Theoretical Foundations for Structural Symmetries of Lifted PDDL Tasks

Silvan Sievers and **Gabriele Röger** and **Martin Wehrle**
University of Basel
Basel, Switzerland
{silvan.sievers,gabriele.roeger}@unibas.ch

Michael Katz
IBM Research
Yorktown Heights, NY, USA
michael.katz1@ibm.com

Abstract

We transfer the notion of structural symmetries to lifted planning task representations, based on abstract structures which we define to model planning tasks. We show that symmetries are preserved by common grounding methods and we shed some light on the relation to previous symmetry concepts used in planning. Using a suitable graph representation of lifted tasks, our experimental analysis of common planning benchmarks reveals that symmetries occur in the lifted representation of many domains. Our work establishes the theoretical ground for exploiting symmetries beyond their previous scope, such as for faster grounding and mutex generation, as well as for state space transformations and reductions.

Introduction

In domain-independent classical planning, symmetries have been intensively investigated to increase the scalability of planning systems (Fox and Long 1999; 2002; Rintanen 2003; Pochter, Zohar, and Rosenschein 2011; Rintanen and Gretton 2013; Domshlak, Katz, and Shleyfman 2012; 2013; 2015; Abdulaziz, Norrish, and Gretton 2015; Sievers et al. 2015b; Wehrle et al. 2015). In particular, Shleyfman et al. (2015) introduced *structural symmetries* as a declarative notion over the representation of propositional STRIPS tasks. These symmetries subsume some earlier symmetry definitions for classical planning, many of which focus on *object* symmetries (Fox and Long 1999; 2002; Riddle et al. 2016). They also generalize further types of symmetries considered for other state-space search problems, e. g. *rotation* and *reflection* (Huber et al. 1986) or *scalarset permutations* (Ip and Dill 1996).

In practice, planning tasks are usually given in a compact *lifted* PDDL description, which is, however, not directly supported by most planning techniques. Instead, they first transform it into a much larger ground representation. Also most of the recent symmetry-based approaches operate only on this ground representation, including techniques based on structural symmetries. However, reasoning about symmetries for applications that work directly on the lifted representation requires a general concept of symmetries of the lifted representation.

In this work, we transfer the notion of *structural* symmetries to the lifted representation. Our aim is to build the theoretical basis for many promising future applications of such symmetries. While these symmetries could be grounded to be used for existing symmetry-based approaches that operate on the ground representation, this is not our intended application, and we will show that there is no theoretical gain in doing so. Instead, structural symmetries of the lifted representation can be used for all purposes operating on lifted representations. In particular, we see potential benefit for state space transformations and reductions on that level.

For transferring the concept of structural symmetries to the lifted representation, we model planning tasks based on a very general concept of *abstract structures*, unlike previous work also covering axioms and conditional effects. We show that lifted structural symmetries are preserved by common grounding methods and how they are related to previous symmetry concepts. We also introduce a graph representation of abstract structures that allows to compute structural symmetries of the abstract structures and thus the planning tasks they represent. A quantitative analysis of planning benchmarks from the International Planning Competitions (IPC) reveals a large number of symmetries in these benchmarks. We conclude the paper with a discussion of initial existing and potential future applications of lifted structural symmetries.

Structural Symmetries

To separate the concept of structural symmetries from the specific application to planning, we first introduce *abstract structures*, which we will later use for representing planning tasks.

Definition 1. *Let S be a set of symbols, where each $s \in S$ is associated with a* symbol type $t(s)$. *The set of* abstract structures *over S is inductively defined as follows:*

- *each symbol $s \in S$ is an abstract structure, and*
- *for abstract structures $A_1, \dots, A_n$, the set $\{A_1, \dots, A_n\}$ and the tuple $\langle A_1, \dots, A_n \rangle$ are abstract structures.*

For an example, consider the set $S = \{a, b, c, \dots, j\}$, where a, b, c have type t_1 and $d, \dots, j$ have type t_2. $A = \{\langle a, \{d, e\}\rangle, \langle b, \{f, g\}\rangle, \langle d, \{h\}\rangle, \langle f, \{h\}\rangle, \langle \{i\}, c, j\rangle\}$ is a possible abstract structure over S.

Informally speaking, a structural symmetry for an abstract structure is a permutation of the symbols that preserves the structure as well as the types of the symbols.

Definition 2. *A* symbol mapping σ *over a set of symbols* S *is a permutation of* S *such that for all* $s \in S : t(\sigma(s)) = t(s)$.

In the example, permutation σ_1 that swaps a with b, d with f and e with g and maps all other symbols onto themselves is a symbol mapping over S. Any permutation that maps a on d is not a symbol mapping because $t(a) = t_1 \neq t_2 = t(d)$.

Definition 3. *For an abstract structure* A *over* S *and a symbol mapping* σ *over* S*, the* abstract structure mapping $\tilde{\sigma}(A)$ *is defined as follows:*

$$\tilde{\sigma}(A) := \begin{cases} \sigma(A) & \text{if } A \in S \\ \{\tilde{\sigma}(A_1), \dots, \tilde{\sigma}(A_n)\} & \text{if } A = \{A_1, \dots, A_n\} \\ \langle\tilde{\sigma}(A_1), \dots, \tilde{\sigma}(A_n)\rangle & \text{if } A = \langle A_1, \dots, A_n\rangle \end{cases}$$

We call σ *a* structural symmetry *for* A *if* $\tilde{\sigma}(A) = A$.

The identity function on S is a trivial structural symmetry. In the example, also the earlier symbol mapping σ_1 is a structural symmetry for A because $\tilde{\sigma}_1(A) = \{\langle b, \{f, g\}\rangle, \langle a, \{d, e\}\rangle, \langle f, \{h\}\rangle, \langle d, \{h\}\rangle, \langle\{i\}, c, j\rangle\} = A$. In contrast, symbol mapping σ_2 that swaps a with b, d with g and e with f (and keeps all other symbols stable) is *not* a structural symmetry for A: while its application preserves some parts of A, it does not preserve *all* of them. For example, $\tilde{\sigma}_2(\langle a, \{d, e\}\rangle) = \langle b, \{g, f\}\rangle$ and vice versa and both are in set A but $\tilde{\sigma}_2(\langle d, \{h\}\rangle) = \langle g, \{h\}\rangle \notin A$.

We establish that the set of all structural symmetries for an abstract structure A forms a group under function composition. We will not only exploit this property in later theorems but it will also provide the basis for the actual computation of such symmetries.

Lemma 1. *Given an abstract structure* A *over a finite set of symbols* S*, let* $\Gamma(A)$ *be the set of all structural symmetries for* A*. Then* $\Gamma(A)$ *is a group.*

Proof. To show that a set of permutations of a finite set forms a group under composition, it is sufficient to show that it is nonempty and closed under composition. It is easy to verify that the identity symbol mapping always is a structural symmetry, and for $\sigma_1, \sigma_2 \in \Gamma(A)$ also $\sigma := \sigma_1 \circ \sigma_2 \in \Gamma(A)$ because $\tilde{\sigma}(A) = \tilde{\sigma}_1(\tilde{\sigma}_2(A)) = \tilde{\sigma}_1(A) = A$. $\square$

Planning Tasks as Abstract Structures

To apply the general notion of structural symmetries to planning, we define planning tasks as abstract structures.

