group of order n is a group that is isomorphic to $\mathbb{Z}_n$.

2. It is important to note that $\mathcal{T}_\Pi$ may contain a pair of arcs $(s, s[\![o]\!]; o)$ and $(s, s[\![o']\!]; o')$ such that $s[\![o]\!] = s[\![o']\!]$ and $o \neq o'$.

Let $\Gamma = (X_\Gamma, \circ)$ denote an arbitrary group. A tuple $\Sigma = (X_\Sigma, \circ|_{X_\Sigma})$ is called a subgroup of Γ if $X_\Sigma \subseteq X_\Gamma$ and $(X_\Sigma, \circ|_{X_\Sigma})$ is a group where $\circ|_\Sigma$ is the restriction of $\circ$ to $X_\Sigma \times X_\Sigma$. We let $\Sigma \leq \Gamma$ denote that Σ is a subgroup of Γ. Given a subset $X_S \subseteq X_\Gamma$, we let $Gen(X_S)$ denote the elements $g \in X_\Gamma$ that can be finitely expressed by using the elements of X_S together with the operations $\circ$ and $\cdot^{-1}$. The set $Gen(X_S)$ always defines a subgroup of Γ. If $X_\Gamma = Gen(X_S)$, then we say that X_S *generates* Γ.

We now give a brief overview of automorphism groups that play a key role in this article. Let $\mathcal{G}_1 = \langle N_1, E_1 \rangle$ and $\mathcal{G}_2 = \langle N_2, E_2 \rangle$ be two (di-)graphs, and let $\sigma : N_1 \to N_2$ be a bijection. We say that σ is a *graph isomorphism* (or simply isomorphism) when $(n, n') \in E_1$ if and only if $(\sigma(n), \sigma(n')) \in E_2$. The graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are called *isomorphic*. The idea behind graph isomorphisms can easily be lifted to other kinds of mathematical structures. An *automorphism* is an isomorphism from a mathematical object to itself. The automorphisms of, for instance, a graph $\mathcal{G} = \langle N, E \rangle$ are closed under function composition and every automorphism has an inverse which is an automorphism, too. Thus, the automorphisms form a group which we call an *automorphism group* and denote it by $Aut(\mathcal{G})$. The automorphism group of a graph $\mathcal{G} = \langle N, E \rangle$ is known to be generated by a fairly small set of generators that contains at most $|N| - 1$ elements. This can be seen as follows. A *permutation group* is a group $\Gamma = (X_\Gamma, \circ)$ whose elements are permutations (viewed as bijective functions) of some set and whose group operation is function composition. Let $\mathbb{S}_n$ denote the *symmetric group* of order n, i.e. the permutation group containing all possible permutations over n objects. Every group $Aut(\mathcal{G})$ is a subgroup of $\mathbb{S}_{|N|}$ since each automorphism $\sigma \in Aut(\mathcal{G})$ is a permutation of the $|N|$ vertices of the graph $\mathcal{G}$. Jerrum (1986) has presented an algorithm (known as *Jerrum's filter*) that, given a permutation group $\Gamma \leq \mathbb{S}_n$, produces a generating set for Γ of size at most $n - 1$.

When investigating relations between automorphism groups of different structures, one typically exploits various structure-preserving mappings. Let $\Gamma = (X, \circ)$ and $\Sigma = (Y, *)$ be groups, and let $\phi : \Gamma \to \Sigma$ be a mapping. If ϕ satisfies $\phi(a \circ b) = \phi(a) * \phi(b)$ for all $a, b \in \Gamma$, then we say that ϕ is a *homomorphism* of Γ to Σ. One may note that if ϕ is a homomorphism, then ϕ maps the identity element of Γ to the identity element of Σ and $\phi(a^{-1}) = \phi(a)^{-1}$ for all $a \in \Gamma$.

If ϕ is a homomorphism that is bijective, then ϕ is called an *isomorphism*, while Γ and Σ are called isomorphic. We denote this as $\Gamma \cong \Sigma$ or simply $\Gamma = \Sigma$. One may note that if two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic, then their automorphism groups are isomorphic, too.

While bijective homomorphisms are useful for verifying that two groups have the same structure, injective homomorphisms are useful for identifying subgroups. We need the following well-known results (Herstein, 1975).

Theorem 1. *Let Γ and Σ be groups and let $\phi : \Gamma \to \Sigma$ be a homomorphism. If ϕ is an injection, then there exists a subgroup $\Lambda \leq \Sigma$ such that $\Lambda \cong \Gamma$. In this case, we will simply write $\Gamma \leq \Sigma$.*

The *kernel* of a group homomorphism is an important concept in this context. In a certain sense, the kernel measures the "non-injectivity" of a homomorphism.

Definition 2. *Let Γ and Σ be two groups, and let $\phi : \Gamma \to \Sigma$ be a group homomorphism. The* kernel *of ϕ ($ker(\phi)$) is the set $\phi^{-1}(id_\Sigma)$ or, equivalently, the set $\{x \in \Gamma \mid \phi(x) = id_\Sigma\}$.*

Theorem 2. *Let Γ and Σ be groups, and let $\phi : \Gamma \to \Sigma$ be a homomorphism. The homomorphism ϕ is injective if and only if $ker(\phi) = \{id_\Gamma\}$.*

2.3 Structural Symmetries

This section introduces the notion of *structural symmetries*. In short, a structural symmetry is a relabelling of a given planning task that preserves the structure of the underlying state transition graph. We follow the definition of structural symmetries for FDR planning tasks as defined by Wehrle *et al.* (2015). For a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$, let $\mathcal{F}$ be the set of Π's facts, and let $\mathcal{F}_\mathcal{V} = \{\mathcal{F}_v \mid v \in \mathcal{V}\}$ be the set of sets of facts attributed to each variable in $\mathcal{V}$.

Definition 3. *Let $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$ be a planning task, and let $\sigma : \mathcal{F} \cup \mathcal{O} \to \mathcal{F} \cup \mathcal{O}$ be a function that is a permutation on $\mathcal{F}$ and a permutation on $\mathcal{O}$. We say that σ is a* structural symmetry *if the following hold:*

1. $\sigma(\mathcal{F}_\mathcal{V}) = \mathcal{F}_\mathcal{V}$,
2. $\sigma(\mathcal{O}) = \mathcal{O}$, *and, for all* $o \in \mathcal{O}$, $\sigma(\mathsf{pre}(o)) = \mathsf{pre}(\sigma(o))$, $\sigma(\mathsf{eff}(o)) = \mathsf{eff}(\sigma(o))$, *and* $\mathsf{cost}(\sigma(o)) = \mathsf{cost}(o)$, *and*
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 down to the level of action labels and facts. For example, let s be a partial state. Thus, s can be represented as a set of facts. Applying σ to s will result in a partial state s', such that 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 $Aut(\Pi)$. We have the following central computational problem.

Problem. Structural Symmetry (StructSym)
Instance. A planning task Π.
Output. A set of generators $\mathcal{S}$ for $Aut(\Pi)$ such that

$$|\mathcal{S}| \leq |\mathcal{O}| + \sum_{v \in \mathcal{V}} (|\mathcal{D}(v)| + 1) \,.$$

The bound on the number of generators matches the number of vertices of the colored graph that we introduce next.

The final concept we need is the *problem description graph* (PDG). PDGs were introduced by Pochter *et al.* (2011), and later reformulated for different purposes by Domshlak *et al.* (2012) and Shleyfman *et al.* (2015). Intuitively, a PDG is a colored graph that preserves some structural properties of a given planning task. This properties allow to transform the symmetries found in PDG into symmetries of the transition graph of this task.

In previous works, shifted action costs ($c + \mathsf{cost}(o)$, where $c = 3$ or 4) were used as colors to distinguish between actions of different costs. This was no problem since costs in

this setting were used only for distinguishing vertices. In this article, however, we need to replace colors with graph gadgets (see Section 4) and this requires us to use colors that are (1) integers and (2) these integers are not too large. To this end, we introduce an auxiliary map on action costs. Note that this mapping of colors into integers does not change any results that are given for PDGs in the literature.

Let $C_{\mathcal{O}}$ be the number of unique costs of actions in the task Π, i.e. $C_{\mathcal{O}} = |\{\mathsf{cost}(o) \mid o \in \mathcal{O}\}|$. We define a function $\mathsf{ord} : \mathcal{O} \to [C_{\mathcal{O}}]$ to be be an order preserving function with respect to costs, i.e., for $o_1, o_2 \in \mathcal{O}$ holds that $\mathsf{cost}(o_1) \leq \mathsf{cost}(o_2)$ implies that $\mathsf{ord}(o_1) \leq \mathsf{ord}(o_2)$. The main properties of the function ord that we are interested in are

1. for $o_1, o_2 \in \mathcal{O}$ it holds $\mathsf{cost}(o_1) = \mathsf{cost}(o_2)$ if and only if $\mathsf{ord}(o_1) = \mathsf{ord}(o_2)$, and
2. for each $o \in \mathcal{O}$ it holds that $\mathsf{ord}(o) \leq C_{\mathcal{O}}$.

In the following definition the function ord will allow us to distinguish between action costs, as well as operating within a reasonable bound on the number of colors we are coloring our graph with.

Definition 4. *Let* Π *be a* FDR *planning task. The* problem description graph *(PDG) of* Π *is the colored directed graph* $PDG_\Pi = \langle N, E, \mathsf{col}\rangle$ *with nodes*

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

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* $N_{\mathcal{O}} = \{n_o^{\mathsf{pre}}, n_o^{\mathsf{eff}} \mid o \in \mathcal{O}\}$. *The node colors are*

$$\mathsf{col}(n) = \begin{cases} 0 & \text{if } n \in N_{\mathcal{V}} \\ 1 & \text{if } n \in \bigcup\limits_{v \in \mathcal{V}} N_{\mathcal{D}(v)} \text{ and } \langle v, d\rangle \in G \\ 2 & \text{if } n \in \bigcup\limits_{v \in \mathcal{V}} N_{\mathcal{D}(v)} \text{ and } \langle v, d\rangle \notin G \\ 3 & \text{if } n_o^{\mathsf{pre}} \in N_{\mathcal{O}} \\ 4 + \mathsf{ord}(o) & \text{if } n_o^{\mathsf{eff}} \in N_{\mathcal{O}}, \end{cases}$$

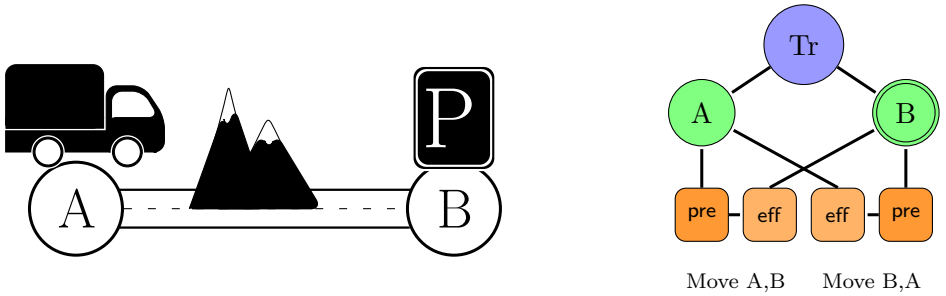
and edges

$$E = \bigcup_{v \in \mathcal{V}} E^v \cup \bigcup_{o \in \mathcal{O}} \left(E_o^{\mathsf{pre}} \cup E_o^{\mathsf{eff}} \cup \{n_o^{\mathsf{pre}}, n_o^{\mathsf{eff}}\} \right),$$

where $E^v = \{\{n_v, n_{\langle v,d\rangle}\} \mid d \in \mathcal{D}(v)\}$, $E_o^{\mathsf{pre}} = \{\{n_o^{\mathsf{pre}}, n_{\langle v,d\rangle}\} \mid \langle v, d\rangle \in \mathsf{pre}(o)\}$, *and* $E_o^{\mathsf{eff}} = \{\{n_{\langle v,d\rangle}, n_o^{\mathsf{eff}}\} \mid \langle v, d\rangle \in \mathsf{eff}(o)\}\}$ *(see Figure 1).*

Pochter *et al.* (2011) have considered *PDG symmetries*, i.e. symmetries of $\mathcal{T}_\Pi$ that are induced by $Aut(PDG_\Pi)$. Shleyfman et al. (2015) have shown that the structural symmetries of Π (given some minor condition that are fulfilled via a preprocessing step) are in a one-to-one correspondence with the PDG symmetries of Π. Later on, Shleyfman (2020) showed that the following result holds even with no preprocessing.

Theorem 3. *Let* Π *be an* FDR *planning task. Then,* $Aut(PDG_\Pi)$ *and* $Aut(\Pi)$ *are isomorphic.*



(a) (b)

Figure 1: Illustration of the example of a planning problem (a) planning problem $\Pi = \langle\{\langle \mathrm{Tr}, A\rangle, \langle \mathrm{Tr}, B\rangle\}, \{o_{\text{Move A,B}}, o_{\text{Move B,A}}\}, \{\langle \mathrm{Tr}, A\rangle\}, \{\langle \mathrm{Tr}, B\rangle\}\rangle$, with the actions $o_{\text{Move A,B}} = \langle\{\langle \mathrm{Tr}, A\rangle\}, \{\langle \mathrm{Tr}, B\rangle\}, 1\rangle$ and $o_{\text{Move B,A}} = \langle\{\langle \mathrm{Tr}, B\rangle\}, \{\langle \mathrm{Tr}, A\rangle\}, 1\rangle$ and (b) its PDG.

3. The Graph Isomorphism Problem

The goal of Sections 3 and 4 is to prove that the complexity of computing structural symmetries is polynomial-time equivalent to the Graph Isomorphism (GI) problem. We introduce GI and some related problems in this section while our reductions are presented in Section 4. GI is the well-known decision problem of determining whether two finite undirected graphs are isomorphic or not.

Problem. Graph Isomorphsim
Instance. Two undirected graphs $\mathcal{G}_1$ and $\mathcal{G}_2$.
Question. Do $\mathcal{G}_1$ and $\mathcal{G}_2$ admit an isomorphism?

This problem has a special status in complexity theory since it is neither known to be solvable in polynomial time nor to be NP-complete. We elaborate upon this in Section 3.1. The reductions in Section 4 are not directly based on GI but on two auxiliary problems AGEN and CAGEN. These problems and some connections with GI are discussed in Section 3.2.

3.1 Complexity Background

We begin this section by recapitulating the definition of Turing reductions. Such reductions will be the basis for many of the forthcoming results. A *Turing reduction* from a decision problem X to a decision problem Y is an oracle machine which decides problem X given an oracle for the problem Y. In other words, it can be viewed as an algorithm that solves X by exploiting an algorithm for solving Y as a subroutine. This idea can analogously be used for function problems. Furthermore, we say that a problem X is *polynomial-time Turing reducible* to Y if the algorithm for solving X runs in polynomial time and we denote this by $X \leq_p Y$. An important consequence is that if $X \leq_p Y$ and Y is polynomial-time solvable, then X is polynomial-time solvable, too. If two problems are polynomial-time reducible to

each other, then we say that these problems are *polynomial-time Turing equivalent* and we denote this by $X =_p Y$. All reductions in this article are Turing reductions so we typically use the term reduction instead of Turing reduction.

The GI problem defines a complexity class in a natural way: we let **GI** denote the set of problems that are polynomial-time reducible to GI and we say that a problem is **GI**-complete if it is polynomial-time equivalent to GI. It is well-known that GI is in **NP** and consequently that **GI** $\subseteq$ **NP**, but no polynomial-time algorithm is known for GI. However, there exist surprisingly efficient algorithms for GI and GI is polynomial-time solvable for important classes of graphs such as trees (Kelly, 1957), bounded-degree graphs (Luks, 1982), and planar graphs (Hopcroft & Wong, 1974). On the other hand, every attempt to prove that GI is **NP**-complete has failed and it appears that GI is of a much more constrained nature than other **NP**-complete problems (Schöning, 1987). Assuming that **P** $\neq$ **NP**, Ladner's theorem (Ladner, 1975) shows that there are problems (referred to as **NP**-*intermediate*) in **NP** that are neither polynomial-time solvable nor **NP**-complete. One may speculate that GI is one of the problems that are **NP**-intermediate: supporting evidence of this hypothesis is the fact that GI cannot be NP-complete unless the polynomial time hierarchy collapses to its second level (Schöning, 1987).

