Structural Symmetries of the Lifted Representation of Classical Planning Tasks

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

Michael Katz
IBM Watson Health, Haifa, Israel
katzm@il.ibm.com

Abstract

We transfer the notion of structural symmetries to lifted planning task representations, based on a generalizing concept of abstract structures we use to model planning tasks. We show that symmetries are preserved by common grounding methods and shed some light on the relation to previous symmetry concepts. An analysis of common planning benchmarks reveals that symmetries occur in the lifted representation of many domains. Our concept prepares the ground for exploiting symmetries beyond their current scope, such as for faster grounding and mutex generation, as well as for state space transformations and state space reductions.

Introduction

In the last decade, the concept of symmetries has been increasingly investigated for the development of techniques to increase the scalability of domain-independent classical planners (Fox and Long 1999a; 1999b; Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012; 2015; Sievers et al. 2015b; Riddle et al. 2016). In particular, Shleyfman et al. (2015) introduced the notion of *structural symmetries*, which is a declarative symmetry definition on the representation of propositional STRIPS tasks. These symmetries subsume several earlier symmetry definitions for classical planning, many of which focus on *object* symmetries (Fox and Long 1999a; 1999b; Riddle et al. 2016). Further, structural symmetries generalize other types of symmetries considered for other state-space search problems in general, e. g. *rotation* and *reflection*.

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 define structural symmetries of the lifted representation. Our aim is to build the theoretical basis for many promising future applications of such symmetries. While structural symmetries of the lifted representation could be grounded to be used for existing symmetry-based approaches that operate on the ground representation, this is not our intended application, and there is no theoretical gain in doing so, as we will show. Rather, structural symmetries of the lifted representation can be used for all purposes operating on lifted representations. In particular, in the long term, we would like to examine the potential of symmetries for faster grounding and mutex generation, as well as for state space transformations and state space reductions.

For this purpose, we transfer the definition of structural symmetries of Shleyfman et al. (2015) to lifted planning tasks. We model planning tasks based on a general concept of *abstract structures*, unlike previous work also covering axioms and conditional effects. We show that symmetries are preserved by common grounding methods and how they are related to previous symmetry concepts. We also describe how symmetries of the lifted representation can be computed and show that planning benchmarks from the International Planning Competition exhibit a large number of such symmetries. We close with a discussion of possible future applications.

Structural Symmetries

We start with defining abstract structures and structural symmetries for these structures.

Definition 1 (Abstract structure)**.** *Let S be a set of symbols, where each $s \in S$ is associated with a* 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 $A_1, \ldots, A_n$, the set $\{A_1, \ldots, A_n\}$ and the tuple $\langle A_1, \ldots, A_n \rangle$ are abstract structures.*

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. Formally:

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

Definition 3 (Structural symmetry)**.** *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:*

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

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

We establish that the set of all structural symmetries for an abstract structure A forms a group. 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, 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 $\tilde{\sigma}(A) = \tilde{\sigma}_1(\tilde{\sigma}_2(A)) = \tilde{\sigma}_1(A) = A$. □

Planning Tasks as Abstract Structures

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

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

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

We call these structures *ground* if they do not contain $Variable$ symbols.

Definition 5 (Planning task)**.** *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 $o = \langle params, pre, \mathit{eff}, cost \rangle$ where*
  - *$params$ is a set of $Variable$ 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 $\langle vars, cond, lit \rangle$, where $vars$ is a set of $Variable$ symbols, $cond$ is a set of literals where all occurring variables are from $params \cup vars$, and lit is a literal of a fluent atom where all variables are from $params \cup vars$, and*
  - *$cost$ is a function term where all occurring variables are from $params$;*
- *$\mathcal{A}$ is a set of axioms, each of the form $a = \langle params, pre, \mathit{eff} \rangle$ where*
  - *$params$ is a set of $Variable$ symbols,*
  - *pre is a set of literals where all occurring variables are from $params$, and*
  - *eff is a derived atom where all occurring variables are from $params$,*

  and this 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. we require that all occurring sets of $Variable$ symbols are disjoint.