Definition 4. *We call a finite set of symbols* S *a* set of symbols for planning *if the associated symbol types are from* $\{\mathit{Object}, \mathit{Variable}, \mathit{FluentPredicate}, \mathit{DerivedPredicate}, \mathit{Function}, \mathit{Negation}\} \cup \mathbb{N}$ *and there is at most one symbol of type* $\mathit{Negation}$.

We also refer to symbols of type T as T symbols. Let S be a set of symbols for planning. For convenience, we define some notions for abstract structures over S:

- An *atom* is a tuple $\langle P, x_1, \dots, x_n\rangle$ of symbols with $t(P) \in \{\mathit{FluentPredicate}, \mathit{DerivedPredicate}\}$, and for $i \in \{1 \dots, n\}$, $t(x_i) \in \{\mathit{Object}, \mathit{Variable}\}$. The atom is *fluent* if $t(P) = \mathit{FluentPredicate}$, otherwise it is *derived*.
- A *literal* is either an atom or an abstract structure $\langle\neg, A\rangle$ where $t(\neg) = \mathit{Negation}$ and A is an atom.
- A *function term* is a tuple $\langle f, x_1, \dots, x_n\rangle$ of symbols with $t(f) = \mathit{Function}$, and for $i \in \{1 \dots, n\}$, $t(x_i) \in \{\mathit{Object}, \mathit{Variable}\}$.
- A *function assignment* is a tuple $\langle F, v\rangle$ where F is a function term and v is a symbol with $t(v) \in \mathbb{N}$.

Structures without $\mathit{Variable}$ symbols are called *ground*.

Definition 5. *A* planning task *is an abstract structure* $\Pi = \langle \mathcal{O}, \mathcal{A}, s_0, s_\star\rangle$ *over a set of symbols* S *for planning, where*

- $\mathcal{O}$ *is a set of operators, each of the form* $o = \langle \mathit{params}, \mathit{pre}, \mathit{eff}, \mathit{cost}\rangle$ *where*
  - params *is a set of* $\mathit{Variable}$ *symbols,*
  - pre *is a set of literals where all occurring variables are from* params*,*
  - eff *is a set of universally quantified conditional effects, each of the form* $\langle \mathit{vars}, \mathit{cond}, \mathit{lit}\rangle$*, where* vars *is a set of* $\mathit{Variable}$ *symbols,* cond *is a set of literals and* lit *is a literal of a fluent atom. All variables occurring in* cond *and* lit *are from* $\mathit{params} \cup \mathit{vars}$.
  - cost *is a symbol* v *with* $t(v) \in \mathbb{N}$ *or a function term where all occurring variables are from* params*;*
- $\mathcal{A}$ *is a set of axioms, each of the form* $a = \langle \mathit{params}, \mathit{pre}, \mathit{eff}\rangle$ *where* params *and* pre *are defined as for operators above and* eff *is a derived atom where all occurring variables are from* params*. The set of axioms must be stratifiable.*[1]
- s_0 *is a set of fluent ground atoms and consistent ground function assignments, i. e. assignments with identical function term are identical;*
- $s_\star$ *is a set of ground literals.*

W.l.o.g. all occurring sets of $\mathit{Variable}$ *symbols are disjoint.*

We assume syntactically unique elements, such as operators, axioms, etc., and thus there is no need for naming the elements, which can be identified by their structure.

This definition of planning tasks corresponds to *normalized PDDL planning tasks* as used by Helmert (2009), extended with support for action costs. We refer to such a planning task as *lifted* task or as *lifted representation* of a task.

Figure 1 shows an operator (from the SPANNER domain) in PDDL representation and as abstract structure. The operator picks up a spanner at a location. Since it does not have all-quantified effects or (non-trivial) effect conditions, the first two elements of each effect are empty.

A *ground* planning task (or *ground representation* of a task) contains no $\mathit{Variable}$ symbols. The semantics of a (lifted) planning task is defined via its induced ground planning task, which we define in the following.

[1]Stratifiability (Thiébaux, Hoffmann, and Nebel 2005) ensures that the result of axiom evaluation is well-defined.

```
(:action pick-up
        :parameters (?s ?l)
        :precondition
            (and (LOCATION ?l)
                 (SPANNER ?s)
                 (bob-at ?l)
                 (spanner-at ?s ?l))
        :effect
            (and (not (spanner-at ?s ?l))
                 (carrying ?s)
                 (increase (total-cost) 1)))
```

$\langle\{s, l\},$
$\{\langle \mathit{location}, l\rangle, \langle \mathit{spanner}, s\rangle, \langle \mathit{bob\text{-}at}, l\rangle, \langle \mathit{spanner\text{-}at}, s, l\rangle\},$
$\{\langle\emptyset, \emptyset, \langle\neg, \langle \mathit{spanner\text{-}at}, s, l\rangle\rangle, \langle\emptyset, \emptyset, \langle \mathit{carrying}, s\rangle\}, 1\rangle$

Figure 1: Operator from the SPANNER domain in PDDL representation and as abstract structure.

For a set S of symbols for planning, we define $\mathit{Objs}(S) = \{s \in S \mid t(s) = \mathit{Object}\}$. For sets X and Y, we denote the set of all functions $f : X \to Y$ by X^Y. We call a function m that maps from the *Variable* symbols in S to $\mathit{Objs}(S)$ a *variable mapping* and write $\tilde{m}(S)$ for the natural extension of m to abstract structures, where symbols outside the domain of m are mapped to themselves.

Grounding instantiates operators and axioms with all possible variable assignments and expands universal effects.

Definition 6. *For a (lifted) planning task* $\Pi = \langle\mathcal{O}, \mathcal{A}, s_0, s_\star\rangle$ *over* S*, the* induced ground *planning task is defined as* $\mathit{ground}(\Pi) = \langle \mathit{ground}(\mathcal{O}), \mathit{ground}(\mathcal{A}), s_0, s_\star\rangle$ *over* S *with*

- $\mathit{ground}(\mathcal{O}) = \bigcup_{o\in\mathcal{O}} \mathit{opground}(o)$*, where*
$$\begin{aligned}&\mathit{opground}(\langle \mathit{params}, \mathit{pre}, \mathit{eff}, \mathit{cost}\rangle)\\ &\quad = \{\langle\emptyset, \tilde{m}(\mathit{pre}), \tilde{m}(\mathit{expand}(\mathit{eff})), \tilde{m}(\mathit{cost})\rangle \mid\\ &\qquad m \in \mathit{params}^{\mathit{Objs}(S)}\}, \text{ with}\\ &\mathit{expand}(\mathit{eff})\\ &\quad = \{\langle\emptyset, \tilde{n}(\mathit{cond}), \tilde{n}(\mathit{lit})\rangle \mid\\ &\qquad \langle \mathit{vars}, \mathit{cond}, \mathit{lit}\rangle \in \mathit{eff}, n \in \mathit{vars}^{\mathit{Objs}(S)}\}\end{aligned}$$
- $\mathit{ground}(\mathcal{A}) = \bigcup_{a\in\mathcal{A}} \mathit{axground}(a)$*, where*
$$\begin{aligned}&\mathit{axground}(\langle \mathit{params}, \mathit{pre}, \mathit{eff}\rangle)\\ &\quad = \{\langle\emptyset, \tilde{m}(\mathit{pre}), \tilde{m}(\mathit{eff})\rangle \mid m \in \mathit{params}^{\mathit{Objs}(S)}\}.\end{aligned}$$