From an algorithmic point of view, Babai (2015) has shown that GI can be solved in *quasi-polynomial time*, i.e., in time bounded by $2^{((\log n)^c)}$ for some fixed c. Quasi-polynomiality works very well in combination with polynomial-time reductions: if $X \leq_p Y$ and Y is solvable in quasi-polynomial time, then so is X (with the immediate implication that every problem in **GI** is solvable in quasi-polynomial time). This property is, for example, not shared by problems that are solvable in single-exponential $2^{O(n)}$ time (such as the Boolean satisfiability problem): the fact that $X \leq_p Y$ and Y is solvable in single-exponential time does not imply that X is solvable in single-exponential time.

3.2 Problems Related to GI

We will now introduce two problems that are related to graph isomorphisms: AGEN and CAGEN. Instead of directly using GI for proving the complexity of the structural symmetry problem StructSym, we will use CAGEN for proving that StructSym is a member of GI and AGEN for proving **GI**-hardness of StructSym.

In the automorphism generation problem AGEN the input is an undirected graph $\mathcal{G}$ and the output is a set of generators for $Aut(\mathcal{G})$. The following is a first attempt to define this problem.

Problem. Automorphism Generation (AGEN)
Instance. An undirected graph $\mathcal{G}$.
Output. A set of generators for the automorphisms of $\mathcal{G}$.

We see that this definition is unsatisfactory since there is no bound on the size of generators. This implies that the output may contain $n!$ generators in the worst case: the empty graph on n vertices has $n!$ automorphisms and this set is trivially a generating set. We thus reformulate AGEN as follows.

Problem. Automorphism Generation (AGEN)
Instance. An undirected graph $\mathcal{G} = \langle N, E \rangle$.
Output. A set $|\mathcal{S}|$ of generators for the automorphisms of $\mathcal{G}$ such that $|\mathcal{S}| \leq |N|$.

It is known that the second version of the problem is **GI**-complete via a result by Mathon (1979). Note that the bound $|N|$ is presented in the proof of the result but it is not stated in the formulation of the theorem. We will exclusively use the second formulation of AGEN in the sequel.

PDGs are colored graphs so we will be highly interested in *color-preserving* automorphisms in the sequel. Consider a colored graph $\mathcal{G} = \langle N, E, \mathsf{col} \rangle$. An automorphism α of $\mathcal{G}$ is *color-preserving* if $\mathsf{col}(\alpha(n)) = \mathsf{col}(n)$ for every $n \in N$. The AGEN problem can readily be generalized to colored graphs.

Problem. Colored Automorphism Generation (CAGEN)
Instance. An undirected colored graph $\mathcal{G} = \langle N, E \rangle$.
Output. A set $\mathcal{S}$ of generators for the color-preserving automorphisms of $\mathcal{G}$ such that $|\mathcal{S}| \leq |N|$.

We continue by taking a closer look at the AGEN problem. As pointed out earlier, Mathon (1979) has proved the following result.

Theorem 4. GI $=_p$ AGEN.

Mathon's reductions are based on colored graphs, and he exploits methods for reducing problems on colored graphs to problems on non-colored graphs. Many details are unfortunately excluded in Mathon's two-page paper so there is not a full specification of how to handle colored graphs. This is problematic for several reasons in our setting. First of all, PDGs are indeed colored graphs and we cannot exploit algorithms for (uncolored) GI unless we know exactly how to simulate colours. This is obvious from the correctness point of view, yet it is also important from the complexity point of view: we do not want our constructions to blow up graph size too much. Secondly, we will later on (in Section 5) study PDGs with bounded degree. Once again, we need to know how the simulation of colours work, since we need to keep the degree bounded. One should note that Mathon's proof has caused some confusion in the literature, for example, Ghosh and Kurur (2014, p. 5) attempt to prove CAGEN $=_p$ GI by adding one-path labels to the graph (see Figure 2).

We continue by explaining the problem with Mathon's proof in greater detail – this will make it easier for the reader to appreciate the construction that we present in Section 4. We begin with a few definitions. Let $\mathcal{G} = \langle N, E \rangle$ denote an arbitrary graph and let Γ be its automorphism group $Aut(\mathcal{G})$. For a vertex $n \in N$ we denote by Γ_n the automorphism group that fixes n, i.e., $\{\sigma \in \Gamma \mid \sigma(n) = n\}$. The group $\Gamma_{n_1,\ldots,n_k}$ is defined by recursively fixing the vertices $n_1, \ldots, n_k \in N$, i.e., $\Gamma_{n_1,n_2} = (\Gamma_{n_1})_{n_2}$. For $m \in \mathbb{N}$ we define a *path graph* by $P_m = \langle N_m, E_m \rangle$, where

$$N_m = \{n_i \mid i \in [m]\} \text{ and } E_m = \{\{n_i, n_{i+1}\} \subseteq N_m \mid i \in [m-1]\}.$$

A path graph on vertices $n_0, \ldots, n_m$ is depicted in the lower part of Figure 2. For a graph $\mathcal{G} = \langle N, E \rangle$, and a vertex $n \in N$ and a positive number $m \in \mathbb{N}$, we define $\mathcal{G}_{\langle n, P_m \rangle} = \langle N \cup P_m, E \cup E_m \cup \{n, n_1\} \rangle$.

Let us now consider the beginning of the proof of the Theorem in Mathon (1979).

> **Proof.** We require the definition of graph labels. Let $\mathcal{G}_{n_1,\ldots,n_k}$ denote a copy of $\mathcal{G}$ with unique distinct labels attached to the vertices $n_1, \ldots, n_k \in N$. This can be accomplished for example by adding paths of appropriate length to the n_i's. (...)

The goal of this construction is to assure that two nodes with different labels (which we call *colors*) can never be isomorphic. This is achieved by appending extra structures (in this case paths) to the vertices of the original graph. Formally, Mathon aims at proving $Aut(\mathcal{G}_{n_1,\ldots,n_k}) = \Gamma_{n_1,\ldots,n_k}$ for any set of vertices $n_1, \ldots, n_k \in N$ and $\Gamma = Aut(\mathcal{G})$. However, it is not obvious how to add paths that achieve this. Let us, for instance, consider

$$\mathcal{G}_{n_1,\ldots,n_k} = \mathcal{G}_{\langle n_1, P^1_{m_1}\rangle,\ldots,\langle n_k, P^k_{m_k}\rangle},$$

where $P^i_{m_i} \cap P^j_{m_j} = \emptyset$ for $i, j \in [k], i \neq j$, that is, each path is unique and attached to a specific graph vertex.

Let us look at the graph $\mathcal{G} = \langle\{n_0\}, \emptyset\rangle$. It is easy to see that $\mathcal{G}_{\langle n_0, P_m\rangle} = P_{m+1}$ for any $m \in \mathbb{N}$ (see Figure 2). Note also that the automorphism group of $\mathcal{G}$ is trivial, i.e., $Aut(\mathcal{G}) = \{e\}$. On the other hand, $Aut(P_{m+1}) = \mathbb{Z}_2$ where $\mathbb{Z}_2$ is the cyclic group of order two.

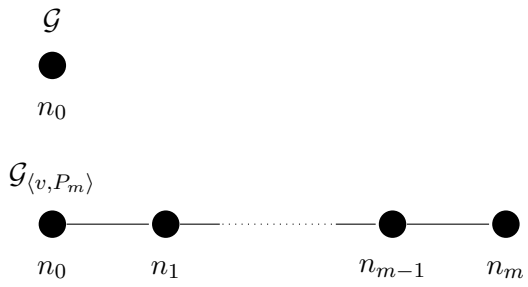


Figure 2: The addition of one path.

From the phrase "This can be accomplished for example by adding paths of appropriate length to the n_i's." it may follow that more than one path can be attached to a specific vertex. Two paths are still not enough here, since for the same graph $G = \langle\{n_0\}, \emptyset\rangle$, the graph that fixes n_0 is $\mathcal{G}_{n_0} = G_{\langle n_0, P^1_{m_1}\rangle, \langle n_0, P^2_{m_2}\rangle}$. Once again, it is easy to see that $Aut(\mathcal{G}_{\langle n_0, P^1_{m_1}\rangle, \langle n_0, P^2_{m_2}\rangle}) = \mathbb{Z}_2$ (see Figure 3).

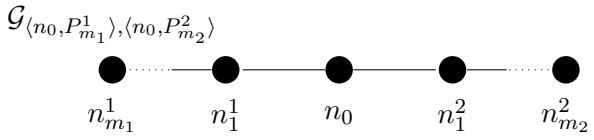


Figure 3: The addition of two paths.

The statement $Aut(\mathcal{G}_{n_1,\ldots,n_k}) = \Gamma_{n_1,\ldots,n_k}$ should hold for three paths attached to a single vertex, i.e.,

$$\mathcal{G}_{n_1,\ldots,n_k} = \mathcal{G}_{\langle n_1,P^1_m\rangle,\langle n_1,P^2_{m+1}\rangle,\langle n_1,P^3_{m+2}\rangle,\ldots,\langle n_k,P^{3k+1}_m\rangle,\langle n_k,P^{3k+2}_{m+1}\rangle,\langle n_k,P^{3k+3}_{m+1+k}\rangle},$$

where $|N| = m$. The proof of this statement, however, is somewhat harder than the construction that we present in the next section and it introduces a larger number of auxiliary vertices.

4. Complexity of Computing Symmetries

The goal of this section is to prove that StructSym is polynomial-time equivalent to GI. The section is divided into two parts where the first part (Section 4.1) presents a reduction from StructSym to GI and the second part (Section 4.2) presents a reduction in the opposite direction. Most of Section 4.2 is devoted to proving that CAGEN $=_p$ GI – the inclusion of StructSym in **GI** is a fairly direct consequence of this result. The **GI**-hardness result in Section 4.2 is based on a reduction from AGEN. To this end, we exploit a method by Zemlyachenko et al. (1985) that transforms any undirected graph $\mathcal{G}$ into a directed graph $\mathcal{G}'$ such that $\mathcal{G}$ and $\mathcal{G}'$ have the same automorphisms, and we introduce a method that transform any directed graph $\mathcal{G}'$ into a planning task whose structural symmetries are in a one-to-one correspondence with the automorphisms of $\mathcal{G}'$.

4.1 Computation of Structural Symmetries

We will now prove that the StructSym problem can be polynomial-time reduced to GI. The basic idea is to view planning tasks as PDGs and to use CAGEN as an intermediate problem. Our aim is to simulate colors in a graph by adding a particular form of subgrapshs with certain properties, denoted *label structures*, that prevent nodes with different colors from being isomorphic, i.e. the label structure $\mathcal{L}^i_k$ can be interpreted as the assignment of color i to vertex n_k. The following definition is the cornerstone of our approach.

Definition 5. *Let $\mathcal{G} = \langle N, E\rangle$ be a undirected graph with $N = \{n_k \mid k \in [m]\}$. We define a* label structure *to be a graph $\mathcal{L}^i_k = \langle N^i_k, E^i_k\rangle$ such that*

$$\begin{aligned} N^i_k &= \{n^k_j \mid j \in [m+4]\},\\ E^i_k &= \{\{n^k_j, n^k_{j+1}\} \mid j \in [m+2]\} \cup\\ &\qquad \{\{n^k_1, n^k_{m+3}\}, \{n^k_{m+1}, n^k_{m+3}\}, \{n^k_i, n^k_{m+4}\}\}, \end{aligned}$$

where $i, k \in [m]$. The attachment of label $\mathcal{L}^i_k$ to vertex $n_k \in N$ is defined to be the graph $\mathcal{G}_{n^i_k} = \langle N \cup N^i_k, E \cup E^i_k \cup \{\{n_k, n^k_{m+3}\}\}\rangle$ (see Figure 4).

Analogously, we define the graph $\mathcal{G}_{n^{i_1}_1,\ldots,n^{i_k}_k}$ to be $\mathcal{G}$ where we have attached labels $\mathcal{L}^{i_t}_{n_t}$ to vertices $n_t \in N$ for all $t \in [k]$, where $i_t \in [m]$. By construction, the order of the vertices is irrelevant. Note also that we, without loss of generality, allow each original vertex $n \in N$ to have at most one label attached to it.

Given an arbitrary graph $\mathcal{G}$ and a graph $\mathcal{G}'$ that is a full assignment of labels to all vertices of $\mathcal{G}$, we see that the degree of $\mathcal{G}'$ is $\max\{4, 1 + \deg \mathcal{G}\}$. Furthermore, $\mathcal{G}'$ contains at

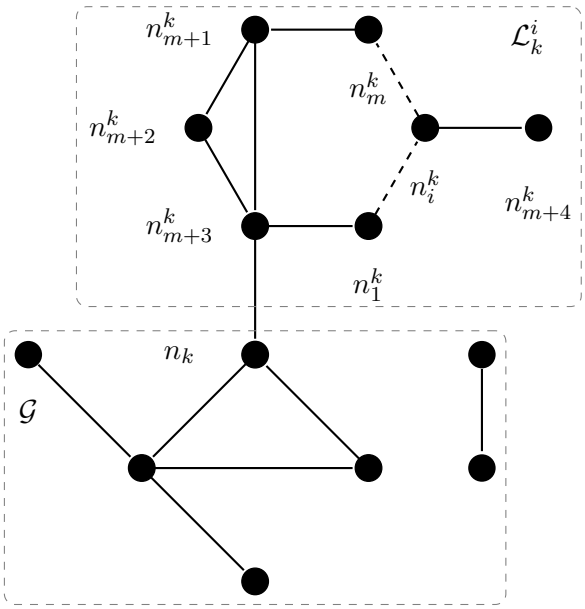


Figure 4: Label $\mathcal{L}^i_k$ is attached to a vertex n_k in some graph $\mathcal{G}$.

most $n^2 + 5n$ vertices if $\mathcal{G}$ contains n vertices. We will now show that labels indeed behave as colors by proving the following result.

Lemma 1. *Let $\mathcal{G} = \langle N, E\rangle$ be an undirected graph with $N = \{n_k \mid k \in [m]\}$ and let $\mathsf{col} : N \to [m]$ be a coloring function. We let $\mathcal{G}_c = \langle N, E, \mathsf{col}\rangle$ denote the colored version of $\mathcal{G}$. Define $\mathcal{G}' = \mathcal{G}_{n_1^{\mathsf{col}(n_1)}, n_2^{\mathsf{col}(n_2)}, \ldots, n_m^{\mathsf{col}(n_m)}}$ to be the uncolored graph with the labels $\mathcal{L}_1^{\mathsf{col}(n_1)}, \mathcal{L}_2^{\mathsf{col}(n_2)}$, $\ldots$, $\mathcal{L}_m^{\mathsf{col}(n_m)}$ attached to all vertices $n_1, n_2, \ldots, n_m$ of the graph $\mathcal{G}$. Then, $Aut(\mathcal{G}_c) = Aut(\mathcal{G}')$.*

The proof is based on two auxiliary lemmas that we present below. Let $\mathcal{G}$ be an undirected graph and let $\mathcal{G}'$ be the graph where some of the vertices of $\mathcal{G}$ has been colored by introducing labels. In the next lemma we prove that there is an injective homomorphism from $Aut(\mathcal{G}')$ to $Aut(\mathcal{G})$ and, consequently, that $Aut(\mathcal{G}')$ is a subgroup of $Aut(\mathcal{G})$ by Theorem 1. Furthermore, this homomorphism preserves the colors that are implied by the labelings. Labels are based on cycles so we frequently need to reason about cycles in graph. This is facilitated by the following definition: let

$$\mathcal{C}_k(N) = \{n \in N \mid n \text{ belongs to a simple cycle of length } k\}.$$

Note that $\mathcal{C}_k(N)$ is not itself a cycle but the set of all vertices in N that are contained in *some* simple cycle of length k.