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

For a set S of symbols for planning, we define $Objs(S) = \{s \in S \mid t(s) = Object\}$. For sets X and Y, we denote the set of all functions $f : X \to Y$ by X^Y. We call functions m mapping from the $Variable$ symbols in S to $Objs(S)$ *variable mappings*. We 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 (Induced ground planning task)**.** *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 $ground(\Pi) = \langle ground(\mathcal{O}), ground(\mathcal{A}), s_0, s_\star \rangle$ over S with*

- *$ground(\mathcal{O}) = \bigcup_{o \in \mathcal{O}} opground(o)$, where*

$$opground(\langle params, pre, \mathit{eff}, cost \rangle) = \{\langle \emptyset, \tilde{m}(pre), \tilde{m}(expand(\mathit{eff})), \tilde{m}(cost) \rangle \mid m \in params^{Objs(S)}\}, \text{ with}$$

$$expand(\mathit{eff}) = \{\langle \emptyset, \tilde{n}(cond), \tilde{n}(lit) \rangle \mid \langle vars, cond, lit \rangle \in \mathit{eff}, n \in vars^{Objs(S)}\}$$

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

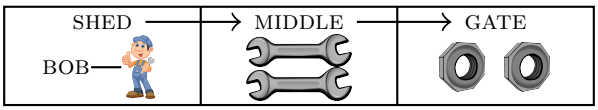

Figure 1: Exemplary initial state of a SPANNER task.

- $ground(\mathcal{A}) = \bigcup_{a \in \mathcal{A}} axground(a)$, *where*

$$axground(\langle params, pre, \mathit{eff} \rangle) = \{\langle \emptyset, \tilde{m}(pre), \tilde{m}(\mathit{eff}) \rangle \mid m \in params^{Objs(S)}\}.$$

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 and no negated atom from C occurs in s. A ground operator $o = \langle \emptyset, pre, \mathit{eff}, cost \rangle$ is *applicable* in a fluent state s if $[\![s]\!]$ satisfies pre and s_0 contains a function assignment for $cost$. The (fluent) successor state $s[o]$ contains a fluent ground atom a if there is an effect $\langle \emptyset, cond, a \rangle \in \mathit{eff}$ such that $[\![s]\!]$ satisfies $cond$ or if $a \in s$ and there is no $\langle \emptyset, cond, \langle \neg, a \rangle \rangle \in \mathit{eff}$ where $[\![s]\!]$ satisfies $cond$. The (fluent) *initial state* consists of the atoms in s_0. A *plan* for Π is a sequence of operators such that their subsequent application to the initial state leads to a state s' so that $[\![s']\!]$ satisfies $s_\star$. Its *cost* is the accumulated operator costs of the sequence, where the actual numeric values are taken from the function assignments in the initial state. *Satisficing* planning deals with finding plans of any cost whereas *optimal* planning is only interested in plans with minimal cost among all plans.

The semantics of Π can also 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 $o = \langle \emptyset, pre, \mathit{eff}, cost \rangle \in \mathcal{O}$ 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. The set of goal states G consists of all $s \in D$ that satisfy $s_\star$. Then a plan for Π corresponds to the sequence of labels along a path in $\mathcal{T}_\Pi$ from $[\![s[o]]\!]$ to a state from G. For a lifted planning task, its transition graph is defined as the transition graph of the induced ground task.

As an example, consider a planning task of the IPC domain SPANNER with the initial state shown in Figure 1. 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. There are operators MOVE(X, Y) to 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 planning 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.

We remark that due to our definition of planning tasks as abstract structures and because structural symmetries require to permute the entire abstract structure, our symmetries *stabilize* both the initial state and the goal condition, i. e. no parts of a planning task can be considered symmetric if they are not symmetric in the initial state or the goal. This differs to the definition of structural symmetries of ground representation in previous work; e. g. Shleyfman et al. (2015) do not stabilize the initial state because for symmetry-based pruning in a forward search, only plans to the goal must be preserved under a structural symmetry, but not the initial state. Even if we were interested in this kind of application, not stabilizing the initial state causes some difficulties due to the specification of PDDL: function assignments and all "static" information (e.g. hard-coded 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 of the planning task, for which stabilizing the initial state is essential. For these applications, we do not need to stabilize the goal, which we can achieve by simply dropping the goal from the abstract structure of a planning task.