A *state* of a ground planning task $\Pi = \langle\mathcal{O}, \mathcal{A}, s_0, s_\star\rangle$ over S is a set of ground atoms. A *fluent* state s is a subset of the fluent ground atoms. The associated *derived* state $[\![s]\!]$ results from s by evaluating the axioms as in stratified logic programming. A state s *satisfies* a set C of ground literals if all atoms in C are also in s, no negated atom from C occurs (positively) in s and C is consistent. A ground operator $o = \langle\emptyset, \mathit{pre}, \mathit{eff}, \mathit{cost}\rangle$ is *applicable* in a fluent state s if $[\![s]\!]$ satisfies pre and s_0 contains a function assignment for cost if it is a function term. The (fluent) successor state $s[o]$ contains a fluent ground atom a if there is an effect $\langle\emptyset, \mathit{cond}, a\rangle \in \mathit{eff}$ such that $[\![s]\!]$ satisfies cond or if $a \in s$ and there is no $\langle\emptyset, \mathit{cond}, \langle\neg, a\rangle\rangle \in \mathit{eff}$ where $[\![s]\!]$ satisfies cond. The (fluent) *initial state* consists of the atoms in s_0.

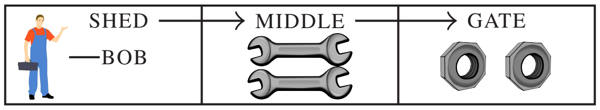

Figure 2: Exemplary initial state of a SPANNER task.

The semantics of Π can naturally be represented via its induced *transition graph*, which is the labeled transition system $\mathcal{T}_\Pi = \langle D, L, T, [\![s_0]\!], G\rangle$ where D is the set of derived states of Π, L corresponds to $\mathcal{O}$, and whenever $o = \langle\emptyset, \mathit{pre}, \mathit{eff}, \mathit{cost}\rangle \in \mathcal{O}$ is applicable in fluent state s, there is a transition $\langle[\![s]\!], o, [\![s[o]]\!]\rangle \in T$ labeled with o. The cost of the transition is the value assigned to cost in s_0 if it is a function term and its (natural number) type otherwise. The set of goal states G consists of all $s \in D$ that satisfy $s_\star$. A *plan* for Π is a sequence of labels (thus operators) along a path in $\mathcal{T}_\Pi$ from $[\![s[o]]\!]$ to a state from G. Its *cost* is the accumulated label costs of the sequence. *Satisficing* planning deals with finding plans of any cost whereas *optimal* planning is only interested in plans with minimal cost among all plans. For a lifted planning task, its transition graph is defined as the transition graph of the induced ground task.

As a running example, consider a planning task of the IPC domain SPANNER with the initial state shown in Figure 2. The goal of BOB, initially at the location SHED, is to tighten the two nuts NUT1 and NUT2 located at the GATE, using the spanners SP1 and SP2, initially at the location MIDDLE. It does not matter which spanner is used for which nut, but spanners can only be used once. Operators MOVE(X, Y) move BOB from X to Y, however there are only one-way connections from the SHED to the MIDDLE and from the MIDDLE to the GATE. Operators PICK-UP(X, Y) let BOB pick up the spanner X at location Y, and once picked up, spanners cannot be dropped again.

In the lifted representation of the task, there are two structural symmetries: the two spanners are symmetric to each other, and so are the two nuts, because both the spanners and the nuts are at the same location initially, the nuts both need to be tightened in the goal, and the same operators work with the spanners and the nuts, respectively. In the abstract structure modeling the planning task, both the spanners and the nuts are modeled as symbols (because they are PDDL objects), and hence the two mentioned structural symmetries permute the corresponding symbols and all abstract (sub)structures of the planning tasks where the spanners or nuts are mentioned.

Note that our symmetries *stabilize* both the initial state and the goal, i. e. parts of a planning task can only be considered symmetric if they are symmetric in the initial state and the goal. This differs from structural symmetries of a ground representation in previous work; e. g. Shleyfman et al. (2015) do not stabilize the initial state because it is not necessary for symmetry-based pruning in a forward search. With a lifted representation, not stabilizing the initial state causes some difficulties due to the specification of PDDL: function assignments and all “static” information (e.g. hardcoded connectivity information) are specified in the initial state, and this information would be lost. However, all applications we have in mind are based on a reachability analysis, for which stabilizing the initial state is essential. In contrast, for most applications we do not need to stabilize the goal, which we can achieve by simply dropping it from the abstract structure.

Structural Symmetries and Grounding

To ensure that our symmetries can also be applied to ground representations and hence are also symmetries in the sense of previous work, we will first establish that structural symmetries of the lifted representation are also structural symmetries of the *induced* ground representation, and then discuss this issue in the light of *optimized* grounding.

Theorem 1. *Let σ be a symbol mapping over S. If σ is a structural symmetry for a planning task $\Pi = \langle \mathcal{O}, \mathcal{A}, s_0, s_\star \rangle$ over S, then σ is a structural symmetry for $\mathit{ground}(\Pi)$.*

Proof. For better readability, in the following we use subscripts $_\downarrow$ to denote ground abstract structures in contrast to lifted ones. We have to show that $\mathit{ground}(\Pi) = \tilde{\sigma}(\mathit{ground}(\Pi))$ and start with $\mathit{ground}(\mathcal{A}) = \tilde{\sigma}(\mathit{ground}(\mathcal{A}))$. Consider $a_\downarrow = \langle \emptyset, \mathit{pre}_\downarrow, \mathit{eff}_\downarrow \rangle \in \mathit{ground}(\mathcal{A})$. Since $a_\downarrow$ is in $\mathit{ground}(\mathcal{A})$ there must be an axiom $a = \langle \mathit{params}, \mathit{pre}, \mathit{eff} \rangle \in \mathcal{A}$ and a variable mapping m such that $a_\downarrow \in \mathit{axground}(a)$. Since σ is a structural symmetry of $\mathcal{A}$, also $\tilde{\sigma}(a) \in \mathcal{A}$. Consider $m' := \sigma \circ m \circ \sigma^{-1} \in \tilde{\sigma}(\mathit{params})^{\mathit{Objs}(S)}$. Variable mapping m' grounds $\tilde{\sigma}(a)$ to $a' := \langle \emptyset, \tilde{m}'(\tilde{\sigma}(\mathit{pre})), \tilde{m}'(\tilde{\sigma}(\mathit{eff})) \rangle \in \mathit{axground}(\tilde{\sigma}(a))$. It holds that $\tilde{m}'(\tilde{\sigma}(\mathit{pre})) = \{\tilde{m}' \circ \tilde{\sigma}(p) \mid p \in \mathit{pre}\} = \{\tilde{\sigma} \circ \tilde{m}(p) \mid p \in \mathit{pre}\} = \tilde{\sigma}(\{\tilde{m}(p) \mid p \in \mathit{pre}\}) = \tilde{\sigma}(\mathit{pre}_\downarrow)$ (*). Analogously, we can show that $\tilde{m}'(\tilde{\sigma}(\mathit{eff})) = \tilde{\sigma}(\mathit{eff}_\downarrow)$, so overall $a' = \tilde{\sigma}(a_\downarrow)$. Therefore, for each $a \in \mathit{ground}(\mathcal{A})$, also $\tilde{\sigma}(a)$ is in $\mathit{ground}(\mathcal{A})$, and, since $\tilde{\sigma}$ is a permutation on Π, $\tilde{\sigma}(\mathit{ground}(\mathcal{A})) = \mathit{ground}(\mathcal{A})$.