Lemma 2. *Let $\mathcal{G} = \langle N, E\rangle$ be an undirected graph with $N = \{n_j \mid j \in [m]\}$ and let $\mathcal{G}' = \mathcal{G}_{n_1^{i_1}, n_2^{i_2}, \ldots, n_k^{i_k}}$ be a graph $\mathcal{G}$ with $k \in [m]$ labels $\mathcal{L}_1^{i_1}, \mathcal{L}_2^{i_2}, \ldots, \mathcal{L}_k^{i_k}$ attached to the vertices $n_1, n_2, \ldots, n_k$, respectively. Then,*

1. *the function* $\phi : Aut(\mathcal{G}') \leq Aut(\mathcal{G})$ *defined such that* $\phi(\sigma) = \sigma|_N$ *is an injective homomorphism, and*
2. *for arbitrary* $\sigma \in Aut(\mathcal{G}')$ *and for all* $t, t' \in [k]$*, it holds that* $\sigma(n_t) = n_{t'} \implies i_t = i_{t'}$.

Proof. We begin by making a few useful observations. Denote $\mathcal{G}' = \langle N', E' \rangle$. Note that $N \cap \mathcal{C}_{m+3}(N') = \emptyset$ since $\mathcal{G}$ has exactly m vertices and none of the vertices in N can be a part of a simple cycle of size $m+3$; the label structures are not interconnected and each of these structures is connected to $\mathcal{G}$ by exactly one edge. Note also that there are exactly $2k$ vertices of distance 1 from $\mathcal{C}_{m+3}(N')$. For label $\mathcal{L}_t^{i_t}$ these vertices are n_{m+4}^t and n_t. These vertices cannot be mapped to each other by any isomorphism since they are connected to the $m+3$ cycle by the edges $\{n_{m+4}^t, n_i^t\}, \{n_{m+3}^t, n_t\}$, respectively, and for the connection vertices we have $3 = \deg n_i^t \neq \deg n_{m+3}^t = 4$.

We continue by showing that ϕ is an homomorphism. Arbitrarily choose σ in $Aut(\mathcal{G}')$. We have $\sigma(N_t^{i_t}) \cap N = \emptyset$ for $t \in [k]$ which implies

$$\sigma\left(\bigcup_{t\in[k]} N_t^{i_t}\right) = \bigcup_{t\in[k]} N_t^{i_t},$$

since automorphisms preserve vertex degrees. This observation implies that $\sigma(N) = N$.

Now consider the function $\phi : Aut(\mathcal{G}') \to Aut(\mathcal{G})$ where $\phi(\sigma) = \sigma|_N$. Note that since σ is an automorphism of $\mathcal{G}'$ that admits $\sigma(N) = N \subseteq N'$ and $\mathcal{G}$ is a subgraph of $\mathcal{G}'$, the restriction $\sigma|_N$ is an automorphism of $\mathcal{G}$. Moreover, composition is preserved by ϕ so it is a group homomorphism. Next, we show that ϕ is an injection. We use Theorem 2 for this: the homomorphism ϕ is injective if and only if its kernel is trivial, that is, $ker(\phi) = \{e'\}$ where e' is the identity element of $Aut(\mathcal{G}')$. We denote the identity element of $Aut(\mathcal{G})$ by e.

Thus, let $\sigma|_N = e$. Then $\sigma(N_t^{i_t}) = N_t^{i_t}$ for each $t \in [k]$. This follows directly from the fact that each label is connected to the original graph by exactly one edge. Note that

$$\deg n_{i_t}^t = \deg n_{m+1}^t = 3, \text{ and } \deg n_{m+4}^t = 1, \deg n_{m+3}^t = 4,$$

while all other vertices in $N_t^{i_t}$ have degree 2. This, in turn, implies that

$$\sigma(n_{m+4}^t) = n_{m+4}^t, \text{ and } \sigma(n_{m+3}^t) = n_{m+3}^t.$$

Since automorphisms preserve distances and cycles, and $n_{m+1}^t \in \mathcal{C}_3(N')$ and $n_{i_t}^t \notin \mathcal{C}_3(N')$, we have

$$\sigma(n_{m+1}^t) = n_{m+1}^t, \text{ and } \sigma(n_{i_t}^t) = n_{i_t}^t.$$

The rest of the vertices are fixed due to distance preservation, i.e., for all $j \in [m+4]$, it holds that $\sigma(n_j^t) = n_j^t$ for arbitrary $t \in [k]$. It follows that $\sigma = e'$. We conclude that $Aut(\mathcal{G}') \leq Aut(\mathcal{G})$.

Next, let n_t and $n_{t'}$ be two vertices in $\mathcal{G}$ attached to labels $\mathcal{L}_t^{i_t}$ and $\mathcal{L}_{t'}^{i_{t'}}$, respectively. Arbitrarily choose $\sigma \in Aut(\mathcal{G}')$ such that $\sigma(n_t) = n_{t'}$. Since each vertex is attached to at most one label, we see that $\sigma(N_t^{i_t}) = N_{t'}^{i_{t'}}$. By the same arguments used in the previous paragraph, there is one vertex per label that is both of degree 3 and does not belong to a cycle of size 3, these vertices are $n_{i_t}^t$ and $n_{i_{t'}}^{t'}$. Consequently, we have $\sigma(n_{i_t}^t) = n_{i_{t'}}^{t'}$ and $i_t = i_{t'}$. □

Assume now that $\mathcal{G}'$ is a fully colored version of $\mathcal{G}$, i.e. every vertex in $\mathcal{G}$ has been decorated with a label. Under this assumption, the next lemma can be viewed as a converse of Lemma 2: every automorphism of $\mathcal{G}$ that preserves the colors implied by the labels can be extended to an automorphism of $\mathcal{G}'$.

Lemma 3. *Let $\mathcal{G}' = \mathcal{G}_{n_1^{i_1},\dots,n_m^{i_m}}$ be a (fully) labeled graph with the labels $\mathcal{L}_1^{i_1},\dots,\mathcal{L}_m^{i_m}$ attached to all vertices $n_1,\dots,n_m$ of the graph $\mathcal{G}$. Arbitrarily choose σ in $Aut(\mathcal{G})$ such that $\sigma(n_t) = n_{t'}$ implies $i_t = i_{t'}$. Then, σ can be extended to $\sigma' \in Aut(\mathcal{G}')$ and the mapping $\cdot'$ is an injective homomorphism from $Aut(\mathcal{G})$ to $Aut(\mathcal{G}')$.*

Proof. Let $\mathcal{G} = \langle N, E\rangle$ be an undirectd graph and let $\mathcal{G}' = \langle N', E'\rangle$ be its fully labeled extension described above. For a color preserving automorphism $\sigma \in Aut(\mathcal{G})$, we define its extension σ' to the labeled graphs $\mathcal{G}'$ as follows:

1. for arbitrary $n_t \in N$, let $\sigma'(n_t) = \sigma(n_t)$, and
2. for all $t \in [m]$ and for all $j \in [m+4]$, let $\sigma'(n_j^t) = n_j^{t'}$ where $\sigma(n_t) = n_{t'}$.

We first note that σ' is a permutation of N' since σ is a permutation of N. This follows directly from point 2. Next, by point 1, we see that σ' preserves the edges in $\mathcal{G}$, i.e., if $\{n, n'\} \in E$, then $\{\sigma'(n), \sigma'(n')\} \in E$, too. Furthermore, we know that $\sigma(n_t) = n_{t'} \implies i_t = i_{t'}$ so σ' preserves the edges in the label structure $\mathcal{L}_{t'}^{i_{t'}}$, $t' \in [m]$. Finally, by point 1 and point 2, we have that $\sigma'(\{n_t, n_{m+3}^t\}) = \{n_{t'}, n_{m+3}^{t'}\}$. Hence, σ' is a permutation of N' that preserves all edges in E' so it is an automorphism.

We next show that the mapping $\cdot'$ is an injective homomorphism. We begin by verifying its injectivity. Arbitrarily choose distinct $\sigma, \psi \in Aut(\mathcal{G})$ and assume to the contrary that $\sigma' = \psi'$. Choose an element $n \in N$ such that $\sigma(n) \neq \psi(n)$. By the definition of $\cdot'$, $\sigma'(n) \neq \psi'(n)$, too, and this contradicts that $\sigma' = \psi'$.

We continue by proving that $\cdot'$ is a group homomorphism. Arbitrarily choose $\sigma, \psi \in Aut(\mathcal{G})$. We verify that $\sigma' \circ \psi' = (\sigma \circ \psi)'$. Arbitrarily choose a vertex $n' \in N'$. If $n' \in N$, then $\sigma'(n') = \sigma(n')$ and $\psi'(n') = \psi(n')$ so both $(\sigma' \circ \psi')(n')$ and $(\sigma \circ \psi)'(n)$ equals $(\sigma \circ \psi)(n')$. If $n' \notin N$, then $n' = n_j^t$ where $t \in [m]$ and $j \in [m+4]$. In this case, $\sigma'(n_j^t) = n_j^{t_1}$ where $\sigma(n_t) = n_{t_1}$ and $\psi'(n_j^t) = n_j^{t_1}$ where $\psi(n_t) = n_{t_1}$. Hence, it holds that

$$(\sigma' \circ \psi')(n_j^t) = \sigma'(\psi'(n_j^t)) = \sigma'(n_j^{t_1}) = n_j^{t_2},$$

where $\sigma(n_{t_1}) = n_{t_2}$. Let $\tau = \sigma \circ \psi$ and note that

$$(\sigma \circ \psi)'(n_j^t) = \tau'(n_j^t) = n_j^{t_3},$$

where $\tau(n_t) = n_{t_3}$. We know that $\sigma(n_{t_1}) = n_{t_2}$ and $\psi(n_t) = n_{t_1}$ so $n_{t_2} = \sigma(\psi(n_t)) = \tau(n_t) = n_{t_3}$. We conclude that $n_j^{t_2} = n_j^{t_3}$ and $(\sigma' \circ \psi')(n_j^t) = (\sigma \circ \psi)'(n_j^t)$. It follows that $(\sigma' \circ \psi')(n') = (\sigma \circ \psi)'(n')$ for arbitrary $\sigma, \psi \in Aut(\mathcal{G})$ and $n' \in N'$ so $\cdot'$ is a homomorphism. □

We now have the tools needed for proving Lemma 1.

Proof. (of Lemma 1) We demonstrate this by showing that there is an inclusion from $Aut(\mathcal{G}')$ to $Aut(\mathcal{G}_c)$ and backwards. This implies that $Aut(\mathcal{G}')$ and $Aut(\mathcal{G}_c)$ are isomorphic.

By Lemma 2(1) we know that the function $\psi : Aut(\mathcal{G}') \to Aut(\mathcal{G}_c)$ defined by $\psi(\sigma) = \sigma|_N$ is an injective homomorphism. By Lemma 2(2), we know that ψ is color preserving over $\mathcal{G}$. Thus, ψ is an injective homomorphism from $Aut(\mathcal{G}')$ to $Aut(\mathcal{G}_c)$ and we conclude that $Aut(\mathcal{G}') \leq Aut(\mathcal{G}_c)$ by Theorem 1.

We show the other direction. Arbitrarily choose $\sigma \in Aut(\mathcal{G}_c)$. Since σ must preserve colors, we have that $\sigma(n_t) = n_{t'}$ implies $\mathsf{col}(n_t) = \mathsf{col}(n_{t'})$. By Lemma 3, there is an extension of σ to $\sigma' \in Aut(\mathcal{G}')$ and the mapping $\cdot'$ is an injective homomorphism. Hence, $Aut(\mathcal{G}_c) \leq Aut(\mathcal{G}')$ by Theorem 1. □

Lemma 1 provides a simple way of proving that CAGEN and GI are polynomial-time equivalent problems. We first note that the function $\psi : Aut(\mathcal{G}') \to Aut(\mathcal{G}_c)$ defined by $\psi(\sigma) = \sigma|_N$ is surjective and, consequently, an isomorphism between $Aut(\mathcal{G}')$ and $Aut(\mathcal{G}_c)$. Assume there is an automorphism α in $Aut(\mathcal{G}_c)$ such that there is no $\beta \in Aut(\mathcal{G}')$ with $\psi(\beta) = \alpha$. Every automorphism in $Aut(\mathcal{G}_c)$ is color-preserving so $\alpha(n_t) = n_{t'} \implies \mathsf{col}(n_t) = \mathsf{col}(n_{t'})$. This implies that $\alpha' \in Aut(\mathcal{G}')$ where $\cdot'$ is the operation used in Lemma 3. However, $\psi(\alpha') = \alpha$ by definition and this leads to a contradiction.

Lemma 4. CAGEN $=_p$ GI.

Proof. CAGEN $\leq_p$ AGEN : Let $\mathcal{G}_c = \langle \mathcal{G}, \mathsf{col} \rangle$ be a colored graph, where $\mathcal{G} = \langle N, E \rangle$ is an undirected graph with $|N| = m$, and $\mathsf{col} : N \to [m]$ is a coloring function. The uncolored graph $\mathcal{G}' = \mathcal{G}_{n_1^{\mathsf{col}(n_1)},\ldots,n_m^{\mathsf{col}(n_m)}}$ can be computed in polynomial time and it contains at most $|N|^2 + 5|N|$ nodes (as was observed immediately after Definition 5). We know that $Aut(\mathcal{G}_c) = Aut(\mathcal{G}')$ by Lemma 1 so the CAGEN problem for $\mathcal{G}_c$ can be solved by solving AGEN for $\mathcal{G}'$. The set of generators $\mathcal{S} = \{\alpha_1, \ldots, \alpha_k\}$ that is returned contains at most $|N|^2 + 5|N|$ elements. The tighter bound of $|N|$ generators that are used in the definition of the CAGEN problem can be obtained as follows. By Lemma 2, we have that for every $\sigma \in Aut(\mathcal{G}')$ it holds that for every vertex $n \in N$, $\sigma(n) = m$ implies that $m \in N$ and that the label $\mathcal{L}_n$ attached to n is uniquely mapped by σ to the label $\mathcal{L}_m$ attached to m. This implies that it is enough to check only the vertices of the graph $\mathcal{G}$, which, in turn, implies that $k \leq |N|$. We know that the function $\psi(\sigma) = \sigma|_N$ is an isomorphism between $Aut(\mathcal{G}_c)$ and $Aut(\mathcal{G}')$. Thus, the set $S' = \{\psi(\alpha_1), \ldots, \psi(\alpha_k)\}$ is a set of generators for $\mathcal{G}_c$, it contains at most $|N|$ elements, and it can be computed in polynomial time. Hence, there is a polynomial-time reduction from CAGEN to AGEN.

AGEN $\leq_p$ GI : This follows directly from the fact that AGEN $=_p$ GI (Theorem 4).

GI $\leq_p$ CAGEN : Let $\mathcal{G}_1 = \langle N_1, E_1 \rangle$ and $\mathcal{G}_2 = \langle N_2, E_2 \rangle$ be two undirected graphs. Assume without loss of generality that $N_1 \cap N_2 = \emptyset$. Define a colored graph $\mathcal{G}_c = \langle \mathcal{G}_1 \cup \mathcal{G}_2, \mathsf{col} \rangle$ where col assigns the same color to all vertices. Compute CAGEN for the graph $\mathcal{G}_c$. This results in a set of generators for $Aut(\mathcal{G}_c)$ whose size is polynomially bounded in the size of $\mathcal{G}_c$. For each generator σ, check if there exists $n \in N_1$ such that $\sigma(n) \in N_2$. If this is the case, then accept and otherwise reject. □

The main result of this section is an immediate consequence of the previous lemma.

Theorem 5. StructSym *is polynomial-time Turing reducible to* GI.

Proof. The generators of $Aut(\Pi)$ are per definition the generators of $Aut(PDG_\Pi)$ as was pointed out at the end of Section 2.3. We know that PDG_Π is a colored undirected graph so the problem of computing the generators of $Aut(PDG_\Pi)$ is in **GI** by Lemma 4. □

4.2 Hardness of Computing Structural Symmetries

The goal of this section is to prove **GI**-hardness of StructSym. We will see (in Theorem 6) that this result holds even for severely restricted planning tasks. Thus, we will tacitly assume that every action has cost 1 and that every variable has domain $\mathcal{D}_2 = \{0, 1\}$ throughout this section.