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 ground *induced* representation, and then discuss 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 $ground(\Pi)$.*

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

To establish that $\mathit{ground}(\mathcal{O}) = \tilde{\sigma}(\mathit{ground}(\mathcal{O}))$, let $o_\downarrow = \langle \emptyset, \mathit{pre}_\downarrow, \mathit{eff}_\downarrow, \mathit{cost}_\downarrow \rangle \in \mathit{ground}(\mathcal{O})$. Since $o_\downarrow$ is in $\mathit{ground}(\mathcal{O})$ there is an operator $o = \langle \mathit{params}, \mathit{pre}, \mathit{eff}, \mathit{cost} \rangle \in \mathcal{O}$ and a variable mapping m such that $o_\downarrow \in \mathit{opground}(o)$. Since σ is a structural symmetry of $\mathcal{O}$, also $\tilde{\sigma}(o) \in \mathcal{O}$. Consider again $m' := \sigma \circ m \circ \sigma^{-1} \in \tilde{\sigma}(\mathit{params})^{\mathit{Objs}(S)}$. Variable mapping m' grounds $\tilde{\sigma}(o)$ to $o' := \langle \emptyset, \tilde{m}'(\tilde{\sigma}(\mathit{pre})), \tilde{m}'(\mathit{expand}(\tilde{\sigma}(\mathit{eff}))), \tilde{m}'(\tilde{\sigma}(\mathit{cost})) \rangle \in \mathit{opground}(\tilde{\sigma}(a))$, where $\mathit{expand}(\tilde{\sigma}(\mathit{eff})) = \{\langle \emptyset, \tilde{n}(\tilde{\sigma}(\mathit{cond})), \tilde{n}(\tilde{\sigma}(\mathit{lit})) \rangle \mid \langle \tilde{\sigma}(\mathit{vars}), \tilde{\sigma}(\mathit{cond}), \tilde{\sigma}(\mathit{lit}) \rangle \in \tilde{\sigma}(\mathit{eff}), n \in \tilde{\sigma}(\mathit{vars})^{\mathit{Objs}(S)}\}$. We get

$$
\begin{aligned}
&\tilde{m}'(\mathit{expand}(\tilde{\sigma}(\mathit{eff}))) \\
&\quad = \{\langle \emptyset, \tilde{m}'(\tilde{n}(\tilde{\sigma}(\mathit{cond}))), \tilde{m}'(\tilde{n}(\tilde{\sigma}(\mathit{lit}))) \rangle \mid \\
&\qquad \langle \tilde{\sigma}(\mathit{vars}), \tilde{\sigma}(\mathit{cond}), \tilde{\sigma}(\mathit{lit}) \rangle \in \tilde{\sigma}(\mathit{eff}), \\
&\qquad n \in \tilde{\sigma}(\mathit{vars})^{\mathit{Objs}(S)}\} \\
&\quad = \{\langle \emptyset, \tilde{n}(\tilde{m}'(\tilde{\sigma}(\mathit{cond}))), \tilde{n}(\tilde{m}'(\tilde{\sigma}(\mathit{lit}))) \rangle \mid \\
&\qquad \langle \tilde{m}'(\tilde{\sigma}(\mathit{vars})), \tilde{m}'(\tilde{\sigma}(\mathit{cond})), \tilde{m}'(\tilde{\sigma}(\mathit{lit})) \rangle \\
&\qquad \in \tilde{m}'(\tilde{\sigma}(\mathit{eff})), n \in \tilde{m}'(\tilde{\sigma}(\mathit{vars}))^{\mathit{Objs}(S)}\} \\
&\quad = \{\langle \emptyset, \tilde{n}(\tilde{\sigma}(\mathit{cond}_\downarrow)), \tilde{n}(\tilde{\sigma}(\mathit{lit}_\downarrow)) \rangle \mid \\
&\qquad \langle \tilde{\sigma}(\mathit{vars}_\downarrow), \tilde{\sigma}(\mathit{cond}_\downarrow), \tilde{\sigma}(\mathit{lit}_\downarrow) \rangle \in \tilde{\sigma}(\mathit{eff}_\downarrow), \\
&\qquad n \in \tilde{\sigma}(\mathit{vars}_\downarrow)^{\mathit{Objs}(S)}\} \\
&\quad = \mathit{expand}(\tilde{\sigma}(\mathit{eff}_\downarrow))
\end{aligned}
$$