Establishing $\mathit{ground}(\mathcal{O}) = \tilde{\sigma}(\mathit{ground}(\mathcal{O}))$ technically works analogously, with the addition that we also have to use argument (*) for the effect condition of operators.

As s_0 and $s_\star$ of the induced ground task are the same as in the lifted task, we immediately get $\tilde{\sigma}(s_0) = s_0$ and $\tilde{\sigma}(s_\star) = s_\star$, and hence overall $\mathit{ground}(\Pi) = \tilde{\sigma}(\mathit{ground}(\Pi))$. □

In practice, the induced ground task is typically too large to be represented and computed in reasonable time (e. g., Helmert 2009). We say that a grounding algorithm is *optimized* if it removes (some, not necessarily all) irrelevant parts of the task representation (e. g., Köhler and Hoffmann 2000). Such grounding is *correct* if the reachable part of the transition graph is not affected. We write $\mathit{ground}_{\mathit{opt}}(\Pi)$ for a correct optimized grounding of Π.

Observation 1. *A structural symmetry for a planning task Π is not necessarily a structural symmetry for $\mathit{ground}_{\mathit{opt}}(\Pi)$.*

As an example for this observation, consider again the SPANNER task in Figure 2. As we have seen before, in the lifted representation, the spanners are symmetric to each other, and so are the nuts. However, consider the ground representation $\mathit{ground}_{\mathit{opt}}(\Pi)$ in which only the ground operator PICK-UP(SP1, SHED) has been been removed, and all other (inapplicable) instantiations of PICK-UP are still present.[2] Then the structural symmetry swapping the spanners in Π is *not* a structural symmetry of $\mathit{ground}_{\mathit{opt}}(\Pi)$, because PICK-UP(SP2, SHED) has no symmetric counterpart.

However, this exploits that the grounding algorithm removes one unreachable operator but retains a symmetric one. This would be a very atypical behavior of a rational grounding algorithm. We say that a grounding algorithm is *rational* if it never removes a component (such as an operator or an atom) and at the same time keeps its symmetric component. We denote the resulting ground task by $\mathit{ground}_{\mathit{rat}}(\Pi)$. Note that in particular the induced ground representation corresponds to a trivial rational grounding.

Theorem 2. *If σ is a structural symmetry for a planning task Π, then it is a structural symmetry for $\mathit{ground}_{\mathit{rat}}(\Pi)$.*

Proof sketch. In Theorem 1, we have shown that every structural symmetry of Π is a structural symmetry of the induced ground representation. Rational grounding can omit some components that occur in the induced ground representation but it always either removes all or none of the symmetric components. Hence any structural symmetry must be preserved through rational grounding. □

We conclude that with any rational grounding approach, our symmetries correspond to symmetries of the grounded representation, and hence we can safely exploit symmetries of the lifted representation for any application.

We find this result more relevant in practice than Observation 1 because we are not aware of any non-hypothetical non-rational grounding algorithm. In particular, approaches based on a reachability analysis, such as the one used in Fast Downward (Helmert 2009), are naturally rational.

Symmetries of the lifted representation define mappings of predicates and objects of a planning task, and as such induce a mapping of ground atoms as used in (propositional) ground representations of planning tasks.[3] However, according to the above theorem, such grounding of lifted symmetries with rational grounding algorithms cannot result in finding more symmetries compared to directly computing structural symmetries of the ground representation, and hence such an application of our symmetries is fruitless.

Relation to Previous Notions of Symmetry

Fox and Long (1999) consider *object symmetries* from the lifted task representation, where objects are symmetric if they cannot be distinguished in the initial state and the goal, i. e. permuting them does not affect these parts of the task description. This requires that objects do not occur in operators. For tasks that mention objects in operators, Fox and

[2]While this might not necessarily be the result of a any existing implementation of a grounding algorithm, it could be the result of some correct optimized grounding algorithm.

[3]A further transformation of a symmetry into finite domain representation (FDR) (Helmert 2009) is not as straight-forward in general but it is trivial in the common case of a *rational* transformation, i. e. if the transformation treats symmetric ground atoms symmetrically when grouping ground atoms into FDR variables.

Long use a task reformulation that identifies these objects in the initial state with the help of additional unary predicates. Operators then require these predicates to be true instead of the specific objects. This prevents such objects to be symmetric with any other object.

The detected object symmetries are structural symmetries in the sense of our definition, where all non-objects symbols are mapped to themselves. However, due to the different treatment of operators, we can find additional such symmetries that the method by Fox and Long cannot detect.

Shleyfman et al. (2015) already introduced *structural symmetries* for STRIPS planning tasks. These symmetries are also structural symmetries in the sense of our definition, but representing planning tasks as different abstract structures. In the following, we denote this other representation the *propositional* task representation. The main difference is that the set of symbols consists of the ground atoms. Ground atoms are therefore not represented as tuples but as symbols.

The different symbol set already gives rise to symmetries that are not symmetries of our task representation: consider a task in propositional representation that has a symmetry σ' with $\sigma'(P(a)) = P(a)$ and $\sigma'(P(b)) = Q(b)$. In our abstract structure representation this task cannot have an analogous symmetry σ because $\tilde{\sigma}(\langle P, a\rangle) = \langle P, a\rangle$ implies $\sigma(P) = P$, so $\tilde{\sigma}(\langle P, b\rangle) = \langle P, \tilde{\sigma}(b)\rangle \neq \langle Q, b\rangle$.

Vice versa, for ground planning tasks, each structural symmetry σ of our task representation corresponds to a structural symmetry σ' of the propositional representation. The key idea of the proof is to define σ' as $\sigma'(P(c_1, \dots, c_n)) = \sigma(P)(\sigma(c_1), \dots, \sigma(c_n))$. A full proof requires a definition of task equivalence bridging the formalisms and an extension of Shleyfman et al.'s definition to axioms and conditional effects. As both are straight-forward but lengthy, we refrain from including them in this paper.