The basic strategy for the hardness proof is to transform (in polynomial time) directed graphs $\mathcal{G}$ into planning tasks Π such that $Aut(\mathcal{G})$ and $Aut(\Pi)$ are isomorphic. Such a transformation allow us to transform AGEN into StructSym and thus prove **GI**-hardness; the fact that AGEN is only defined for undirected graphs can be overcome by a method suggested by Zemlyachenko et al. (1985). The transformation of directed graphs into planning tasks is presented in Lemma 6. The transformation itself is straightforward but proving that it has the required properties needs some work. To simplify the analysis of this transformation, we exploit the causal graph of Π and divide the correctness proof into two steps: we first prove that $Aut(CG_\Pi) = Aut(\mathcal{G})$ and then that $Aut(CG_\Pi) = Aut(\Pi)$. The first isomorphism holds almost trivially given the construction of Π but the second isomorphism requires some effort. To show the second isomorphism, we introduce a homomorphism Φ from the structural symmetries of a planning task to the automorphism group of its causal graph. This homomorphism is not bijective in general but we prove that this is indeed the case when considering planning tasks that are the result of our transformation. This implies that $Aut(CG_\Pi) = Aut(\Pi)$ since bijective homomorphisms (i.e. isomorphisms) preserve automorphims groups (as was discussed in Section 2.2).

We begin by introducing the mapping Φ and we prove in Lemma 5 that it is a homomorphism from $Aut(\Pi)$ to $Aut(CG_\Pi)$ for every planning task Π. In order to describe Φ, arbitrarily choose a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$ and pick $\sigma \in Aut(\Pi)$. The function σ is a permutation on $\mathcal{F} \cup \mathcal{O}$ that maps facts to facts and actions to actions. We connect σ to the set of variables $\mathcal{V}$ via the following observation: each variable v can be represented by the set of facts $\mathcal{F}_v$ (as defined in Section 2.1). With this viewpoint, we let $\sigma(v) = \{\sigma(\langle v, d \rangle) \mid d \in \mathcal{D}(v)\}$ for $v \in \mathcal{V}$. It follows that $\sigma(\mathcal{F}_v) = \mathcal{F}_u$ for some $u \in \mathcal{V}$ and it makes sense to impose a restriction on σ such that $\bar{\sigma}(v) = u$. The map $\bar{\sigma}$ is formally defined in the proof of Lemma 5 (see below). Under this convention, we can view $\bar{\sigma}$ as a permutation on $\mathcal{V}$, since by Definition 3 we have that $\sigma(\mathcal{F}_\mathcal{V}) = \mathcal{F}_\mathcal{V}$. For each variable $v \in \mathcal{V}$, we let n_v denote the corresponding vertex in CG_Π and we define the mapping $l(\mathcal{F}_v) = n_v$. Finally, we define the mapping Φ from $Aut(\Pi)$ to $Aut(CG_\Pi)$ as follows: $\Phi(\sigma)(n_v) = l(\sigma(\mathcal{F}_v))$ for every vertex n_v in CG_Π.

Lemma 5. *The mapping* Φ *is a homomorphism from* $Aut(\Pi)$ *to* $Aut(CG_\Pi)$ *for every planning task* Π.

Proof. Arbitrarily choose a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$ and assume $CG_\Pi = (N, E)$ where $N = \{n_v \mid v \in \mathcal{V}\}$. Let $\Omega(\mathcal{V})$ be the group of all possible permutations over the

variables of the planning task Π. We prove the claim by chaining two homomorphisms, $Aut(\Pi) \to \Gamma \to Aut(CG_\Pi)$, where Γ is a subgroup of $\Omega(\mathcal{V})$.

First, let us note that a structural symmetry σ maps a set of facts associated with one variable to a set of facts that are associated with another (not necessarily different) variable, i.e., $\sigma(\mathcal{F}_\mathcal{V}) = \mathcal{F}_\mathcal{V}$. This definition, unfortunately, makes the reasoning about variable symmetries cumbersome. To avoid unnecessary complications in notation we introduce a bijective function $\mathsf{vl} : \mathcal{F}_\mathcal{V} \to \mathcal{V}$ such that $\mathsf{vl}(\mathcal{F}_v) = v$ that maps a set of variable facts to their *variable label*. For $\sigma \in Aut(\Pi)$, define $\bar{\sigma} \in \Omega(\mathcal{V})$ as $\bar{\sigma}(v) = \mathsf{vl} \circ \sigma \circ \mathsf{vl}^{-1}(v)$. Say $\sigma(\mathcal{F}_v) = \mathcal{F}_u$, then we have

1. $\bar{\sigma}(v) = \mathsf{vl} \circ \sigma \circ \mathsf{vl}^{-1}(v) = \mathsf{vl} \circ \sigma(\mathcal{F}_v) = \mathsf{vl}(\mathcal{F}_u) = u$, and
2. $\bar{\sigma}_1 \circ \bar{\sigma}_2 = \mathsf{vl} \circ \sigma_1 \circ \mathsf{vl}^{-1} \circ \mathsf{vl} \circ \sigma_2 \circ \mathsf{vl}^{-1} = \mathsf{vl} \circ \sigma_1 \circ \sigma_2 \circ \mathsf{vl}^{-1} = \overline{\sigma_1 \circ \sigma_2} \in \Omega(\mathcal{V})$.

I.e., the map is well-defined and closed under composition, which implies $\bar{\cdot} : Aut(\Pi) \to \Gamma \leq \Omega(\mathcal{V})$ is a group homomorphism (see Section 2.2). We have that Γ is indeed a group, since it is a homomorphic image of a group. Thus, we defined $\bar{\sigma}$ that allows us to reason directly about the indices of the causal graph CG_Π while still preserving the automorphism structure imposed by structural symmetries. Hence, for a causal graph vertex n_v we can write $\Phi(\sigma)(n_v) = l(\sigma(\mathcal{F}_v)) = n_{\bar{\sigma}(v)}$.

Next, let us show that if $\sigma \in Aut(\Pi)$, then $\Phi(\sigma)$ is in $Aut(CG_\Pi)$; in other words, for every $\sigma \in Aut(\Pi)$, $(n_u, n_v) \in E \Leftrightarrow (\Phi(\sigma)(n_v), \Phi(\sigma)(n_u)) \in E$. We first show that if $(n_u, n_v) \in E$, then $(\Phi(\sigma)(n_u), \Phi(\sigma)(n_v)) \in E$, too. For each edge $(n_u, n_v) \in E$, there exists an $o \in \mathcal{O}$ such that $u \in vars(\mathsf{pre}(o)) \cup vars(\mathsf{eff}(o))$ and $v \in vars(\mathsf{eff}(o))$. Therefore, for each $\sigma \in Aut(\Pi)$ it holds that $\bar{\sigma}(u) \in vars(\mathsf{pre}(\sigma(o))) \cup vars(\mathsf{eff}(\sigma(o)))$ and $\bar{\sigma}(v) \in vars(\mathsf{eff}(\sigma(o)))$ so $(n_{\bar{\sigma}(u)}, n_{\bar{\sigma}(v)}) \in E$. We conclude that

$$(n_u, n_v) \in E \Rightarrow (n_{\bar{\sigma}(u)}, n_{\bar{\sigma}(v)}) \in E \Rightarrow (l(\sigma(\mathcal{F}_u)), l(\sigma(\mathcal{F}_v))) \in E \Rightarrow (\Phi(\sigma)(n_u), \Phi(\sigma)(n_v)) \in E.$$

Assume instead that $(\Phi(\sigma)(n_u), \Phi(\sigma)(n_v)) \in E$. We see that

$$(\Phi(\sigma)(n_u), \Phi(\sigma)(n_v)) \in E \Rightarrow (n_{\bar{\sigma}(u)}, n_{\bar{\sigma}(v)}) \in E \Rightarrow (n_u, n_v) \in E.$$

Next, we verify that $\Phi(\sigma_1) \circ \Phi(\sigma_2)(n_v) = (\Phi(\sigma_1 \circ \sigma_2))(n_v)$ for arbitrary $\sigma_1, \sigma_2 \in Aut(\Pi)$ and $v \in \mathcal{V}$:

$$(\Phi(\sigma_1 \circ \sigma_2))(n_v) = n_{\overline{\sigma_1 \circ \sigma_2}(v)} = n_{\bar{\sigma}_1 \circ \bar{\sigma}_2(v)} = \Phi(\sigma_1)(n_{\bar{\sigma}_2(v)}) = \Phi(\sigma_1) \circ \Phi(\sigma_2)(n_v).$$

□

We continue by proving that for every directed graph $\mathcal{G}$, there exists a planning task Π such that $Aut(\mathcal{G}) = Aut(\Pi)$. The homomorphism Φ is the key for proving that these two automorphism groups are isomorphic.

Lemma 6. *Let $\mathcal{G}$ be a directed graph. Then, there exists a planning task Π such that $Aut(\mathcal{G}) = Aut(\Pi)$ and Π can be computed in polynomial time.*

Proof. Given a directed graph $\mathcal{G} = \langle N, E \rangle$, we construct a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$ that satisfies the conditions of the lemma. Let $\mathcal{V} = \{v \mid v \in N\}$ with $\mathcal{D}(v) = \mathcal{D}_2$ for each $v \in N$, i.e., $\mathcal{F} = \mathcal{V} \times \mathcal{D}_2$. For arbitrary $u, v \in \mathcal{V}$, we define the actions $o_{v:0 \to v:1} = \langle \{\langle v, 0 \rangle\}, \{\langle v, 1 \rangle\} \rangle$ and $o_{u:0 \to v:0} = \langle \{\langle u, 0 \rangle\}, \{\langle v, 0 \rangle\} \rangle$. All actions are of cost one, thus the comparison of costs is omitted. We will refer to the first type as *inner actions* and the second type as *outer actions*. Finally, let

- $\mathcal{O} = \{o_{v:0 \to v:1} \mid v \in \mathcal{V}\} \cup \{o_{v:0 \to u:0} \mid (v, u) \in E\}$,
- $s_0 = \{\langle v, 0 \rangle \mid v \in \mathcal{V}\}$, and
- $G = \{\langle v, 1 \rangle \mid v \in \mathcal{V}\}$.

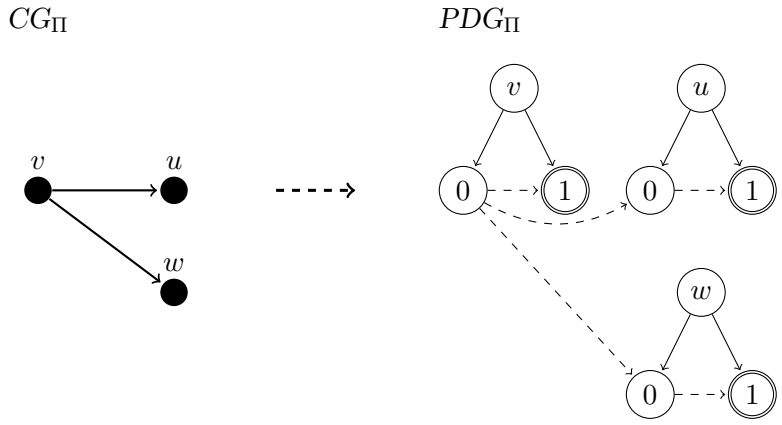


Figure 5: Toy example of a given graph CG_Π, and the PDG_Π imposed by the constructed task Π, where $\mathcal{V} = \{v, u, w\}$, and the dashed arcs represent the actions (we omit the pre and eff vertices of each action, since they are obvious from the context). Recall that by Theorem 3 we have that $Aut(\Pi) = Aut(PDG_\Pi)$.

It follows that Π can be computed in polynomial time and that CG_Π and $\mathcal{G}$ are isomorphic graphs due to the actions in $\mathcal{O}$: specifically, the outer actions. Thus, $Aut(CG_\Pi) = Aut(\mathcal{G})$ and it is sufficient to prove that $Aut(CG_\Pi) = Aut(\Pi)$.

Let Φ denote the mapping that we introduced just before Lemma 5. The lemma implies that Φ is a homomorphism from $Aut(\Pi)$ to $Aut(CG_\Pi)$. We prove below that the mapping Φ is bijective; this implies that $Aut(CG_\Pi) = Aut(\Pi)$ and, consequently, that $Aut(\mathcal{G}) = Aut(\Pi)$. Let $\langle N', E' \rangle = CG_\Pi$ with $N' = \{n_v \mid v \in \mathcal{V}\}$. Furthermore, let $\mathcal{O}_i$ be the set of inner actions in $\mathcal{O}$ and $\mathcal{O}_o$ be the set of outer actions.

Φ *is surjective.* Arbitrarily choose $\psi \in Aut(CG_\Pi)$. We prove that there is a σ in $Aut(\Pi)$ such that $\Phi(\sigma) = \psi$. Recall that the set of facts $\mathcal{F}$ is $\{\langle v, 0 \rangle, \langle v, 1 \rangle \mid v \in \mathcal{V}\}$. Define $\sigma : \mathcal{F} \cup \mathcal{O} \to \mathcal{F} \cup \mathcal{O}$ such that $\sigma(\langle v, d \rangle) = \langle \psi(v), d \rangle$ for $\langle v, d \rangle \in \mathcal{F}$ and extend it to $\mathcal{O}$ in the natural way. We verify that σ is a structural symmetry, i.e. it is a permutation on $\mathcal{F}$, a permutation on $\mathcal{O}$, and it satisfies conditions 1-3 of Definition 3. We first remind the reader

that there is a one-to-one correspondence between the vertices in N' and the variables in $\mathcal{V}$. We will abuse notation and view ψ as a function on $\mathcal{V}$ whenever convenient. With this convention, it follows immediately that σ is a permutation on $\mathcal{F}$ since ψ is a permutation on $\mathcal{V}$. We continue with the other conditions.

1. $\sigma(\mathcal{F}_v) = \sigma(\{\langle v, 0\rangle, \langle v, 1\rangle\}) = \{\langle \psi(v), 0\rangle, \langle \psi(v), 1\rangle\} = \mathcal{F}_u$ for some $u \in \mathcal{V}$. Since ψ is a permutation on $\mathcal{V}$, it follows that $\sigma(\mathcal{F}_\mathcal{V}) = \mathcal{F}_\mathcal{V}$.

2. We consider the two types of actions.

   (a) Inner actions. Arbitrarily choose an inner action $a = \langle\{\langle v, 0\rangle\}, \{\langle v, 1\rangle\}\rangle$ and note that $\sigma(a) = \langle\{\langle \psi(v), 0\rangle\}, \{\langle \psi(v), 1\rangle\}\rangle$. Hence, $\sigma(a)$ is an inner action, too. Consequently, $\sigma(\mathcal{O}_i) = \mathcal{O}_i$ since ψ is a permutation on $\mathcal{V}$.

   (b) Outer actions. Observe that each edge $(u, v) \in E'$ has a one-to-one correspondence with an outer action $a = \langle\{\langle u, 0\rangle\}, \{\langle v, 0\rangle\}\rangle$. Arbitrarily choose an outer action $a = \langle\{\langle u, 0\rangle\}, \{\langle v, 0\rangle\}\rangle$. We see that $\langle\{\langle u, 0\rangle\}, \{\langle v, 0\rangle\}\rangle \in \mathcal{O} \iff (u, v) \in E' \iff (\psi(u), \psi(v)) \in E' \iff \langle\{\langle \psi(u), 0\rangle\}, \{\langle \psi(v), 0\rangle\}\rangle \in \mathcal{O}$. Note that $\langle\{\langle \psi(u), 0\rangle\}, \{\langle \psi(v), 0\rangle\}\rangle = \sigma(a)$ so $\sigma(a)$ is an outer action. We conclude that $\sigma(\mathcal{O}_o) = \mathcal{O}_o$ since ψ is a permutation on $\mathcal{V}$.

3. $\sigma(\langle v, 1\rangle) = \langle \psi(v), 1\rangle \in G$ for all $v \in \mathcal{V}$. Since ψ is a permutation on $\mathcal{V}$, it follows that $\sigma(G) = G$.

Finally, 2(a) and 2(b) directly implies that σ is a permutation on $\mathcal{O}$. We have thus verified that $\Phi : Aut(\Pi) \to Aut(CG_\Pi)$ is a surjective homomorphism.

Φ is injective. We use Theorem 2 for proving injectivity; it is thus sufficient to prove that $ker(\Phi) = \{id_{CG}\}$ where id_{CG} is the identity element in $Aut(CG_\Pi)$.