where the first step (switching m' and n) is possible because $\tilde{\sigma}(\mathit{params}) \cap \tilde{\sigma}(\mathit{vars}) = \emptyset$, and the second step uses the definition of m' and the same argumentation as for axioms, cf. (*). With the latter, we also get that $\tilde{m}'(\tilde{\sigma}(\mathit{pre})) = \tilde{\sigma}(\mathit{pre}_\downarrow)$. Furthermore, we have that $\tilde{m}'(\tilde{\sigma}(\mathit{cost})) = \tilde{m}' \circ \tilde{\sigma}(\mathit{cost}) = \tilde{\sigma} \circ \tilde{m}(\mathit{cost}) = \tilde{\sigma}(\mathit{cost}_\downarrow)$, and so overall $o' = \tilde{\sigma}(o_\downarrow)$. Therefore, for each $o \in \mathit{ground}(\mathcal{O})$, also $\tilde{\sigma}(o)$ is in $\mathit{ground}(\mathcal{O})$, and since $\tilde{\sigma}$ is a permutation on Π, $\tilde{\sigma}(\mathit{ground}(\mathcal{O})) = \mathit{ground}(\mathcal{O})$.

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. For example, the induced ground representation of task #28 of the IPC domain LOGISTICS98 contains $5.82 \cdot 10^{10}$ operators, compared to $3 \cdot 10^6$ operators in a ground representation where operators that are inapplicable due to mismatching types of parameters or statically unsatisfiable preconditions are removed (Helmert 2009).

We say that a grounding algorithm is *optimized* if it removes (some, not necessarily all) irrelevant parts of the task representation (Köhler and Hoffmann 2000). Such grounding is *correct* if the reachable part of the transition graph is not affected. We denote ground representations of lifted tasks Π that result from correct optimized grounding by $\mathit{ground}_{\mathit{opt}}(\Pi)$.

Observation 1. *Let Π be a planning task and let σ be a structural symmetry for Π. Then σ is not necessarily a structural symmetry for $\mathit{ground}_{\mathit{opt}}(\Pi)$.*

As an example for this observation, consider again the planning task of the IPC domain SPANNER shown in Figure 1. 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 mapping 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 reasonable 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 a symmetric component. We denote the resulting ground representation by $\mathit{ground}_{\mathit{rat}}(\Pi)$.

Theorem 2. *Let Π be a planning task and let σ be a structural symmetry for Π. Then σ 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. As any structural symmetry maps the initial state onto itself, we can easily show that any symmetric state of a reachable state is reachable, and hence any symmetric operator of a reachable operator is reachable. Thus, no structural symmetry can map a non-reachable operator to a reachable operator of the planning task or vice versa. Hence, as rational grounding either removes all or none of the symmetric components, any structural symmetry must be preserved through rational grounding. □

We conclude that with any reasonable 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 remark that 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

[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.

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

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, we can show that 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 $\sigma'(P(c_1, \ldots, c_n)) = \sigma(P)(\sigma(c_1), \ldots, \sigma(c_n))$. 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 the result of Theorem 2 in the previous section, this observation again emphasizes that there is no theoretical gain in computing structural symmetries of the lifted representation for the purpose of grounding them. However, we can utilize structural symmetries of the lifted representation for any application that works on this lifted representation, and these structural 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 so-called transition graph symmetries, as we will show next. Since, as mentioned above, 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 and their cost 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 a certain graphical structure. In the following we introduce a suitable graph representation for general abstract structures.