Together with Theorem 2, this observation emphasizes the lack of theoretical gain in grounding structural symmetries of the lifted representation. Instead, structural symmetries of the lifted representation can be utilized for applications that work on this lifted representation, and these symmetries are symmetries in the same sense as in previous work.

A second aspect where our symmetries are similar to those of Shleyfman et al. is that they are *transition graph symmetries*. Since Shleyfman et al. did not cover axioms and conditional effects, we discuss transition graph symmetries independently. A transition graph symmetry of a planning task is a goal-stable automorphism of the induced transition graph of the task, i. e. a mapping of derived states and operators, preserving transitions with costs as well as goal states.

Theorem 3. *Let* $\Pi = \langle \mathcal{O}, \mathcal{A}, s_0, s_\star\rangle$ *be a ground planning task over* S *and let* σ *be a structural symmetry for* Π. *Then* $\tilde{\sigma}$ *(viewed as a function on the states and operators) is a transition graph symmetry of* $\mathcal{T}_\Pi$.

Proof. We have to show that $\tilde{\sigma}$ preserves transitions and their cost as well as goal states of $\mathcal{T}_\Pi$. We begin with showing the former. Let $\langle [\![s]\!], o, [\![s[o]]\!]\rangle$ be a transition of $\mathcal{T}_\Pi$ where s is a fluent state and $o = \langle \emptyset, \mathit{pre}, \mathit{eff}, \mathit{cost}\rangle \in \mathcal{O}$ an operator such that $[\![s]\!]$ satisfies pre. Then $\tilde{\sigma}([\![s]\!])$ satisfies the precondition of $\tilde{\sigma}(o)$ and $\tilde{\sigma}(s)[\tilde{\sigma}(o)] = \tilde{\sigma}(s[o])$. The (stratified) evaluation of the axioms deriving $[\![s[o]]\!]$ from $s[o]$ directly translates to a symmetric axiom evaluation deriving $[\![\tilde{\sigma}(s[o])]\!]$ from $\tilde{\sigma}(s[o])$. So overall, also $\langle \tilde{\sigma}([\![s]\!]), \tilde{\sigma}(o), \tilde{\sigma}([\![s[o]]\!])\rangle$ is a transition of $\mathcal{T}_\Pi$. The other direction follows directly from the same argument and the fact that σ^{-1} is a structural symmetry for Π (because the set of symmetries of Π form a group, c. f. Lemma 1).

To show that costs are preserved, let F be the function term specifying the cost of operator o. Let s_0 contain a function assignment $\langle F, c\rangle$ for some numeric value c. Then all transitions induced by o have the same cost c. As $\sigma(c) = c$ and $\tilde{\sigma}(s_0) = s_0$, this implies that s_0 contains a function assignment $\langle \tilde{\sigma}(F), c\rangle$, so that all transitions induced by $\tilde{\sigma}(o)$ have the same cost c. As $\tilde{\sigma}(s_\star) = s_\star$ implies that $[\![s]\!]$ satisfies $s_\star$ iff $\tilde{\sigma}([\![s]\!])$ satisfies $s_\star$, $\tilde{\sigma}$ preserves the set of goal states. □

Computation

Pochter, Zohar, and Rosenschein (2011) already established that symmetries can be computed as automorphisms of the so-called problem description graph (PDG). Later on, Shleyfman et al. (2015) showed that PDG symmetries correspond to structural symmetries under the assumption that the task specification does not contain redundant propositions. In the following we introduce a suitable graph representation for general abstract structures.

Definition 7. *Let* A *be an abstract structure over* S. *The* abstract structure graph (ASG) ASG_A *is the colored digraph* $\langle N, E\rangle$, *that contains the following nodes and edges:*

- N *contains a node* A *for the abstract structure* A. *If* N *contains a node for* $A' = \{A_1, \dots, A_n\}$ *or* $A' = \langle A_1, \dots, A_n\rangle$, *it also contains the nodes for* $A_1, \dots, A_n$.
- *For every set (sub-)structure* $A' = \{A_1, \dots, A_n\}$ *there are edges* $A' \to A_i$ *for* $i \in \{1, \dots, n\}$.
- *For every tuple (sub-)structure* $A' = \langle A_1, \dots, A_n\rangle$, *the graph contains auxiliary nodes* $n_1^{A'}, \dots, n_n^{A'}$, *edge* $A' \to n_1^{A'}$ *and for* $1 < i \leq n$ *edges* $n_{i-1}^{A'} \to n_i^{A'}$. *For each component* A_i, *there is an edge* $n_i^{A'} \to A_i$.
- *For each node* A', *if* $A' \in S$ *then* $\mathit{color}(A') = t(A')$. *If* $A' = \{A_1, \dots, A_n\}$, *then* $\mathit{color}(A') = \mathit{set}$, *and if* $A' = \langle A_1, \dots, A_n\rangle$, *then* $\mathit{color}(A') = \mathit{tuple}$. *All other (auxiliary) nodes have color* aux.

Figure 3 shows an example of an abstract structure graph.

Theorem 4. *Let* A *be an abstract structure over* S. *Then every colored graph automorphism of* ASG_A *induces a structural symmetry of* A.

Proof sketch. Let α be a colored graph automorphism of ASG_A. Then $\alpha|_S$ is a symbol mapping over S because for all nodes $A' \in S$ we have $\mathit{color}(A') = t(A')$ and no other node has a type color.

We can show by induction over the structure of A that for every sub-structure $A' = \{A'_1, \dots, A'_n\}$ and $A'' = \{A''_1, \dots A''_n\}$ of A it holds that $\alpha(A') = \{\alpha(A'_1), \dots, \alpha(A'_n)\}$ and $\alpha(A'') = \langle \alpha(A''_1), \dots, \alpha(A''_n)\rangle$. Thus, α restricted to the non-auxiliary nodes is an abstract

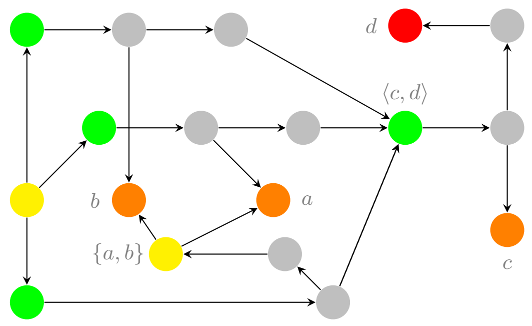

Figure 3: Abstract structure graph for $\{\langle a,\langle c,d\rangle\rangle, \langle b,\langle c,d\rangle\rangle, \langle\langle c,d\rangle,\{a,b\}\rangle\}$, where a,b,c have type t_1 and d type t_2. Set nodes are yellow, tuple nodes green, nodes from the set of symbols orange (t_1) and red (t_2), and auxiliary nodes gray. Labels next to some nodes are not part of the graph but should help the reader to match the components. Note that several occurrences of the same sub-structure (e. g. $\langle c,d\rangle$) are only represented once.

structure mapping for A with underlying symbol mapping $\alpha|_S$. As A is the only node with no incoming edges, it holds that $\alpha(A) = A$, so $\alpha|_S$ is a structural symmetry of A. □

An immediate consequence is that we can use any graph automorphism tool to compute structural symmetries of a planning task Π: construct ASG_Π and let the tool compute a set of automorphisms which generate a subgroup of the automorphism group $\Gamma(ASG_\Pi)$.[4] This subgroup corresponds to a symmetry group of Π. This procedure is complete in the sense that it can find all structural symmetries of A.