Arbitrarily choose $\sigma \in Aut(\Pi)$ such that $\Phi(\sigma) = id_{CG}$. This means that $\Phi(\sigma)(n_v) = n_v$ for all $v \in \mathcal{V}$. By the construction of Π, we know that $\langle v, 0\rangle$ is a fact in the initial state and $\langle v, 1\rangle$ is a fact in the goal state for arbitrary $v \in \mathcal{V}$. Hence, $\sigma(\langle v, 1\rangle) = \langle v, 1\rangle$ and $\sigma(\langle v, 0\rangle) = \langle v, 0\rangle$ for each $v \in \mathcal{V}$ and, consequently, $\sigma(f) = f$ for every fact $f \in \mathcal{F}$. The function σ is a structural symmetry so it satisfies $\sigma(\mathsf{pre}(o)) = \mathsf{pre}(\sigma(o))$ and $\sigma(\mathsf{eff}(o)) = \mathsf{eff}(\sigma(o))$ for every $o \in \mathcal{O}$ by Definition 3. Arbitrarily choose an action $o = \langle p, e\rangle$ in $\mathcal{O}$, where $p \subseteq \mathcal{F}$ and $e \subseteq \mathcal{F}$ are precondition and effect, respectively. We see that

$$\sigma(o) = \sigma(\langle p, e\rangle) = \langle \sigma(p), \sigma(e)\rangle = \langle p, e\rangle = o.$$

Thus, we have proved that $\sigma(x) = x$ for all $x \in \mathcal{F} \cup \mathcal{O}$ so σ is the identity function and the kernel of Φ is trivial.

We conclude that Φ is an isomorphism between $Aut(\Pi)$ and $Aut(CG_\Pi)$ and that $Aut(\Pi) = Aut(CG_\Pi) = Aut(\mathcal{G})$. □

We can finally prove the main result of this section by combining the **GI**-hardness of AGEN (Theorem 4) with Lemma 6 and a result due to Zemlyachenko et al. (1985).

Theorem 6. StructSym *is* ***GI****-hard even if restricted to instances with the following properties:*

1. *variable domains are* $\{0, 1\}$*,*
2. *every action has cost 1,*
3. *actions have at most one precondition* $\langle v, 0\rangle$*, and*
4. *actions have one effect.*

Proof. Let StructSym$_R$ denote StructSym restricted to instances with the properties listed above. Let $\mathcal{G}$ be an arbitrary undirected graph. It is possible to construct a directed graph $\mathcal{G}'$ such that $Aut(\mathcal{G}) = Aut(\mathcal{G}')$ in polynomial time (Zemlyachenko et al., 1985). There is a planning task Π such that $Aut(\mathcal{G}') = Aut(\Pi)$ and Π can be computed in polynomial time by Lemma 6. It can easily be verified that the planning task Π satisfies restrictions 1–4 by inspecting the proof of Lemma 6. Thus, there is a polynomial-time reduction from AGEN to StructSym$_R$. We know that AGEN is **GI**-complete by Theorem 4 and this concludes the proof. □

By combining Theorems 5 and 6, we conclude that the StructSym problem is **GI**-complete. This implies that this problem can be solved in quasi-polynomial time but it is unlikely that it can be solved in polynomial time. It may be interesting to note that Theorem 6 implies that the StructSym problem can be **GI**-hard even for classes of planning tasks that are polynomial-time solvable. Tasks that satisfy restrictions 1–4 have properties U, B, and S (Bäckström & Nebel, 1995) and the plan existence problem can consequently be solved in polynomial time.

5. Polynomial-Time Computation of Symmetries

The goal of this section is to exhibit a fragment of FDR planning where the computation of structural symmetries can be performed in polynomial time. The idea is to exploit connections between the planning task and its causal graph in such a way that the corresponding PDG has bounded degree. This allows us to use an algorithm by Luks (1982) to solve StructSym in polynomial time. Let $\mathcal{G} = \langle N, E\rangle$ be a directed graph and let $n \in N$. We let $\deg_{\mathsf{in}}(n, \mathcal{G})$ denote the *in-degree* of n, i.e. $|\{(n', n) \in E \mid n' \in N\}|$, and we define the *out-degree* of n ($\deg_{\mathsf{out}}(n, \mathcal{G})$) analogously. The *degree* of n is $\deg(n, \mathcal{G}) = \deg_{\mathsf{in}}(n, \mathcal{G}) + \deg_{\mathsf{out}}(n, \mathcal{G})$ and the degree of the graph $\mathcal{G}$ is $\deg(\mathcal{G}) = \max\{\deg_{\mathsf{in}}(n) + \deg_{\mathsf{out}}(n) \mid n \in N\}$.

Definition 6. *Arbitrarily choose* $B, C, D \in \mathbb{N}$*. Define* $X(B, C, D)$ *to be the set of planning tasks* $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G\rangle$ *that satisfies the following three conditions:*

1. $\deg(CG_\Pi) \leq B$*,*
2. $C_\mathcal{O} = |\{\mathsf{cost}(o) \mid o \in \mathcal{O}\}| \leq C$*, and*
3. $|\mathcal{D}(v)| \leq D$ *for all* $v \in \mathcal{V}$*.*

The plan existence problem for instances in $X(B, C, D)$ becomes **NP**-hard even for small values of B, C, and D: for instance, choosing $B = 2$, $C = 1$, and $D = 5$ is sufficient (Giménez & Jonsson, 2009).

In what follows we make some assumptions on planning tasks Π: we assume that there are no two identical actions with different names and we assume that every action in Π has at least one effect. Note that actions violating these conditions can be removed from the planning task without loss of generality and this can be done by straightforward polynomial-time preprocessing.

We will now prove that every PDG_Π has bounded degree when $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G\rangle \in X(B, C, D)$. Recall from Definition 4 that the vertices of PDG_Π can be divided into four sets

$$N = N_\mathcal{V} \cup \bigcup_{v\in\mathcal{V}} N_{\mathcal{D}(v)} \cup N_\mathcal{O}^{\mathsf{pre}} \cup N_\mathcal{O}^{\mathsf{eff}}.$$

The proof is easy for the subsets $N_\mathcal{V}$, $N_\mathcal{O}^{\mathsf{pre}}$, and $N_\mathcal{O}^{\mathsf{eff}}$ while more effort is needed for the subset $\bigcup_{v\in\mathcal{V}} N_{\mathcal{D}(v)}$. Thus, we divide the proof into two parts where we first analyse the vertices in $\bigcup_{v\in\mathcal{V}} N_{\mathcal{D}(v)}$ (Lemma 7) and consider the three other sets in Lemma 8.

Lemma 7. *For arbitrary $B, C, D \in \mathbb{N}$,*

$$\deg(n, PDG_\Pi) \leq D + C \cdot (D+1)^{B+1} + C \cdot (B+1) \cdot (D+1)^{2B+1}.$$

when $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G\rangle \in X(B, C, D)$ and $n \in \bigcup_{v\in\mathcal{V}} N_{\mathcal{D}(v)}$.

Proof. Arbitrarily choose $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G\rangle \in X(B, C, D)$ and assume $CG_\Pi = \langle M, E\rangle$ with $M = \{m_v \mid v \in \mathcal{V}\}$. Let B_{in} and B_{out} denote the maximum in-degree and out-degree of CG_Π, respectively. Let $f = \langle v, d'\rangle$ be a fact and let $n_f \in N_{\mathcal{D}(v)}$ be the corresponding fact vertex. There are three types of vertices that are attached to n_f: variable vertices, action effect vertices, and action precondition vertices. We divide the proof into three parts where the different types of vertices are analyzed.

Variable vertices. There are at most D variable vertices connected to n_f by the construction of PDGs.

Action effect vertices. We compute the maximal number of same-cost actions that can have f as their effect. Recall that $\deg_{\mathsf{in}}(m_v, CG_\Pi) \leq B_{\mathsf{in}}$ and

$$\deg_{\mathsf{in}}(m_v, CG_\Pi) = |\{u \in \mathcal{V} \mid \exists o \in \mathcal{O} : u \in \mathit{vars}(\mathsf{pre}(o)) \cup \mathit{vars}(\mathsf{eff}(o)) \wedge v \in (\mathit{vars}(\mathsf{eff}(o))\} \setminus \{v\})|.$$

Let $o' \in \mathcal{O}$ be an action such that $f = \langle v, d\rangle \in \mathsf{eff}(o')$. Thus,

$$\mathit{vars}(\mathsf{pre}(o')) \cup \mathit{vars}(\mathsf{eff}(o')) \subseteq \{u \in \mathcal{V} \mid \exists o \in \mathcal{O} : u \in \mathit{vars}(\mathsf{pre}(o)) \cup \mathit{vars}(\mathsf{eff}(o)) \wedge v \in \mathit{vars}(\mathsf{eff}(o))\}.$$

Thus, we have $|\mathit{vars}(\mathsf{pre}(o'))| \leq B_{\mathsf{in}} + 1$ and $|\mathit{vars}(\mathsf{eff}(o'))| \leq B_{\mathsf{in}} + 1$. This means that there are at most $B_{\mathsf{in}} + 1$ variables that can be in the precondition of o' (including v itself), and there are B_{in} variables that can be in the effect of the action (v is set to d). The domain of each variable has a size of at most D, plus 1, since the variable can be also undefined for

this action. We conclude that there are at most $(D+1)^{2B_{\mathsf{in}}+1}$ same-cost actions that have the fact f in their effects, and at most $C \cdot (D+1)^{2B_{\mathsf{in}}+1} \leq C \cdot (D+1)^{2B+1}$ actions in total.

Action precondition vertices. We compute the maximal number of same-cost actions that have f as their precondition. We aim to prove that the upper bound on the number of these actions is $(D+1)^{B+1} + B \cdot (D+1)^{2B+1}$. We achieve this bound by using the combinatorial approach sometimes referred to as *combinations counting*. In other words, we would like to give an upper bound on the size of the set $A^f = \{o \in \mathcal{O} \mid f \in \mathsf{pre}(o)\}$ where we divide A^f into three subsets:

1. $A^f_0 = \{o \in \mathcal{O} \mid \mathsf{eff}(o) = \emptyset\} \cap A^f$,
2. $A^f_1 = \{o \in \mathcal{O} \mid \mathit{vars}(\mathsf{eff}(o)) = \{v\}\} \cap A^f$, and
3. $A^f_2 = \{o \in \mathcal{O} \mid \mathit{vars}(\mathsf{eff}(o)) \setminus \{v\} \neq \emptyset\} \cap A^f$.

It holds that $A^f = A^f_0 \cup A^f_1 \cup A^f_2$ so $|A^f| \leq |A^f_0| + |A^f_1| + |A^f_2|$.

By our initial assumptions, it holds that $|A^f_0| = 0$ since these actions have no effect on the state variables.

Now, let $o \in A^f_1$. As we have seen before, for each $o \in \mathcal{O}$ it holds that $|\mathit{vars}(\mathsf{pre}(o))| \leq B_{\mathsf{in}} + 1$. Every action in A^f affects the variable v so

$$\left| \bigcup_{o \in A^f} \mathit{vars}(\mathsf{pre}(o)) \right| \leq B_{\mathsf{in}} + 1,$$

since, otherwise, it contradicts the fact that $\deg_{\mathsf{in}}(m_v) \leq B_{\mathsf{in}}$. Let us now count these actions. For each action in $o \in A^f_1$, one of the preconditions is already fixed to be f, i.e., $f \in \mathsf{pre}(o)$. Thus we have $\mathsf{pre}(o) \setminus \{f\} \leq B_{\mathsf{in}}$ and there are at most B_{in} other facts in $\mathsf{pre}(o)$, each corresponding to a different variable. To determine each of these facts we need to choose B_{in} variable/value pairs. There are at most $D+1$ values for each variable (including "undefined"). Thus, we have a multiplicative factor of $(D+1)^{B_{\mathsf{in}}}$. We continue by determining the effect of o. Note that $\mathit{vars}(\mathsf{eff}(o)) = \{v\}$ so v cannot take the undefined value. It also holds that $f \in \mathsf{pre}(o)$ thus $f \notin \mathsf{eff}(o)$. This means that v can take at most $D-1$ different values. Multiplying these factors results in

$$|A^f_1| \leq (D-1)(D+1)^{B_{\mathsf{in}}} \leq (D+1)^{B_{\mathsf{in}}+1}.$$

Lastly, we estimate the size of A^f_2. Arbitrarily choose $o \in A^f_2$. We claim that

$$|A^f_2| \leq B_{\mathsf{out}} \cdot D \cdot (D+1)^{2B_{\mathsf{in}}},$$

and justify the multiplicative factors as follows.

1. B_{out} – Let $u \in \mathit{vars}(\mathsf{eff}(o))$ such that $u \neq v$. Choosing u is equivalent to choosing an outgoing edge of $m_v \in M$, for example, $(m_v, m_u) \in E$. Note that there can be at most B_{out} such edges.
2. D – Pick a value d' of u, i.e., $f' = \langle u, d' \rangle \in \mathsf{eff}(o)$. By our initial assumptions, $|\mathcal{D}(u)| \leq D$.

3. $(D+1)^{2B_{\mathsf{in}}}$ – Since we need to estimate the number of effect edges of $n_{f'}$, we repeat the argument that we used for analysing the number of effect edges of n_f. Recall that we have established that $f \in \mathsf{pre}(o)$ and $f' \in \mathsf{eff}(o)$. Each action of the form of o has at most B_{in} precondition variables since, otherwise, it would contradict the fact that $\deg_{\mathsf{in}}(m_u) \leq B_{\mathsf{in}}$. Note that the action o has B_{in} affected variables (u excluded) by the same argument. Each of these variables can take at most D values together with one "extra value" if the variable is undefined for this action.

Hence, we have

$$|A_2^f| \leq B_{\mathsf{out}} \cdot D \cdot (D+1)^{2B_{\mathsf{in}}} \leq B_{\mathsf{out}} \cdot (D+1)^{2B_{\mathsf{in}}+1}.$$

The three estimations above sum up to the following

$$|A^f| \leq (D+1)^{B_{\mathsf{in}}+1} + B_{\mathsf{out}} \cdot (D+1)^{2B_{\mathsf{in}}+1} \leq (D+1)^{B+1} + B \cdot (D+1)^{2B+1}.$$

Finally, there are at most C distinct action costs so

$$\deg(n_f, PDG_\Pi) \leq C \cdot (D+1)^{B+1} + C \cdot B \cdot (D+1)^{2B+1}.$$

Conclusion. The calculations above imply that

$$\begin{aligned} \deg(n_f, PDG_\Pi) &\leq D + C \cdot (D+1)^{2B+1} + C \cdot (D+1)^{B+1} + C \cdot B \cdot (D+1)^{2B+1} \\ &= D + C \cdot (D+1)^{B+1} + C \cdot (B+1) \cdot (D+1)^{2B+1}, \end{aligned}$$

whenever $n_f \in \bigcup_{v \in \mathcal{V}} N_{\mathcal{D}(v)}$. □

Lemma 8. *For arbitrary $B, C, D \in \mathbb{N}$, there exists a constant $K = K(B,C,D) \in \mathbb{N}$ such that $\deg(PDG_\Pi) \leq K$ when $\Pi \in X(B,C,D)$.*

Proof. Let $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle \in X(B,C,D)$ and assume $CG_\Pi = \langle M, E \rangle$ with $M = \{m_v \mid v \in \mathcal{V}\}$. Let B_{in} and B_{out} denote the maximum in-degree and out-degree of CG_Π, respectively. Definition 4 tells us that the vertex set N of PDG_Π is the union of four sets

$$N = N_{\mathcal{V}} \cup \bigcup_{v \in \mathcal{V}} N_{\mathcal{D}(v)} \cup N_{\mathcal{O}}^{\mathsf{pre}} \cup N_{\mathcal{O}}^{\mathsf{eff}}.$$

The set $\bigcup_{v \in \mathcal{V}} N_{\mathcal{D}(v)}$ was analyzed in Lemma 7 so it is sufficient to cover the other three sets.