Definition 7 (Abstract structure graph)**.** *Let A be an abstract structure over S. The* abstract structure graph *ASG_A is a colored digraph $\langle N, E\rangle$, defined as follows.*

- *N contains a node A for the abstract structure A. If N contains a node for $A' = \{A_1, \ldots, A_n\}$ or $A' = \langle A_1, \ldots, A_n\rangle$, it also contains the nodes for $A_1, \ldots, A_n$.*
- *For each node A', if $A' \in S$ then $\mathit{color}(A') = t(A')$. If $A' = \{A_1, \ldots, A_n\}$, then $\mathit{color}(A') = \mathit{set}$, and if $A' = \langle A_1, \ldots, A_n\rangle$, then $\mathit{color}(A') = \mathit{tuple}$.*
- *For each node $A_0 \in N$, E contains the following edges. If $A_0 = \{A_1, \ldots, A_n\}$, there are edges $\langle A_0, A_i\rangle \in E$ for $1 \leq i \leq n$. If $A_0 = \langle A_1, \ldots, A_n\rangle$, there are edges $\langle A_{i-1}, A_i\rangle \in E$ for $1 \leq i \leq n$.*

Theorem 4. *Let A be an abstract structure over S. Then a colored graph automorphism of ASG_A, interpreted as a mapping of abstract structures corresponding to nodes of ASG_A, is an abstract structure mapping of A such that its underlying symbol mapping is a structural symmetry.*

Proof sketch. Let $\tilde{\sigma}$ be a colored graph automorphism of ASG_A. Consider some node A' of ASG_A. If $A' \in S$, then also $\tilde{\sigma}(A') \in S$, because $\mathit{color}(A') = t(A')$. We immediately get that σ is a permutation on S. If $A' = \{A_1, \ldots A_n\}$, then from stabilizing colors and the structure-preserving property of automorphisms, we get $\tilde{\sigma}(A') = \{\tilde{\sigma}(A_1), \ldots, \tilde{\sigma}(A_n)\}$. Analogously if $A' = \langle \ldots \rangle$. Finally, note that A is the only node with no incoming edges, and hence $\tilde{\sigma}(A) = A$. □

An immediate consequence is that we can use any graph automorphism tool to compute structural symmetries of a planning task Π: construct the abstract structure graph 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.

[4]While no polynomial-time algorithms are known for computing the set of generators of the automorphism group of a graph,

Quantitative Analysis of Lifted Symmetries

Previous work established that structural symmetries arise across nearly all common STRIPS planning benchmarks (Domshlak, Katz, and Shleyfman 2013; Shleyfman et al. 2015; Sievers et al. 2015a). As we have seen that we might find fewer structural symmetries of the lifted representation, we report quantitative results for computing structural symmetries of the lifted representation, including planning benchmarks with conditional effects and axioms. We implemented the symmetry graph described in the previous section in the translator component of the Fast Downward planning system (Helmert 2006). Using the graph automorphism tool Bliss (Junttila and Kaski 2007), we then compute a symmetry group for a given planning task in PDDL.

We use the full set of planning benchmarks from the sequential tracks of all International Planning Competitions (IPCs), including tasks that were used several times only once. This gives rise to 2518 problems in 77 domains. Each run is limited to 2GB of memory and 30 minutes runtime.

Ideally, we would report the size of the symmetry groups, but as pointed out earlier, even computing a set of generators of the automorphism group is not known to be polynomial-time. Instead, we report the number of automorphisms found by Bliss, i. e. the number of generators of the subgroup we find for the planning task at hand. Additionally, we report the order for these generators. The *order* of a symmetry generator σ is defined as the minimum number of compositions with itself that yields the identity element.

Table 1 shows domain-wise results. The first two columns list the total number of tasks and the number of tasks where at least one generator can be found. Columns 3 and 4 show the sum and the median of the number of found generators. The fifth column reports the geometric mean of the runtime required to compute the generators. The last two columns show the geometric mean and the median of the order of the generators. The last row aggregates the results over all domains, using the same aggregation functions as for the domains.