Theorem 5. *Let σ be a structural symmetry for abstract structure A. Then σ can be extended to a colored graph automorphism of the abstract structure graph ASG_A.*

Proof sketch. We can extend σ to a suitable permutation α of the nodes of ASG_A as follows: on the non-auxiliary nodes, α is identical to $\tilde{\sigma}$. For the auxiliary nodes, it maps the node n'_i that was introduced for the i-th component of a tuple A' to the auxiliary node that was introduced for the i-th component of tuple $\tilde{\sigma}(A')$. Then α is a permutation of the nodes of ASG_A that only maps nodes to nodes of the same color. For showing that there is an edge $n \to n'$ iff there is an edge $\alpha(n) \to \alpha(n')$, the full proof makes a case distinction for the different cases where edges are introduced in Definition 7. For example, for every edge $A' \to A'_i$ that was introduced for set $A' = \{A'_1,\dots,A'_n\}$, there is edge $\alpha(A') \to \alpha(A'_i)$ because $\alpha(A') = \{\alpha(A'_1),\dots,\alpha(A'_n)\}$ is again a set, inducing the required edge in ASG_A. □

Quantitative Analysis of Lifted Symmetries

Previous work established that propositional structural symmetries arise across nearly all common STRIPS planning

[4]While no polynomial-time algorithms are known for computing the set of generators of the automorphism group of a graph, graph automorphism tools can efficiently compute the generators of a subgroup thereof even for large graphs.

regular representation: FDR vs lifted					lifted: regular vs bagged		
total	FDR	lifted	+	-	total	regular	bagged
2518	1807	1352	70 (6)	455 (19)	1764	918	782

Table 1: Left block: number of tasks (total), out of which with symmetries (FDR/lifted), and number of tasks (domains) with lifted but no FDR symmetries (+) and vice versa (-). Right block: analogously number of tasks with and without lifted symmetries on regular and bagged benchmarks.

benchmarks (Domshlak, Katz, and Shleyfman 2013; Sievers et al. 2015a). As we saw that we might find fewer structural symmetries on the lifted representation, we report quantitative results on all benchmarks (including those with axioms and conditional effects) from the sequential tracks of all IPCs up to 2014 (including identical domains used in different IPCs only once), consisting of 77 domains with a total of 2518 tasks. We implemented the ASG in the translator component of Fast Downward (Helmert 2006) and used the graph automorphism tool Bliss (Junttila and Kaski 2007) to compute generators of symmetry groups. Furthermore, even though our intended application of lifted symmetries is *not* for any applications on ground tasks, for completeness, we also include a comparison to propositional structural symmetries in our analysis. To compute these, we use the PDG described by Shleyfman et al. (2015) (extended to axioms and conditional effects), implemented for the FDR representation used in Fast Downward. To allow a fair comparison, we not only stabilize the goal, but also the initial state, and additionally include symmetries that only act on operators (which is not useful respectively necessary for the application of symmetry-based pruning as in previous work). To further relate our approach to the task transformation into the so-called bagged representation (Riddle et al. 2016), which aims at eliminating symmetries, we also compute lifted symmetries on these reformulated tasks. Each task is run on Intel Xeon E5-2660 CPUs with 2.2 GHz, using a memory limit of 3.5 GiB and a time limit of 300 seconds.[5]

Table 1 shows aggregated quantitative results. First consider the comparison of computing FDR and lifted symmetries on regular IPC benchmarks, shown in the left block. We observe that in total, more than half of the tasks (1352/2518) exhibit lifted symmetries, distributed across 63 out of 77 domains, compared to 1807 tasks with FDR symmetries, distributed across the same 63 domains plus 3 additional domains. While this already establishes that there are lots of lifted symmetries, we further observed that in 6 domains, there are *more* tasks with lifted symmetries than with FDR symmetries (the converse is true in 19 domains). A closer examination of this surprising fact revealed that these lifted symmetries are irrelevant in the reachable state space and hence get removed during the grounding process so that previous notions of structural symmetries could not capture them. For example, in ASSEMBLY, the normalization of

[5]Implementation: https://doi.org/10.5281/zenodo.2621897, dataset with benchmarks: https://doi.org/10.5281/zenodo.2621424.

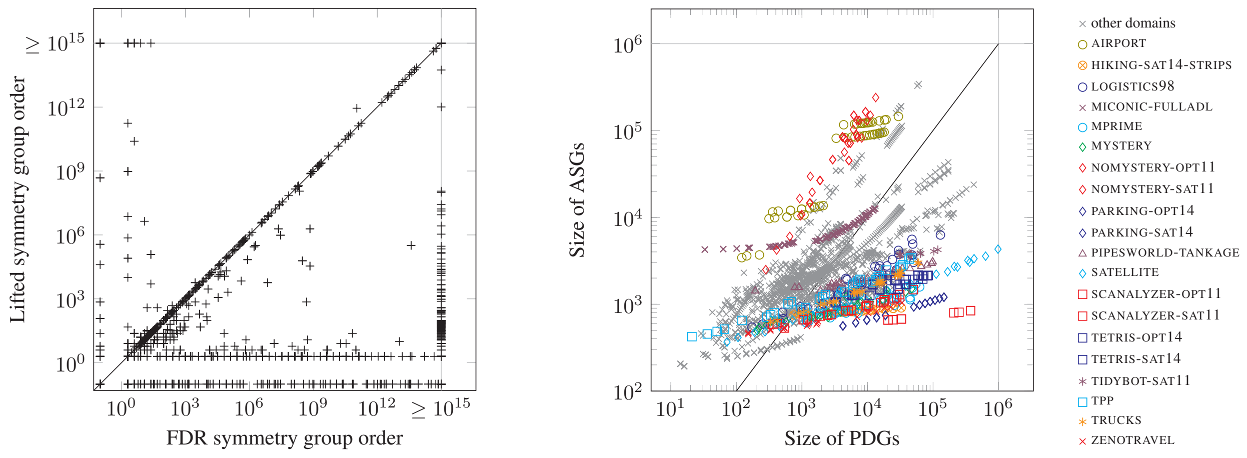

Figure 4: Left: order of lifted vs FDR symmetry groups. Right: number of nodes in ASGs vs PDGs.

PDDL in the translator introduces symmetric axioms, which we detect on the lifted representation, but which get removed again when grounding. The overall much lower number of tasks with lifted symmetries mainly originates from some domains without lifted symmetries containing many more tasks than other domains (e.g., MICONIC, SCHEDULE). For the large majority of domains where both FDR and lifted symmetries arise, the number of tasks with symmetries is the same or similar. Besides the comparison to FDR symmetries, we also analyzed the type of lifted symmetries found in each domain: there are 5 domains with tasks that have only symmetries mapping predicates, 54 domains have symmetries only mapping objects, and the remaining 4 domains contain mixed symmetries.