Let $n_v \in N_{\mathcal{V}}$ be the vertex associated with the variable v. We see that

$$\deg(n_v, PDG_\Pi) = |\{(n_v, n_{\langle v,d \rangle}) \mid d \in \mathcal{D}(v)\}| = |\mathcal{D}(v)| \leq D.$$

We continue with the vertices in $N_{\mathcal{O}}^{\mathsf{pre}}$ and $N_{\mathcal{O}}^{\mathsf{eff}}$. Arbitrarily choose an action $o \in \mathcal{O}$. Let $n_o^{\mathsf{pre}} \in N_{\mathcal{O}}^{\mathsf{pre}}$ be the vertex associated with the preconditions of action o. We know that $|\mathsf{eff}(o)| \geq 1$. Arbitrarily choose $u \in \mathcal{V}$ such that variable u is affected by o. Assume $\deg(n_o^{\mathsf{pre}}, PDG_\Pi) > B_{\mathsf{in}} + 1$. This implies that $|\mathsf{pre}(o)| > B_{\mathsf{in}}$ and, in turn, that $\deg_{\mathsf{in}}(m_u, CG_\Pi) > B_{\mathsf{in}}$ and we have a contradiction. Thus, $\deg(n_o^{\mathsf{pre}}, PDG_\Pi) \leq B_{\mathsf{in}} + 1 \leq B + 1$.

Let $n_o^{\mathsf{eff}} \in N_{\mathcal{O}}^{\mathsf{eff}}$ be the vertex associated with the effects of action o and arbitrarily choose a variable $v \in \mathcal{V}$ that is affected by o. Assume $\deg(n_o^{\mathsf{pre}}, PDG_\Pi) > B_{\mathsf{in}} + 2$. Then, $|\mathsf{eff}(o)| > B_{\mathsf{in}} + 1$ which implies that $\deg_{\mathsf{in}}(m_v, CG_\Pi) > B_{\mathsf{in}}$ so we have once again reached a contradiction. Thus, $\deg(n_o^{\mathsf{pre}}, PDG_\Pi) \leq B_{\mathsf{in}} + 2 \leq B + 2$.

We conclude the proof by combining the bounds above with Lemma 7. Arbitrarily choose a vertex n with maximal degree in PDG_Π and note that $\deg(n, PDG_\Pi)$ is bounded by

$$D + (B_{\mathsf{in}} + 1) + (B_{\mathsf{in}} + 2) + (D + C \cdot (D+1)^{B+1} + C \cdot (B+1) \cdot (D+1)^{2B+1}).$$

We see that $K \geq \deg(PDG_\Pi) = \max_{n \in \mathcal{V}} \deg(n, PDG_\Pi)$ can be expressed as a function in the parameters B, C, and D.

□

We can now prove the main result of this section.

Theorem 7. *Arbitrarily choose $B, C, D \in \mathbb{N}$. The problem of computing generators for $Aut(\Pi)$ is polynomial-time solvable when the set of inputs is restricted to $X(B, C, D)$.*

Proof. Arbitrarily choose Π in $X(B, C, D)$. Compute (in polynomial time) PDG_Π. This colored graph has at most degree $K = K(B, C, D)$ by Lemma 8. By using label structures, we can in polynomial time construct an uncolored undirected graph $\mathcal{G}$ such that $Aut(\mathcal{G}) = Aut(PDG_\Pi)$ by Lemma 1. The graph $\mathcal{G}$ has degree at most $\max\{4, 1 + K\}$. Mathon (1979) has proved that AGEN $\leq_p$ GI. By inspecting Mathon's original proof, one can see that if the input is restricted to graphs with degrees bounded by K', then graph isomorphism testing will exclusively be performed on graphs with degrees bounded by some K'' that only depends on K'. Luks (1982) have shown that the GI problem is polynomial-time solvable when restricted to graphs of bounded degree. The combination of these two facts concludes the theorem. □

Luks' algorithm runs in $n^{O(d)}$ time where d is the maximum degree of the given graph. Algorithms for isomorphism testing of degree-bounded graph is an active area of research: faster algorithms than Luks' original algorithm have been presented by, for instance, Babai, Kantor, and Luks (1983) and Grohe, Neuen, and Schweitzer (2018). The algorithm by Babai et al. runs in $n^{O(d/\log d)}$ time and the algorithm by Grohe et al. runs in $n^{O(\log(d)^c)}$ time for some univeral constant c.

6. General Symmetries of State Transition Graphs

We consider general symmetries of state transition graphs in this section. Computing a set of generators for the automorphism group of an FDR state transition graph is a very costly operation: the state transition graph of a k variable instance contains d^k vertices (where d is the domain size), and the smallest set of generators for the automorphism group may contain up to $d^k - 1$ elements. However, important information may be extracted from the state transition graph anyway. Consider, for instance, the following problem: given a graph $\mathcal{G}$ and two vertices s, t, is there an automorphism that maps s to t? If this problem is efficiently solvable, then it may provide information that is relevant for reducing the search

space even though the full automorphism group (or its generators) is not constructed. It is easy to see that this problem is in **GI** if the graph $\mathcal{G}$ is given explicitly. However, the state transition graph in a planning task is only implicitly given and this may affect the complexity adversely. This is indeed the case: we prove that this problem for FDR planning tasks is **PSPACE**-hard and, thus, at least as hard as the planning problem itself. We do not have matching upper bounds for the symmetry problem so it may, in fact, be computationally harder than the planning problem. Unfortunately, upper bounds for this kind of problem seem hard to obtain. For instance, Das, Scharpfenecker, and Torán (2017) have considered similar problems related to isomorphisms and failed to obtain matching upper and lower bounds.

The rest of this section is divided into two parts. Section 6.1 contains a formal definition of the computational problem that we will analyse together with some auxiliary results, and the **PSPACE**-hardness proof is presented in Section 6.2.

6.1 Computational Problems

We simplify the presentation by using FDR *structures*: we say that $\langle \mathcal{V}, \mathcal{O} \rangle$ is a FDR *structure* whenever $\mathcal{V}$ and $\mathcal{O}$ are sets of variables and actions, respectively. If $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$ is a FDR task, then the state transition graph $\mathcal{T}_\Pi$ is fully determined by the underlying FDR structure $\langle \mathcal{V}, \mathcal{O} \rangle$. Recall that the states of a transition graph are $S = \bigtimes_{v \in \mathcal{V}} \mathcal{D}(v)$ (Definition 1) and this product also defines the states of the FDR structure.

The problem that we will analyse is the following.

Problem. State Transition Graph Symmetry (SGS)
Instance. An FDR structure $\Theta = \langle \mathcal{V}, \mathcal{O} \rangle$ and two states $s, t \in S$.
Question. Does $\mathcal{T}_\Theta$ admit an automorphism $\rho : S \cup \mathcal{O} \to S \cup \mathcal{O}$ such that $\rho(s) = t$?

We remind the reader that $\mathcal{T}_\Theta$ is a labelled graph so an automorphism must preserve both the structure of the graph and its labels, i.e., a labeled edge $(s, s'; o)$ is in $\mathcal{T}_\Theta$ iff its counterpart $(\rho(s), \rho(s'); \rho(o))$ is also an edge of $\mathcal{T}_\Theta$.

All results presented in this section hold even if the planning tasks are restricted to variable domain $\mathcal{D}_2 = \{0, 1\}$ and all action costs are 1. This is a strengthening of the results since we only present hardness results. Note that the state space $S = \mathcal{D}_2^{|\mathcal{V}|}$ can be viewed as a set of 0/1 $|\mathcal{V}|$-dimensional vectors: assume that the variables are ordered $v_1, v_2, \ldots$ and let the i:th entry of the vector denote the value of the i:th variable.

We streamline the forthcoming proofs by making certain assumptions about the FDR structures that we consider. An FDR structure $\Theta = \langle \mathcal{V}, \mathcal{O} \rangle$ is *invertible*[3] if for every $s \in S = \mathcal{D}^{|\mathcal{V}|}$ and every action $o \in \mathcal{O}$ that is applicable in s, there exists an action $o' \in \mathcal{O}$ such that $(s[\![o]\!])[\![o']\!] = s$. We say that o' is the *inverse* of o. Various related concepts appear in the literature and we refer the reader to the paper by Morak, Chrpa, Faber, and Fiser (2020) for an in-depth discussion. If Θ is invertible, then one can view $\mathcal{T}_\Theta$ as an undirected graph. The planning problem becomes no easier when restricted to tasks based on invertible FDR structures.

3. The original term used by Jonsson, Haslum, and Bäckström (2000) is *symmetric*. We do not want to overload this term with yet another meaning.

Lemma 9 (Theorem 18 in Jonsson et al. (2000))**.** *The plan existence problem is **PSPACE**-complete for invertible* FDR *instances with binary domains.*

We slightly strengthen the previous result by showing that the planning problem is **PSPACE**-complete even if we "standardize" the initial and goal state.

Lemma 10. *The plan existence problem is **PSPACE**-complete for invertible* FDR *instances with binary domains, even when restricted to instances $\langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$ (with $\mathcal{V} = \{v_i \mid i \in [n]\}$) such that $s_0 = \{\langle v_i, 0 \rangle \mid i \in [n]\}$ and $G = \{\langle v_i, 1 \rangle \mid i \in [n]\}$.*[4]

Proof. Membership in **PSPACE** is straightforward. We show **PSPACE**-hardness via a polynomial-time reduction from the problem without restrictions on initial and goal states; this problem is **PSPACE**-hard by Lemma 9. Hence, let $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$ be an arbitrary invertible FDR instance with $\mathcal{V} = \{v_i \mid i \in [n]\}$ and domain $\mathcal{D}_2 = \{0, 1\}$. Introduce two fresh variables z_i, z_g and let $\mathcal{V}' = \mathcal{V} \cup \{z_i, z_g\}$. The variable z_i will be used for indicating when we have reached a state that is compatible with the initial state s_0 and z_g for indicating when we have reached a goal state.

We create a new set of actions $\mathcal{O}'$ as follows. First of all, let $\mathcal{O}^+$ contain all actions in $\mathcal{O}$ extended with the precondition $\{\langle z_i, 1 \rangle, \langle z_g, 0 \rangle\}$. Then, let $\mathcal{O}'$ denote the set $\mathcal{O}^+$ together with the following actions.

$$
\begin{array}{lcl}
\text{init}_+ & : & \langle \{\langle v, 0 \rangle \mid v \in \mathcal{V}'\}, s_0 \cup \{\langle z_i, 1 \rangle, \langle z_g, 0 \rangle\} \rangle \\
\text{init}_- & : & \langle s_0 \cup \{\langle z_i, 1 \rangle, \langle z_g, 0 \rangle\}, \{\langle v, 0 \rangle \mid v \in \mathcal{V}'\} \rangle \\
\text{goal}_+ & : & \langle G, \{\langle z_g, 1 \rangle\} \rangle \\
\text{goal}_- & : & \langle G, \{\langle z_g, 0 \rangle\} \rangle \\
\text{chng}(v)_+ & : & \langle \{\langle z_g, 1 \rangle\}, \{\langle v, 1 \rangle\} \rangle \text{ for all } v \in \mathcal{V}' \setminus \{z_g\} \\
\text{chng}(v)_- & : & \langle \{\langle z_g, 1 \rangle\}, \{\langle v, 0 \rangle\} \rangle \text{ for all } v \in \mathcal{V}' \setminus \{z_g\}
\end{array}
$$

We first verify that the structure $\langle \mathcal{V}', \mathcal{O}' \rangle$ is invertible by showing that each action in $\mathcal{O}'$ has an inverse. Every action in $\mathcal{O}^+$ has an inverse since the instance Π is invertible. The action init_+ is only applicable in the state $\langle \{\langle v, 0 \rangle \mid v \in \mathcal{V}'\} \rangle$ and it transforms this state into

$$s_0 \cup \{\langle z_i, 1 \rangle, \langle z_g, 0 \rangle\} = \langle \{\langle v, 0 \rangle \mid v \in \mathcal{V}' \setminus \{z_g\}\} \cup \{\langle z_g, 1 \rangle\} \rangle.$$

Similarly, the action init_- is only applicable in the state $\langle \{\langle v, 0 \rangle \mid v \in \mathcal{V}' \setminus \{z_g\}\} \cup \{\langle z_g, 1 \rangle\} \rangle$ and it transforms this state into $\langle \{\langle v, 0 \rangle \mid v \in \mathcal{V}'\} \rangle$. Hence, init_- is the inverse of init_+ and vice versa. The action goal_+ is applicable in four different states since $G = \{\langle v_i, 1 \rangle \mid i \in [p]\}$. It changes such a state by setting z_g to 1. The action goal_- is applicable in the same four states and it changes z_g to 0. We conclude that goal_- is the inverse of goal_+ and vice versa. Finally, the action $\text{chng}(v)_+$ with $v \in \mathcal{V}' \setminus \{z_g\}$ is applicable in every state with $z_g = 1$ and it sets $v = 1$. We see that $\text{chng}(v)_-$ is its inverse. Analogously, $\text{chng}(v)_+$ is the inverse of $\text{chng}(v)_-$.

We continue by proving that $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$ has a solution if and only if $\Pi' = \langle \mathcal{V}', \mathcal{O}', s_0', G' \rangle$ has a solution where $s_0' = \{\langle v, 0 \rangle \mid v \in \mathcal{V}'\}$ and $G' = \{\langle v, 1 \rangle \mid v \in \mathcal{V}'\}$. Assume that Π has a solution. In the state s_0', action init_+ is applicable and its application yields the state $s_0 \cup \{\langle z_i, 1 \rangle, \langle z_g, 0 \rangle\}$. By using the actions in $\mathcal{O}^+$, we can now reach a state

4. Note that G is a total assignment to the state variables so G can be referred to as a state.

that satisfies G. We apply action goal_+ and then use the actions $\text{chng}(v)_+$ for reaching the goal state G'.

Assume instead that Π does not have a solution. The only action that is applicable in s'_0 is init_+ and its application results in the state $s_0 \cup \{\langle z_i, 1\rangle, \langle z_g, 0\rangle\}$. The instance Π does not have a solution, so we are not able to reach a state satisfying G using the actions in $\mathcal{O}^+$. Note here that the (possibly dangerous) actions $\text{chng}(v)_+$ and $\text{chng}(v)_-$ are not applicable before a state satisfying G has been encountered. We conclude that Π' has no solution. □

6.2 PSPACE-Hardness of SGS

The basic ingredients of our **PSPACE**-hardness result are two methods for constructing planning structures. These two methods make it easier to understand the structure of the state transition graphs of the FDR structures that we work with in the proof. The first one (*direct union*) takes two FDR structures Θ, Θ' and produces an FDR structure whose state transition graph can be viewed as the union of $\mathcal{T}_\Theta$ and $\mathcal{T}_{\Theta'}$ together with a number of isolated vertices. A *hole* in an undirected graph (also known as an *induced simple cycle*) is a simple cycle within the graph such that no two vertices of the cycle are connected by an edge that does not itself belong to the cycle. To simplify the presentation, we will also say that a n-vertex graph *is a hole* if it contains a hole of length n, i.e. the graph is isomorphic to the graph $\langle\{1,\ldots,n\}, \{(i,i+1) \mid i \in [n-1]\} \cup \{n,1\}\rangle$. Our second construction (*hole extension*) takes an FDR structure Φ, a hole length and a state, and produces an FDR structure Ψ whose state transition graphs contain $\mathcal{T}_\Phi$ but with a hole of the specified length attached to the chosen state. The resulting graph will additionally contain a number of isolated vertices, just as the direct union construction.

We begin by presenting the direct union. Let $\Theta = \langle \mathcal{V}, \mathcal{O}\rangle$ and $\Theta' = \langle \mathcal{V}', \mathcal{O}'\rangle$ denote two FDR structures. Assume without loss of generality that $\mathcal{V} \cap \mathcal{V}' = \emptyset$. Introduce a fresh variable z and extend the preconditions of the actions in $\mathcal{O}$ with $\{\langle z, 1\rangle\} \cup \{\langle v, 0\rangle \mid v \in \mathcal{V}'\}$ and the preconditions of the actions in $\mathcal{O}'$ with $\{\langle z, 0\rangle\} \cup \{\langle v, 0\rangle \mid v \in \mathcal{V}\}$. Let $\mathcal{O}''$ be the resulting set of extended actions. We let the *direct union* of Θ and Θ' (denoted $\Theta \Cup \Theta'$) consist of variable set $\mathcal{V} \cup \mathcal{V}' \cup \{z\}$ together with the actions in $\mathcal{O}''$. We refer to z as the *control variable*. It is clear that $\Theta \Cup \Theta'$ can be computed in polynomial time.