Looking at the number of tasks with symmetries, we note that almost all domains exhibit symmetries. Furthermore, most of these domains exhibit symmetries in most of their tasks. Put the other way round, there are only 9 domains with no symmetries and 26 domains where the median of the number of generators is 0, i. e. there are more tasks without than with symmetries. In total, more than half of the tasks (1430/2518) exhibit symmetries. We also observe that the computation of symmetries is cheap in terms of runtime. To be precise, there are only 31 tasks for which the computation takes *more* than 2s. Only for one task (in PSR-SMALL) the symmetry computation did not finish within 300s, but this is in fact a ground task with a very large number of duplicate actions in the PDDL formalization. For all other tasks, the maximum computation time is 18.81s.

As mentioned above, we also assess the orders of the generators we find on the benchmarks. With the only exception of two domains, namely OPTICAL-TELEGRAPHS and

graph automorphism tools can efficiently compute the generators of a subgroup thereof even for large graphs.

	# tasks		# generators		time	orders	
	total	symm	sum	med	mean	mean	med
AIRPORT	50	50	177	4	0.5	2	2
ASSEMBLY	30	29	260	8	0	2	2
BARMAN-OPT14-STRIPS	14	14	45	3	0	2	2
BARMAN-SAT14-STRIPS	20	20	98	5	0	2	2
BLOCKS	35	0	0	0	0	-	-
CAVEDIVING-14-ADL	20	5	6	0	0	2	2
CHILDSNACK-OPT14-STRIPS	20	20	765	38	0.1	2	2
CHILDSNACK-SAT14-STRIPS	20	20	1241	59.5	0.1	2	2
CITYCAR-OPT14-ADL	20	20	87	4	0	2	2
CITYCAR-SAT14-ADL	20	20	107	5	0	2	2
DEPOT	22	22	72	2	0	2	2
DRIVERLOG	20	14	18	1	0	2	2
ELEVATORS-OPT11-STRIPS	20	2	2	0	0	2	2
ELEVATORS-SAT11-STRIPS	20	14	30	2	0	2	2
FLOORTILE-OPT14-STRIPS	20	1	1	0	0	2	2
FLOORTILE-SAT14-STRIPS	20	0	0	0	0	-	-
FREECELL	80	1	1	0	0	2	2
GED-OPT14-STRIPS	20	20	40	2	0	2	2
GED-SAT14-STRIPS	20	20	40	2	0	2	2
GRID	5	0	0	0	0	-	-
GRIPPER	20	20	460	23	0	2	2
HIKING-OPT14-STRIPS	20	20	60	3	0	2	2
HIKING-SAT14-STRIPS	20	20	85	4.5	0	2	2
LOGISTICS00	28	19	25	1	0	2	2
LOGISTICS98	35	33	1467	17	0	2	2
MAINTENANCE-OPT14-ADL	5	3	5	1	0	2	2
MAINTENANCE-SAT14-ADL	20	0	0	0	0.2	-	-
MICONIC	150	11	12	0	0	2	2
MICONIC-FULLADL	150	150	171	1	0.1	2	2
MICONIC-SIMPLEADL	150	11	12	0	0	2	2
MOVIE	30	30	2895	96.5	0.1	2	2
MPRIME	35	21	230	2	0	2	2
MYSTERY	30	16	169	1	0	2	2
NOMYSTERY-OPT11-STRIPS	20	10	14	0.5	0.4	2	2
NOMYSTERY-SAT11-STRIPS	20	14	24	1	0.9	2	2
OPENSTACKS-OPT08-ADL	30	30	224	7	0	2	2
OPENSTACKS-OPT14-STRIPS	20	14	257	14	0.1	2	2
OPENSTACKS-SAT08-ADL	30	30	225	7.5	0	2	2
OPENSTACKS-SAT14-STRIPS	20	12	87	2	0.4	2	2
OPTICAL-TELEGRAPHS	48	48	96	2	0.1	6.4	2
PARCPRINTER-OPT11-STRIPS	20	6	18	0	0	2	2
PARCPRINTER-SAT11-STRIPS	20	4	12	0	0	2	2
PARKING-OPT14-STRIPS	20	0	0	0	0	-	-
PARKING-SAT14-STRIPS	20	0	0	0	0	-	-
PATHWAYS	30	30	257	9	0.1	2	2
PATHWAYS-NONEG	30	30	257	9	0.1	2	2
PEGSOL-OPT11-STRIPS	20	8	13	0	0	2	2
PEGSOL-SAT11-STRIPS	20	7	12	0	0	2	2
PHILOSOPHERS	48	48	48	1	0	20.3	25.5
PIPESWORLD-NOTANKAGE	50	36	69	1	0	2	2
PIPESWORLD-TANKAGE	50	47	977	13.5	0.1	2	2
PSR-LARGE	50	4	4	0	0	2	2
PSR-MIDDLE	50	3	3	0	0	2	2
PSR-SMALL	50	48	2024	8	0	2	2
ROVERS	40	1	1	0	0	2	2
SATELLITE	36	36	1813	12	0	2	2
SCANALYZER-OPT11-STRIPS	20	17	138	6.5	0	2	2
SCANALYZER-SAT11-STRIPS	20	18	150	8.5	0	2	2
SCHEDULE	150	32	37	0	0	2	2
SOKOBAN-OPT11-STRIPS	20	20	1005	37.5	0.2	2	2
SOKOBAN-SAT11-STRIPS	20	20	1128	44	0.2	2	2
STORAGE	30	28	157	3	0	2	2
TETRIS-OPT14-STRIPS	17	1	1	0	0	2	2
TETRIS-SAT14-STRIPS	20	4	4	0	0	2	2
THOUGHTFUL-SAT14-STRIPS	20	20	20	1	0	2	2
TIDYBOT-OPT14-STRIPS	20	0	0	0	0	-	-
TIDYBOT-SAT11-STRIPS	20	0	0	0	0	-	-
TPP	30	29	105	3	0	2	2
TRANSPORT-OPT14-STRIPS	20	4	4	0	0	2	2
TRANSPORT-SAT14-STRIPS	20	0	0	0	0.1	-	-
TRUCKS	30	28	85	3	0	2	2
TRUCKS-STRIPS	30	28	85	3	1.6	2	2
VISITALL-OPT14-STRIPS	20	14	21	1	0	2	2
VISITALL-SAT14-STRIPS	20	20	30	1.5	1	2	2
WOODWORKING-OPT11-STRIPS	20	11	18	1	0	2	2
WOODWORKING-SAT11-STRIPS	20	11	43	1	0	2	2
ZENOTRAVEL	20	13	18	1	0	2	2
Summary	**2518**	**1430**	**18553**	**5**	**0**	**2.1**	**2**