To further quantify the differences of lifted and FDR structural symmetries, we do not report statistics on the group generators as it was done in previous work, but instead we compute the *order* (= cardinality) of the resulting group using the Python library SymPy[6] with the same memory and time limits, as suggested recently by Shleyfman (2018). Although the equivalence in group order is not a guarantee of group isomorphism (or vice versa), it is indeed a better indicator than the size or the order of a generating set. The left plot in Figure 4 shows the comparison of group orders. Note that we clamp values larger than 10^{15} to 10^{15} for readability. As expected, for many of the domains where there are both lifted and FDR symmetries, the order of FDR symmetry groups is larger than that of lifted groups.

We also analyzed the graphs used to compute FDR and lifted symmetries. The right plot in Figure 4 compares the number of nodes of ASGs and PDGs, highlighting domains with tasks where these numbers are at least a factor 20 apart. For some domains that specify full ground tasks in PDDL, ASGs are larger than PDGs by a constant factor. Furthermore, for small planning tasks, the overhead of having nodes for parameters of actions and literals in the ASGs results in mildly larger graphs compared to PDGs. For most other cases, however, the plot indicates that the size of ASGs grows significantly slower than that of PDGs. An analysis of the runtime to compute symmetries confirms this result: computing lifted symmetries, there are only 25 tasks for which the computation takes *more* than 2s. In these tasks, the maximum computation time is 8.23s. In contrast, computing FDR symmetries takes more than 2s in 53 tasks and up to 224.113s.

Finally, consider now the right block of Table 1. It compares computing lifted symmetries on regular and bagged domains, restricting the comparison to tasks where the reformulation was possible (1764 tasks across 60 domains). Even though the bagged representation aims at eliminating symmetries, of the 918 tasks across 51 domains with lifted symmetries in the original representation, 782 tasks across 45 domains still exhibit lifted structural symmetries after the reformulation. Furthermore, we also analyzed the orders of the symmetries and found them to be identical in 38 of the 45 domains. This means that many structural symmetries are preserved throughout the reformulation into a bagged representation, thus showing that they capture a stronger concept of symmetry.

Discussion and Future Work

The exploitation of symmetries is not only interesting in the context of planning but for many areas working on some form of transition system or model-finding. For example, there is a large body of work on symmetries available in the model checking community (Clarke et al. 1996, tutorial-style survey by Wahl and Donaldson 2010), which already introduced most concepts later adopted by the planning community. Symmetries get likewise exploited in SMT (Déharbe et al. 2011), propositional satisfiability (Crawford et al. 1996) or SAT representations of transition systems (Rintanen 2003). Other areas also aready investigated structural symmetries on lifted representations and how they carry over to the state space (e. g. Starke 1991 for petri nets). We did an analgous analysis for the semantics of planning.

Generally, one is interested in symmetries leading to semantically equivalent theories or tasks, where symmetries

[6]www.sympy.org

are often defined as permutations of symbols of the underlying language. As semantic equivalence is typically too expensive to decide, it is common to use syntactical equivalence instead. For instance, Crawford et al. consider syntactical equivalence in a normal form and Déharbe et al. use rewriting, which can for example reorder parts in a logic conjunction. We also use *syntactical* equivalence, where the abstract structures serve as an alternative to such normal forms, e. g. by representing parts of a planning task for which the order does not matter as sets.

Crawford et al. also relate symmetries to graph isomorphisms. This is nowadays the dominant approach for symmetry detection (Emerson and Sistla 1996; Donaldson, Miller, and Calder 2005; Pochter, Zohar, and Rosenschein 2011) but requires to define a new graph for each new application. We contribute the graph for abstract structures.

Independent from symmetries, this graph representation has already proven to be useful within Delfi1, the planning system that won the classical optimal track of the IPC 2018 (Katz et al. 2018). Delfi1 is an online portfolio, selecting a planner for a given task using a convolutional neural network. For the representation of the task, Delfi1 uses a fixed-sized image constructed from the ASGs defined in this work.

While it is quite obvious that structural symmetries computed on the lifted representation of planning tasks carry over to the induced ground task, we pointed out that one needs to be careful when applying *optimized* grounding, which can potentially eliminate symmetries. However, with *rational* grounding, symmetries of the lifted representation are guaranteed to be symmetries of the grounded task. Furthermore, we established that with such grounding, each lifted structural symmetry is a transition graph symmetry of the grounded task and can thus be exploited the same way, e. g. for symmetry breaking in forward search (Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012) or orbit space search (Domshlak, Katz, and Shleyfman 2015). However, we saw that – already due to representational limitations – this approach would find fewer symmetries than the approach by Shleyfman et al. (2015) for finding structural symmetries of STRIPS representations.

While these theoretical results are important to clarify the relation of our work to previous notions of symmetry, we see the potential for practical applications of lifted structural symmetries in areas where the earlier notions are inapplicable, namely applications that operate directly on the lifted representation or at the transition point between the lifted and the ground task representation.

One such application is the generation of invariants, which are used for strengthening other techniques, e. g. in constrained PDBs (Haslum, Bonet, and Geffner 2005), dead-end detection (Lipovetzky, Muise, and Geffner 2016) or for computing upper bounds on plan length (Rintanen and Gretton 2013). Invariants are also crucial for the transformation of the task into finite-domain representation (Helmert 2009), which many planning heuristics rely on (Edelkamp 2001; Helmert et al. 2014; Seipp and Helmert 2013; Helmert 2006). Traditionally, invariant generation methods fall into two groups: those that operate only on the ground representation (Blum and Furst 1997; Rintanen 1998; 2008) and those that work directly on the lifted representation (Gerevini and Schubert 1998; Edelkamp and Helmert 1999; Rintanen 2000; Lin 2004; Helmert 2009). The latter usually scale better with the size of the task but require a certain amount of first-order reasoning that suffers from complicated operator specifications.

Recent work on invariants (Li, Fan, and Liu 2013; Rintanen 2017) shows that it is often feasible to only consider a subset of objects for the verification of lifted invariants. An extension of this line of work already exploits structural symmetries as described in this paper, using a symmetry-based task reduction for speeding up the relaxed reachability analysis for grounding and mutex computation (Röger, Sievers, and Katz 2018). The analysis is performed on a smaller task and the full (non-reduced) result gets recovered in a subsequent expansion. While Rintanen considers regression and Röger, Sievers, and Katz a forward analysis, they have identified similar bounds for the number of objects that need to be preserved. We therefore expect that this work can be generalized to cover further invariant synthesis methods.

Our structural symmetries could also be applicable in *lifted* planning. For example, Ridder (2013) uses symmetry breaking of object symmetries in a lifted relaxed planning graph heuristic, also considering so-called *almost symmetry*. It would be interesting to compare the approaches to analyse whether his or our concepts could be further generalized.