The state transition graph $\mathcal{T}_{\Theta \Cup \Theta'}$ can be divided into three distinct parts.

1. A subgraph that is isomorphic to $\mathcal{T}_\Theta$. This subgraph is spanned by the vertices/states such that $z = 1$ and all variables in $\mathcal{V}'$ equal to 0.

2. A subgraph that is isomorphic to $\mathcal{T}_{\Theta'}$. This subgraph is spanned by the vertices/states such that $z = 0$ and all variables in $\mathcal{V}$ equal to 0.

3. A number of isolated vertices that correspond to *mixed states*. A mixed state is a state such that at least one variable in $\mathcal{V}$ equals 1 and at least one variable in $\mathcal{V}'$ equals 1. Due to the construction, such states have no arcs attached to them.

The direct union construction is illustrated in Figure 6.

Before we present the hole extension, we recapitulate the machinery behind *counters*. They have been used for various purposes in the study of planning complexity, cf.

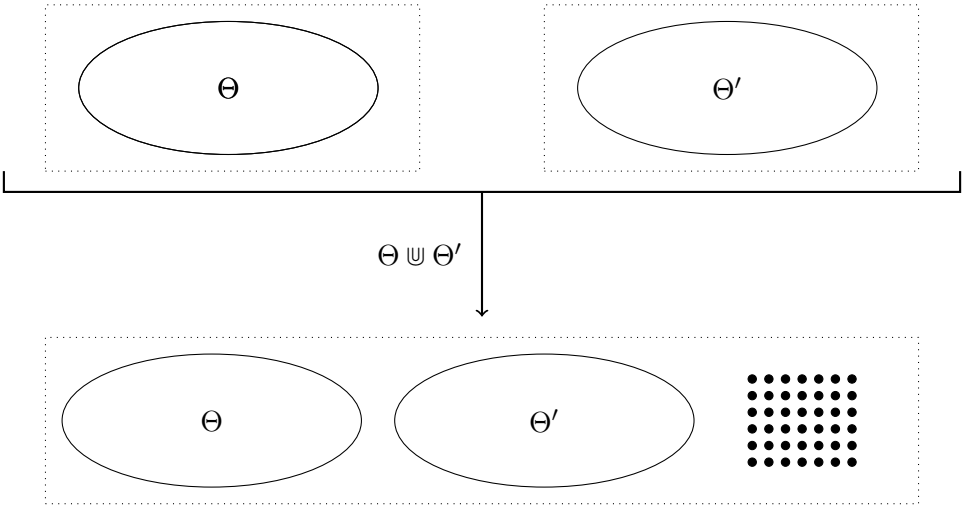


Figure 6: The state-transition graph of the direct union of structures Θ and Θ'. The dots to the right represent mixed states.

Bäckström and Nebel (1995), Jonsson (1999), and Bäckström and Jonsson (2017). Let $\mathcal{V}_n = \{v_1, \ldots, v_n\}$ and let $\mathcal{O}_n$ contain the following $2(n+1)$ actions.

$$
\begin{array}{lcl}
o_1^+ & : & \langle \{\langle v_1, 0\rangle\}, \{\langle v_1, 1\rangle\}\rangle \\
o_i^+ & : & \langle \{\langle v_1, 1\rangle, \langle v_2, 1\rangle, \ldots, \langle v_{i-1}, 1\rangle, \langle v_i, 0\rangle\}, \\
 & & \{\langle v_1, 0\rangle, \langle v_2, 0\rangle, \ldots, \langle v_{i-1}, 0\rangle, \langle v_i, 1\rangle\}\rangle \ \ (2 \leq i \leq n) \\
o_{n+1}^+ & : & \langle \{\langle v_j, 0\rangle \mid j \in [n]\}, \{\langle v_j, 1\rangle \mid j \in [n]\}\rangle \\
 & & \\
o_1^- & : & \langle \{\langle v_1, 1\rangle\}, \{\langle v_1, 0\rangle\}\rangle \\
o_i^- & : & \langle \{\langle v_1, 0\rangle, \langle v_2, 0\rangle, \ldots, \langle v_{i-1}, 0\rangle, \langle v_i, 1\rangle\}, \\
 & & \{\langle v_1, 1\rangle, \langle v_2, 1\rangle, \ldots, \langle v_{i-1}, 1\rangle, \langle v_i, 0\rangle\}\rangle \ \ (2 \leq i \leq n) \\
o_{n+1}^- & : & \langle \{\langle v_j, 1\rangle \mid j \in [n]\}, \{\langle v_j, 0\rangle \mid j \in [n]\}\rangle
\end{array}
$$

Let H_n denote the FDR structure $\langle \mathcal{V}_n, \mathcal{O}_n \rangle$ and note that the state transition graph of H_n corresponds to a hole of length 2^n (see Figure 7).

We continue by defining hole extensions. Given an FDR structure $\Psi = \langle \mathcal{U}, \mathcal{O}_\Psi \rangle$ (where $\mathcal{U} = \{u_i \mid i \in [m]\}$), a total state $s = \{\langle u_1, b_1\rangle, \ldots, \langle u_m, b_m\rangle\}$ (where $b_i \in \mathcal{D}_2$ for each $i \in [m]$), and an integer k, we define an FDR structure $\Phi = \langle \mathcal{V}, \mathcal{O} \rangle$ that combines Ψ with the counter $H_k = (\mathcal{V}_k, \mathcal{O}_k)$. We let $\mathcal{V} = \mathcal{V} \cup \mathcal{U} \cup \{z\}$ (where z is a fresh variable) and $\mathcal{O} = \mathcal{O}'_H \cup \mathcal{O}'_\Psi$ where

- $\mathcal{O}'_H$ contains the actions in the counter H_k extended with the precondition $s \cup \{\langle z, 1\rangle\}$, together with the two actions
  1. $\langle s \cup \{\langle v_1, 0\rangle, \ldots, \langle v_k, 0\rangle, \langle z, 0\rangle\}, \{\langle z, 1\rangle\}\rangle$ and
  2. $\langle s \cup \{\langle v_1, 0\rangle, \ldots, \langle v_k, 0\rangle, \langle z, 1\rangle\}, \{\langle z, 0\rangle\}\rangle$.

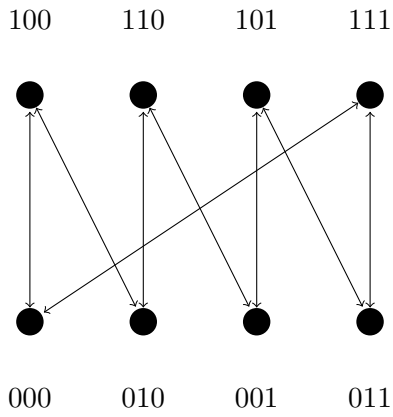


Figure 7: State transition graph of H_3.

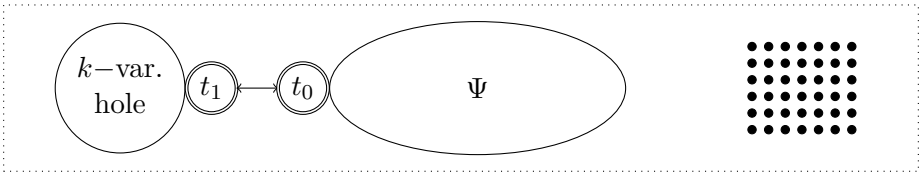


Figure 8: The state-transition graph of structure Ψ extended with an k-variable hole at state $(b_1, \ldots, b_m)$. The state t_0 is $(b_1, \ldots, b_m, 0, \ldots, 0, 0)$ and t_1 is state $(b_1, \ldots, b_m, 0, \ldots, 0, 1)$. The dots to the right represent mixed states.

- $\mathcal{O}'_\Psi$ contains the actions in $\mathcal{O}_\Psi$ extended by preconditions $\{\langle v_1, 0\rangle, \ldots, \langle v_k, 0\rangle, \langle z, 0\rangle\}$.

The state transition graph of structure Φ may be viewed as follows. The graph $\mathcal{T}_\Phi$ contains subgraphs $\mathcal{T}_\Psi$, a hole of length 2^k, and a number of isolated vertices corresponding to mixed states (in the analogous sense as mixed states were used in connection with the direct union). There is just one connection between $\mathcal{T}_\Psi$ and the hole: the states

$$(\underbrace{b_1, \ldots, b_m}_{\mathcal{U}}, \underbrace{0, \ldots, 0}_{\mathcal{V}_k}, \underbrace{0}_{z})$$

and

$$(\underbrace{b_1, \ldots, b_m}_{\mathcal{U}}, \underbrace{0, \ldots, 0}_{\mathcal{V}_k}, \underbrace{1}_{z})$$

are connected by arcs in both directions. Note that this is an important difference between this construction and the direct union of two instances. We say that Ψ is *extended* by a k-variable hole at state s. The construction is depicted in Figure 8. Hole extensions can trivially be computed in polynomial time.

We are now ready to prove the **PSPACE**-hardness of SGS. The formal proof will be discussed in intuitive terms immediately afterwards.

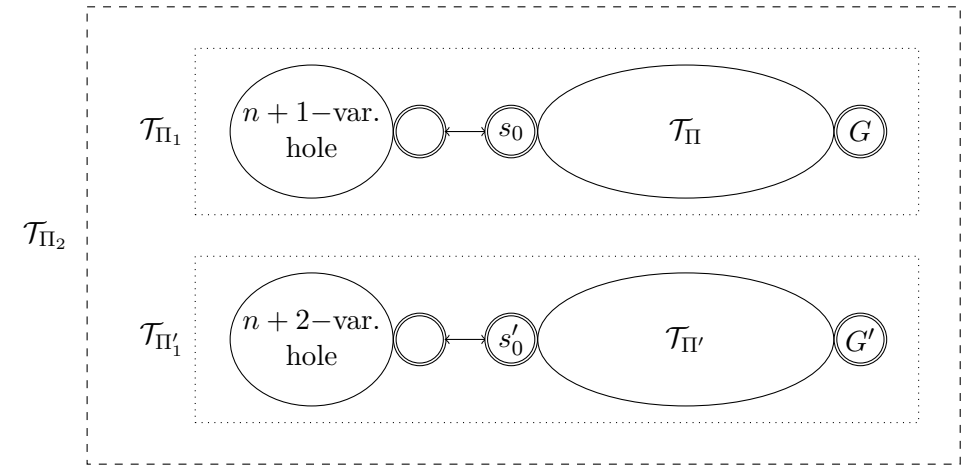


Figure 9: Schematic figure of the state transition graph of Π_2 in Theorem 8. The mixed states are not shown in the figure.

Theorem 8. SGS *is* ***PSPACE****-hard even when restricted to invertible* FDR *structures with variable domain* $\mathcal{D}_2 = \{0, 1\}$.

Proof. We present a polynomial-time reduction from planning tasks that satisfy the conditions of Lemma 10. The reduction will prove the following:

1. if the given FDR task has a solution, then the resulting SGS instance will be a "no"-instance, and
2. if the given FDR task has no solution, then the resulting SGS instance will be a "yes"-instance.

Such a reduction will prove **PSPACE**-hardness of SGS since **PSPACE** is closed under complementation, i.e. **PSPACE**=co-**PSPACE**.

Arbitrarily choose an invertible FDR task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, G \rangle$ with variable domain $\{0, 1\}$ where $\mathcal{V} = \{v_i \mid i \in [n]\}$, $s_0 = \{\langle v_i, 0\rangle \mid i \in [n]\}$, and $G = \{\langle v_i, 1\rangle \mid i \in [n]\}$. The plan existence problem for such FDR tasks is **PSPACE**-complete by Lemma 10. Let $\Pi' = \langle \mathcal{V}', \mathcal{O}', s'_0, G' \rangle$ be the very same task but with variables renamed such that $\mathcal{V}' = \{u_i \mid i \in [n]\}$. Define the following FDR tasks.

- Let Π_1 denote Π extended by an $n+1$-variable hole at state s_0.
- Let Π'_1 denote Π' extended by an $n+2$-variable hole at state s'_0.
- Let Π_2 denote the direct union of Π_1 and Π'_1 with control variable z.

The graph $\mathcal{T}_{\Pi_2}$ is outlined in Figure 9. We observe that the structure Π_2 is invertible and its variable domain is $\{0, 1\}$. We henceforth view a state of Π_2 as a vector

$$(\mathbf{a}, \mathbf{b}, c, \mathbf{d}, \mathbf{e}, f, z),$$

where

1. **a** are the variables in Π_1 corresponding to Π,
2. **b** are the variables in Π_1 corresponding to the $n+1$-variable hole,
3. c is the auxiliary variable in the construction of Π_1,
4. **d** are the variables in Π_1' corresponding to Π,
5. **e** are the variables in Π_1' corresponding to the $n+2$-variable hole,
6. f is the auxiliary variable in the construction of Π_1', and
7. z is the control variable in the construction of Π_2.

We claim that Π has a solution if and only if $\mathcal{T}_{\Pi_2}$ does not admit an automorphism σ that maps

$$(\underbrace{1,\ldots,1}_{\mathbf{a}},\underbrace{0,\ldots,0}_{\mathbf{b}},\underbrace{0}_{c},\underbrace{0,\ldots,0}_{\mathbf{d}},\underbrace{0,\ldots,0}_{\mathbf{e}},\underbrace{0}_{f},\underbrace{1}_{z})$$

into

$$(\underbrace{0,\ldots,0}_{\mathbf{a}},\underbrace{0,\ldots,0}_{\mathbf{b}},\underbrace{0}_{c},\underbrace{1,\ldots,1}_{\mathbf{d}},\underbrace{0,\ldots,0}_{\mathbf{e}},\underbrace{0}_{f},\underbrace{0}_{z}).$$

Colloquially speaking, the automorphism σ swaps the goal states in the two components of Π_2 corresponding to Π_1 and Π_1'.

Assume Π has a solution and that such an automorphism σ exists. The automorphism σ swaps the goal states, and this implies that if there is a path from some state s to G in $\mathcal{T}_{\Pi_1}$, then $\sigma(s)$ must be a state in $\mathcal{T}_{\Pi_1'}$, too. Note that there is a path in $\mathcal{T}_{\Pi_1}$ from s_0 to G. If we look at $\mathcal{T}_{\Pi_1}$, then it contains a hole of length 2^{n+1} connected via a path of length, say, m that ends in the state G. If we instead look at $\mathcal{T}_{\Pi_1'}$, then it contains a hole of length 2^{n+2} connected via a path of length m to G'. The automorphism σ must thus map the hole of length 2^{n+1} in $\mathcal{T}_{\Pi_1}$ into a hole of length 2^{n+1} in $\mathcal{T}_{\Pi_1'}$, and the hole of length 2^{n+2} in $\mathcal{T}_{\Pi_1'}$ into a hole of length 2^{n+2} in $\mathcal{T}_{\Pi_1}$. We know that $\mathcal{T}_{\Pi_1}$ does not contain any holes of length 2^{n+2} – recall that Π only contains n variables. Similarly, $\mathcal{T}_{\Pi_1'}$ does not contain any hole of length 2^{n+1}. Hence, σ does not exist.