Table 1: Domain-wise results: number of tasks without and with symmetries, number of generators (sum and median), computation time in seconds (geometric mean), and orders of symmetry generators (geometric mean and median).

PHILOSOPHERS, all generators are of (the smallest possible) order 2, except one generator of order 4 in SOKOBAN-OPT11-STRIPS. In each PHILOSOPHERS task, there is exactly one generator that rotates through all philosophers and forks, and hence the order corresponds to the number of philosophers (and forks). Similarly for all OPTIMAL-TELEGRAPHS tasks, there is one generator that rotates the stations, and one simple generator of order 2 which swaps stations pairwise.

Having established that we can find many structural symmetries directly of the lifted representation, in what follows we discuss their potential applications.

Discussion and Future Work

We transferred the notion of structural symmetries to the lifted representation of planning tasks and showed that with rational grounding techniques, these are also 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 as these, 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 have seen 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.

Still, this theoretical result is important to clarify the relation of our work to previous notions of symmetry. For practical applications we see the potential of our structural symmetries rather 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 grounded representation of the task.

One potential application is the generation of invariants, which are used for strengthening other techniques, e. g. in constrained PDBs (Haslum, Bonet, and Geffner 2005) or dead-end detection (Lipovetzky, Muise, and Geffner 2016). Invariants are also crucial for the transformation of the task into Finite Domain Representation, 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 group usually scales better with the size of the planning task but requires 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 possible to only consider a limited number of objects for the verification of lifted invariant candidates. We expect that with our structural symmetries it is possible to further extend the scope of this line of work.

Another interesting direction for future work is speeding up the grounding process. As for generating invariants, the core question for grounding is what is reachable in the state space. We plan to exploit structural symmetries by only considering a subset of the objects in this reachability analysis.