Another interesting direction is task reformulation. It can be beneficial to reformulate a planning task so that for some objects only the number of objects with specific properties is represented but not the exact objects (Riddle et al. 2016; Fuentetaja and de la Rosa 2016). This reformulation is related to symmetry-based counter abstraction (Emerson and Wahl 2003) used in model checking, and many of the criteria Riddle et al. use for detecting suitable objects are naturally subsumed by our symmetries. We therefore expect that we can exploit structural symmetries to apply similar state space transformations to a wider range of planning domains.

In this paper, we laid the theoretical foundation for a sound exploitation of symmetries in all above existing and potential applications. Our experiments show that a large number of planning benchmarks exhibits structural symmetries in the lifted representation, so a further investigation of this line of research appears indeed promising.

Acknowledgments

We have received funding for this work from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 817639).

References

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

Blum, A., and Furst, M. L. 1997. Fast planning through planning graph analysis. *AIJ* 90(1–2):281–300.

Clarke, E. M.; Jha, S.; Enders, R.; and Filkorn, T. 1996. Exploiting symmetry in temporal logic model checking. *Formal Methods in System Design* 9(1/2):77–104.

Crawford, J.; Ginsberg, M.; Luks, E.; and Roy, A. 1996. Symmetry-breaking predicates for search problems. In *Proc. KR 1996*, 148–159.

Déharbe, D.; Fontaine, P.; Merz, S.; and Woltzenlogel Paleo, B. 2011. Exploiting symmetry in SMT problems. In *23rd Intl. Conf. Automated Deduction (CADE 2011)*, volume 6803 of *LNCS*, 222–236. Springer.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced symmetry breaking in cost-optimal planning as forward search. In *Proc. ICAPS 2012*, 343–347.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2013. Symmetry breaking: Satisficing planning and landmark heuristics. In *Proc. ICAPS 2013*, 298–302.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2015. Symmetry breaking in deterministic planning as forward search: Orbit space search algorithm. Technical Report IS/IE-2015-03, Technion.

Donaldson, A. F.; Miller, A.; and Calder, M. 2005. Finding symmetry in models of concurrent systems by static channel diagram analysis. In *Proc. AVoCS 2004*, 161–177.

Edelkamp, S., and Helmert, M. 1999. Exhibiting knowledge in planning problems to minimize state encoding length. In *Proc. ECP 1999*, 135–147.

Edelkamp, S. 2001. Planning with pattern databases. In *Proc. ECP 2001*, 84–90.

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

Emerson, E. A., and Wahl, T. 2003. On combining symmetry reduction and symbolic representation for efficient model checking. In *Advanced Research Working Conference on Correct Hardware Design and Verification Methods (Charme 2003)*, volume 2860 of *LNCS*, 216–230. Springer.

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

Fox, M., and Long, D. 2002. Extending the exploitation of symmetries in planning. In *Proc. AIPS 2002*, 83–91.

Fuentetaja, R., and de la Rosa, T. 2016. Compiling irrelevant objects to counters. special case of creation planning. *AI Communications* 29(3):435–467.

Gerevini, A. E., and Schubert, L. 1998. Inferring state constraints for domain-independent planning. In *Proc. AAAI 1998*, 905–912.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *Proc. AAAI 2005*, 1163–1168.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *JACM* 61(3):16:1–63.

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

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *AIJ* 173:503–535.

Huber, P.; Jensen, A. M.; Jepsen, L. O.; and Jensen, K. 1986. Reachability trees for high-level petri nets. *Theoretical Computer Science* 45:261 – 292.

Ip, C. N., and Dill, D. L. 1996. Better verification through symmetry. *Formal Methods in System Design* 9(1–2):41–75.

Junttila, T., and Kaski, P. 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proc. ALENEX 2007*, 135–149.

Katz, M.; Sohrabi, S.; Samulowitz, H.; and Sievers, S. 2018. Delfi: Online planner selection for cost-optimal planning. In *IPC-9 planner abstracts*, 57–64.

Köhler, J., and Hoffmann, J. 2000. On the instantiation of ADL operators involving arbitrary first-order formulas. In *ECAI 2000 PuK Workshop*, 74–82.

Li, N.; Fan, Y.; and Liu, Y. 2013. Reasoning about state constraints in the situation calculus. In *Proc. IJCAI 2013*, 997–1003.

Lin, F. 2004. Discovering state invariants. In *Proc. KR 2004*, 536–544.

Lipovetzky, N.; Muise, C.; and Geffner, H. 2016. Traps, invariants, and dead-ends. In *Proc. ICAPS 2016*, 211–215.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting problem symmetries in state-based planners. In *Proc. AAAI 2011*, 1004–1009.

Ridder, B. 2013. *Lifted Heuristics: Towards More Scalable Planning Systems*. Ph.D. Dissertation, King's College London.

Riddle, P.; Douglas, J.; Barley, M.; and Franco, S. 2016. Improving performance by reformulating PDDL into a bagged representation. In *ICAPS 2016 Workshop on Heuristics and Search for Domain-independent Planning*, 28–36.

Rintanen, J., and Gretton, C. O. 2013. Computing upper bounds on lengths of transition sequences. In *Proc. IJCAI 2013*, 2365–2372.

Rintanen, J. 1998. A planning algorithm not based on directional search. In *Proc. KR 1998*, 617–624.

Rintanen, J. 2000. An iterative algorithm for synthesizing invariants. In *Proc. AAAI 2000*, 806–811.

Rintanen, J. 2003. Symmetry reduction for SAT representations of transition systems. In *Proc. ICAPS 2003*, 32–40.

Rintanen, J. 2008. Regression for classical and nondeterministic planning. In *Proc. ECAI 2008*, 568–572.

Rintanen, J. 2017. Schematic invariants by reduction to ground invariants. In *Proc. AAAI 2017*, 3644–3650.

Röger, G.; Sievers, S.; and Katz, M. 2018. Symmetry-based task reduction for relaxed reachability analysis. In *Proc. ICAPS 2018*, 208–217.

Seipp, J., and Helmert, M. 2013. Counterexample-guided Cartesian abstraction refinement. In *Proc. ICAPS 2013*, 347–351.

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

Shleyfman, A. 2018. On computational complexity of automorphism groups in classical planning. In *ICAPS 2018 Workshop on Heuristics and Search for Domain-independent Planning*, 66–72.

Sievers, S.; Wehrle, M.; Helmert, M.; and Katz, M. 2015a. An empirical case study on symmetry handling in cost-optimal planning as heuristic search. In *Proc. KI 2015*, 151–165.

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

Starke, P. H. 1991. Reachability analysis of petri nets using symmetries. *Systems Analysis Modelling Simulation* 8(4–5):293–303.

Thiébaux, S.; Hoffmann, J.; and Nebel, B. 2005. In defense of PDDL axioms. *AIJ* 168(1–2):38–69.

Wahl, T., and Donaldson, A. 2010. Replication and abstraction: Symmetry in automated formal verification. *Symmetry* 2:799–847.

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