Assume Π has no solution. Then both $\mathcal{T}_{\Pi_1}$ and $\mathcal{T}_{\Pi_1'}$ are disconnected graphs where s_0 and G (and s_0' and G') appear in different components. This follows from the fact that Π is invertible and $\mathcal{T}_{\Pi}$ can be viewed as an undirected graph: there is no path from s_0 to G if and only if s_0 and G are members of distinct strongly connected components. Let

$$S = \{G\} \cup \{s \mid \text{there is a path from } s \text{ to } G \text{ in } \mathcal{T}_{\Pi_1}\}$$

and

$$S' = \{G'\} \cup \{s' \mid \text{there is a path from } s' \text{ to } G' \text{ in } \mathcal{T}_{\Pi_1'}\}.$$

The structure Π' is a copy of Π where variable and action names are different. However, the variable and action names in Π and Π' are in a one-to-one correspondence. This implies

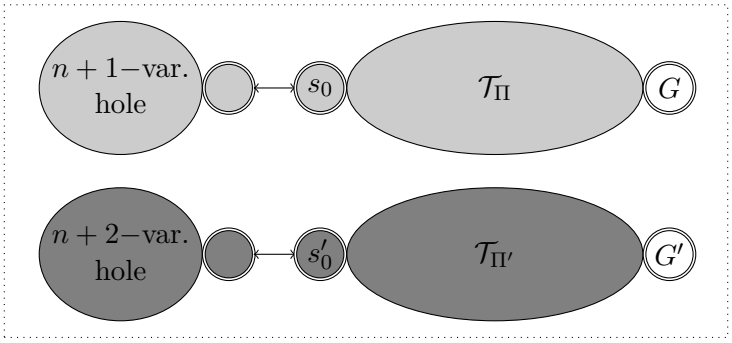


Figure 10: The state transition graph of Π_2 when Π has a solution.

that $\mathcal{T}_\Pi$ and $\mathcal{T}_{\Pi'}$ are isomorphic and it also implies that the graph $\mathcal{T}_{\Pi_1}$ restricted to vertex set S is isomorphic to $\mathcal{T}_{\Pi'_1}$ restricted to vertex set S'. Define a function σ on the states of Π_2 as follows:

$$\sigma(\underbrace{b_1,\ldots,b_n}_{\mathbf{a}},\underbrace{0,\ldots,0}_{\mathbf{b}},\underbrace{0}_{c},\underbrace{0,\ldots,0}_{\mathbf{d}},\underbrace{0,\ldots,0}_{\mathbf{e}},\underbrace{0}_{f},\underbrace{1}_{z}) =$$

$$(\underbrace{0,\ldots,0}_{\mathbf{a}},\underbrace{0,\ldots,0}_{\mathbf{b}},\underbrace{0}_{c},\underbrace{b_1,\ldots,b_n}_{\mathbf{d}},\underbrace{0,\ldots,0}_{\mathbf{e}},\underbrace{0}_{f},\underbrace{0}_{z})$$

if $\{\langle v_1, b_1\rangle,\ldots,\langle v_n, b_n\rangle\} \in S$,

$$\sigma(\underbrace{0,\ldots,0}_{\mathbf{a}},\underbrace{0,\ldots,0}_{\mathbf{b}},\underbrace{0}_{c},\underbrace{b_1,\ldots,b_n}_{\mathbf{d}},\underbrace{0,\ldots,0}_{\mathbf{e}},\underbrace{0}_{f},\underbrace{1}_{z}) =$$

$$(\underbrace{b_1,\ldots,b_n}_{\mathbf{a}},\underbrace{0,\ldots,0}_{\mathbf{b}},\underbrace{0}_{c},\underbrace{0,\ldots,0}_{\mathbf{d}},\underbrace{0,\ldots,0}_{\mathbf{e}},\underbrace{0}_{f},\underbrace{0}_{z})$$

if $\{\langle u_1, b_1\rangle,\ldots,\langle u_n, b_n\rangle\} \in S'$, and $\sigma(x) = x$ otherwise. First note that σ is a bijective function since $\sigma \circ \sigma$ is the identity function. We see that σ swaps the two components that are connected to G and G' but it does not change any other states. We have verified that these two components are isomorphic so σ is indeed an automorphism.

We know that $G = \{\langle v_1, 1\rangle,\ldots,\langle v_n, 1\rangle\}$ and $G' = \{\langle u_1, 1\rangle,\ldots,\langle u_n, 1\rangle\}$. Hence, σ maps

$$(\underbrace{1,\ldots,1}_{\mathbf{a}},\underbrace{0,\ldots,0}_{\mathbf{b}},\underbrace{0}_{c},\underbrace{0,\ldots,0}_{\mathbf{d}},\underbrace{0,\ldots,0}_{\mathbf{e}},\underbrace{0}_{f},\underbrace{1}_{z})$$

into

$$(\underbrace{0,\ldots,0}_{\mathbf{a}},\underbrace{0,\ldots,0}_{\mathbf{b}},\underbrace{0}_{c},\underbrace{1,\ldots,1}_{\mathbf{d}},\underbrace{0,\ldots,0}_{\mathbf{e}},\underbrace{0}_{f},\underbrace{0}_{z}).$$

We conclude that SGS is a **PSPACE**-hard problem. □

We illustrate the proof of Theorem 8 in Figures 10 and 11. Pick a planning task Π (that satisfies the preconditions of the theorem) and assume that it contains n variables. Assume

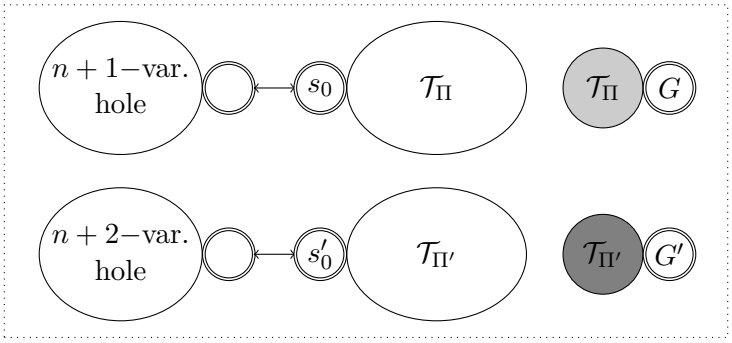


Figure 11: The state transition graph of Π_2 when Π has no solution.

first that the planning task Π has a solution and consider the state transition graph of Π_2 as depicted in Figure 10. The light gray areas correspond to the states that are connected to G and the dark gray areas correspond to the states that are connected to G'. There may additionally be states that are not connected to G or G' but they are not interesting for our argument. If there exists an automorphism σ that moves G into G' and G' into G, then σ must move every light gray state into a dark gray state (in a structure-preserving way) and vice-versa. This is not possible due to the holes of length 2^{n+1} and 2^{n+2}: these two holes are not isomorphic and there cannot exist holes of suitable lengths within the Π_1 and Π'_1 areas since they are based on variable sets with n variables (and thus can contain holes of length at most 2^n).

Assume instead that Π has no solution. The state transition graph of Π is undirected so Π has no solution if and only if the initial state and goal state appears in two distinct connected components of the graph – note that this statement is not true for directed graphs. Thus, the state transition graph of Π_2 has the general appeareance depicted in Figure 11. Here, the light gray states and the dark gray states form isomorphic graphs (since $\mathcal{T}_\Pi$ and $\mathcal{T}_{\Pi'}$ are isomorphic graphs) and there is indeed an automorphism that moves G into G' and G' into G.

7. Conclusions

We have proved that the StructSym problem is **GI**-complete. We pointed out in the introduction that this may be considered both "bad" and "good" news. The "bad" news is that the problem is probably not solvable in polynomial time while the "good" news is that the problem is solvable much faster than most computationally hard problems encountered in AI (under plausible complexity-theoretic assumptions such as the strong exponential time hypothesis).

The "bad" news implies that there are reasons to identify classes of planning tasks where the StructSym problem can be solved in polynomial time – we are obviously interested in finding structural symmetries as fast as possible. The tractability results in Section 5 exploit connections between structural symmetries and the causal graph: in particular, the degree of the causal graph turns out to be important. A natural starting point for analysing causal graphs with unbounded degree is to consider special cases such as *forks*, *inverse forks*,

and more generally, *polytrees*. Even though the structure of such causal graphs may appear simple, we do not understand the borderline between tractability and **GI**-hardness of the corresponding StructSym problem. A more ambitious project is to study problems with acyclic causal graphs and where certain graph-theoretic parameters are bounded. One interesting parameter is the *treewidth* of the undirected graph underlying the causal graph; this parameter has been exploited by, for instance, Brafman and Domshlak (2006). A broad range of combinatorial problems defined on graphs can be efficiently solved by dynamic programming as long as the graphs under consideration have treewidth bounded by some constant. It is thus not very surprising that the graph isomorphism problem restricted to graphs of bounded tree-width can be solved in polynomial-time (Bodlaender, 1990). With this in mind, it seems plausible that polynomial-time solvable cases can be identified this way. We additionally note that this parameter connects in a natural way to polytrees since the underlying undirected graph of a polytree is a tree and thus has treewidth 1.

The "good" news implies that there are reasons to investigate stronger symmetries than the structural symmetries considered in this article: there may exist symmetries that are better at pruning the search space and at the same time being reasonably efficient to compute. The main result of Section 6 shows that defining such symmetries is a non-trivial task: considering automorphisms of the full state transition graph is a natural idea, but we immediately run into problems. The set of generators for the automorphism group is huge and may, in the worst case, take exponential time just to write down. If we circumvent this problem by accessing the automorphism group via the State Transition Graph Symmetry problem, then this problem is **PSPACE**-hard and consequently at least as hard as the FDR planning problem itself. We are thus facing a delicate balancing problem between pruning power and computational complexity. A possible source of inspiration for future work is *factored symmetries* that have been introduced in the context of merge-and-shrink heuristics (Sievers, Wehrle, Helmert, Shleyfman, & Katz, 2015).

Acknowledgments

Alexander Shleyfman is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities. Peter Jonsson is partially supported by the Swedish Research Council (VR) under grant 2017-04112.

References

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

Babai, L. (2016). Graph isomorphism in quasipolynomial time [extended abstract]. In *Proc. 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC-2016)*, pp. 684–697.

Babai, L., Kantor, W. M., & Luks, E. M. (1983). Computational complexity and the classification of finite simple groups. In *Proc. 24th IEEE Annual Symposium on Foundations of Computer Science (FOCS-1983)*, pp. 162–171.

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

Bäckström, C., & Jonsson, P. (2017). Time and space bounds for planning. *Journal of Artificial Intelligence Research*, *60*, 595–638.

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

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

Bodlaender, H. L. (1990). Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, *11*(4), 631–643.

Brafman, R. I., & Domshlak, C. (2006). Factored planning: How, when and when not. In *Proc. 21st National Conference on Artificial Intelligence (AAAI-2006)*, pp. 809–814.

Das, B., Scharpfenecker, P., & Torán, J. (2017). CNF and DNF succinct graph encodings. *Information and Computation*, *253*, 436–447.

Domshlak, C., & Brafman, R. I. (2002). Structure and complexity in planning with unary operators. In *Proc. 6th International Conference on Artificial Intelligence Planning Systems (AIPS-2002)*, pp. 34–43.

Domshlak, C., Katz, M., & Shleyfman, A. (2012). Enhanced symmetry breaking in cost-optimal planning as forward search. In *Proc. 22nd International Conference on Automated Planning and Scheduling (ICAPS-2012)*.

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

Fišer, D., Torralba, Á., & Shleyfman, A. (2019). Operator mutexes and symmetries for simplifying planning tasks. In *Proc. 33rd AAAI Conference on Artificial Intelligence (AAAI-2019)*, pp. 7586–7593.

Fox, M., & Long, D. (1999). The detection and exploitation of symmetry in planning problems. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI-1999)*, pp. 956–961.

Ghosh, S., & Kurur, P. P. (2014). Permutation groups and the graph isomorphism problem. *Progress in Computer Science and Applied Logic (Perspectives in Computational Complexity)*, *26*, 183–202.

Giménez, O., & Jonsson, A. (2009). Planning over chain causal graphs for variables with domains of size 5 is NP-hard. *Journal of Artificial Intelligence Research*, *34*, 675–706.

Gnad, D., Torralba, Á., Shleyfman, A., & Hoffmann, J. (2017). Symmetry breaking in star-topology decoupled search. In *Proc. 27th International Conference on Automated Planning and Scheduling (ICAPS-2017)*, pp. 125–134.

Grohe, M., Neuen, D., & Schweitzer, P. (2018). A faster isomorphism test for graphs of small degree. In *Proc. 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS-2018)*, pp. 89–100.

Haslum, P., Helmert, M., & Jonsson, A. (2013). Safe, strong, and tractable relevance analysis for planning. In *Proc. 23rd International Conference on Automated Planning and Scheduling (ICAPS-2013)*, pp. 317–321.

Helfgott, H., Bajpai, J., & Dona, D. (2017). Graph isomorphisms in quasi-polynomial time. *CoRR*, *abs/1710.04574*.

Helmert, M. (2004). A planning heuristic based on causal graph analysis. In *Proc. 14th International Conference on Automated Planning and Scheduling (ICAPS-2004)*, pp. 161–170.

Helmert, M. (2009). Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, *173*, 503–535.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In *Proc. 19th International Conference on Automated Planning and Scheduling (ICAPS-2009)*, pp. 162–160.

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

Hopcroft, J. E., & Wong, J. K. (1974). Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proc. 6th Annual ACM Symposium on Theory of Computing (STOC-1974)*, pp. 172–184.

Impagliazzo, R., & Paturi, R. (2001). On the complexity of k-SAT. *Journal of Computer and System Sciences*, *62*(2), 367–375.

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

Jonsson, P. (1999). Strong bounds on the approximability of two PSPACE-hard problems in propositional planning. *Annals of Mathematics and Artificial Intelligence*, *26*(1-4), 133–147.

Jonsson, P., & Bäckström, C. (1998). Tractable plan existence does not imply tractable plan generation. *Annals of Mathematics and Artificial Intelligence*, *22*(3), 281–296.

Jonsson, P., Haslum, P., & Bäckström, C. (2000). Towards efficient universal planning: A randomized approach. *Artificial Intelligence*, *117*(1), 1–29.

Kelly, P. J. (1957). A congruence theorem for trees. *Pacific Journal of Mathematics*, *7*, 961–968.

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

Ladner, R. E. (1975). On the structure of polynomial time reducibility. *Journal of the ACM*, *22*(1), 155–171.

Luks, E. M. (1982). Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, *25*(1), 42–65.

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

Morak, M., Chrpa, L., Faber, W., & Fiser, D. (2020). On the reversibility of actions in planning. In *Proc. 17th International Conference on Principles of Knowledge Representation and Reasoning (KR-2020)*, pp. 652–661.

Pochter, N., Zohar, A., & Rosenschein, J. S. (2011). Exploiting problem symmetries in state-based planners. In *Proc. 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pp. 1004–1009.

Rintanen, J. (2003). Symmetry reduction for SAT representations of transition systems. In *Proc. 13th International Conference on Automated Planning and Scheduling (ICAPS-2013)*, pp. 32–41.

Röger, G., Sievers, S., & Katz, M. (2018). Symmetry-based task reduction for relaxed reachability analysis. In *Proc. 28th International Conference on Automated Planning and Scheduling (ICAPS-2018)*, pp. 208–217.

Schöning, U. (1987). Graph isomorphism is in the low hierarchy. In *Proc. 4th Annual Symposium on Theoretical Aspects of Computer Science (STACS-1987)*, pp. 114–124.

Shleyfman, A. (2019). On computational complexity of automorphism groups in classical planning. In *Proc. 29th International Conference on Automated Planning and Scheduling, (ICAPS-2019)*, pp. 428–436.

Shleyfman, A. (2020). *Symmetry Breaking and Operator Pruning in Classical Planning and Beyond*. Ph.D. thesis, Technion – Israel Institute of Technology.

Shleyfman, A., Katz, M., Helmert, M., Sievers, S., & Wehrle, M. (2015). Heuristics and symmetries in classical planning. In *Proc. 33rd AAAI Conference on Artificial Intelligence (AAAI-2019)*, pp. 3371–3377.

Sievers, S., Röger, G., Wehrle, M., & Katz, M. (2019). Theoretical foundations for structural symmetries of lifted PDDL tasks. In *Proc. 29th International Conference on Automated Planning and Scheduling (ICAPS-2019)*, pp. 446–454.

Sievers, S., Wehrle, M., Helmert, M., Shleyfman, A., & Katz, M. (2015). Factored symmetries for merge-and-shrink abstractions. In *Proc. 29th AAAI Conference on Artificial Intelligence (AAAI-2015)*, pp. 3378–3385.

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. (2012). About partial order reduction in planning and computer aided verification. In *Proc. 22nd International Conference on Automated Planning and Scheduling (ICAPS-2012)*, pp. 297–305.

Wehrle, M., Helmert, M., Shleyfman, A., & Katz, M. (2015). Integrating partial order reduction and symmetry elimination for cost-optimal classical planning. In *Proc. 24th International Joint Conference on Artificial Intelligence (IJCAI-2015)*, pp. 1712–1718.

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