Yet another potential direction is task reformulation. Riddle et al. (2016) have shown that it can be beneficial to reformulate a planning task so that for symmetric objects only the number of objects that share specific properties is represented but not which exact objects these are. Many of the criteria they use for detecting suitable objects are naturally subsumed by structural symmetries, so we expect that we can exploit them to apply similar state space transformations to a wider range of planning domains.

In this paper, we have laid the theoretical foundation for a sound exploitation of symmetries in these 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 seems indeed promising.

References

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

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced symmetry breaking in cost-optimal planning as forward search. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*. AAAI Press.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2013. Symmetry breaking: Satisficing planning and landmark heuristics. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 298–302. AAAI Press.

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, Haifa.

Edelkamp, S., and Helmert, M. 1999. Exhibiting knowledge in planning problems to minimize state encoding length. In Biundo, S., and Fox, M., eds., *Recent Advances in AI Planning. 5th European Conference on Planning (ECP 1999)*, volume 1809 of *Lecture Notes in Artificial Intelligence*, 135–147. Heidelberg: Springer-Verlag.

Edelkamp, S. 2001. Planning with pattern databases. In Cesta, A., and Borrajo, D., eds., *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, 84–90. AAAI Press.

Fox, M., and Long, D. 1999a. The detection and exploitation of symmetry in planning problems. In Dean, T., ed., *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 1999)*, 956–961. Morgan Kaufmann.

Fox, M., and Long, D. 1999b. The detection and exploitation of symmetry in planning problems. Technical Report 1/99, Department of Computer Science, University of Durham.

Gerevini, A., and Schubert, L. 1998. Inferring state constraints for domain-independent planning. In Rich, C., and Mostow, J., eds., *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI 1998)*, 905–912. AAAI Press.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, 1163–1168. AAAI Press.

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

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

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

Junttila, T., and Kaski, P. 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX 2007)*, 135–149. SIAM.

Köhler, J., and Hoffmann, J. 2000. On the instantiation of ADL operators involving arbitrary first-order formulas. In *Proceedings of the ECAI 2000 Workshop on New Results in Planning, Scheduling and Design (PuK2000)*, 74–82.

Li, N.; Fan, Y.; and Liu, Y. 2013. Reasoning about state constraints in the situation calculus. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 997–1003. AAAI Press.

Lin, F. 2004. Discovering state invariants. In Dubois, D.; Welty, C. A.; and Williams, M.-A., eds., *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR 2004)*, 536–544. AAAI Press.

Lipovetzky, N.; Muise, C.; and Geffner, H. 2016. Traps, invariants, and dead-ends. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 211–215. AAAI Press.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting problem symmetries in state-based planners. In Burgard, W., and Roth, D., eds., *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, 1004–1009. AAAI Press.

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. 1998. A planning algorithm not based on directional search. In Cohn, A. G.; Schubert, L.; and Shapiro, S. C., eds., *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR 1998)*, 617–624. Morgan Kaufmann.

Rintanen, J. 2000. An iterative algorithm for synthesizing invariants. In Kautz, H., and Porter, B., eds., *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*, 806–811. AAAI Press.

Rintanen, J. 2008. Regression for classical and nondeterministic planning. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, 568–572.

Rintanen, J. 2017. Schematic invariants by reduction to ground invariants. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, 3644–3650. AAAI Press.

Seipp, J., and Helmert, M. 2013. Counterexample-guided Cartesian abstraction refinement. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 347–351. AAAI Press.

Shleyfman, A.; Katz, M.; Helmert, M.; Sievers, S.; and Wehrle, M. 2015. Heuristics and symmetries in classical planning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, 3371–3377. AAAI Press.

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 Hölldobler, S.; Krötzsch, M.; Peñaloza-Nyssen, R.; and Rudolph, S., eds., *Proceedings of the 38th Annual German Conference on Artificial Intelligence (KI 2015)*, volume 9324 of *Lecture Notes in Artificial Intelligence*, 151–165. Springer-Verlag.

Sievers, S.; Wehrle, M.; Helmert, M.; Shleyfman, A.; and Katz, M. 2015b. Factored symmetries for merge-and-shrink abstractions. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, 3378–3385. AAAI Press.

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