Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)

Symmetry-Based Task Reduction for Relaxed Reachability Analysis

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

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

Abstract

Relaxed reachability analysis is relevant to efficient grounding, invariant synthesis as well as the computation of relaxation-based heuristics. Planning domains are typically specified in a lifted representation, where the size of the tasks grows exponentially with the number of objects in the world. This growth also affects the analysis of relaxed reachability. We present a task reduction based on symmetries of the lifted representation that allows to perform the same analysis on smaller tasks.

1 Introduction

In the field of automated planning, problems are often described in a high level language, such as PDDL (McDermott et al. 1998). However, most planning approaches do not operate on the level of the schematic PDDL but on structurally simpler, non-schematic formalisms such as STRIPS (Fikes and Nilsson 1971) or SAS$^+$ (Bäckström and Nebel 1995). Therefore, a translation to one of these simpler formalisms constitutes the first step of many planning systems.

Grounding denotes the transformation from a schematic to a non-schematic representation. In principle, one just needs to replace variables in the task description in all possible ways with constants. While it is unavoidable that grounding can increase the task size exponentially, we can still do much better than such naive grounding. For example, we do not need to create an operator that will never be applicable in any reachable state of the task.

For the transformation to the finite-domain representation of SAS$^+$ tasks, we need to detect so-called *mutexes* – pairs of propositional variables of which at most one is true in any reachable state. The resulting finite-domain representation is crucial for the success of many heuristic functions, such as abstraction based heuristics (Culberson and Schaeffer 1998; Edelkamp 2001; Helmert et al. 2014; Katz and Domshlak 2010; Seipp and Helmert 2013). Mutexes also can be used directly to enhance the quality of a heuristic, for example in constrained PDBs (Haslum, Bonet, and Geffner 2005) or the landmark heuristic (Richter and Westphal 2010).

Intelligent grounding and mutex detection depend on properties of all reachable states. As the analysis of the reachable states is infeasible in general, these techniques are commonly based on a *relaxed* reachability analysis that operates on an over-approximation of the reachable states.

In this paper we examine the potential of *structural symmetries* of PDDL tasks (Sievers et al. 2017) for such reachability analysis: we will exploit symmetries to reduce the size of the task before analyzing reachability, and re-construct the result for the original task in a post-processing step. We will consider both, the very strong relaxation typically used for grounding, and the more moderate relaxation for mutex detection. In both cases, we can perform a task reduction.

In our experimental evaluation we show that the approach is most beneficial for mutex generation, leading to a substantial speedup across several domains.

2 Background

In this paper, we cover the full fragment of non-numeric, non-temporal PDDL tasks, working on a normalized representation, which can cheaply be generated (Helmert 2009).

Normalized PDDL Tasks

A *normalized PDDL task* $\Pi = \langle \mathcal{L}, \mathcal{O}, \mathcal{A}, I, G \rangle$ is defined over a first-order language $\mathcal{L}$ that consists of a finite number of predicates, variables and constants. The predicates are partitioned into *fluent predicates* and *derived predicates*. For formulas over $\mathcal{L}$ the *free variables* are defined as usual in first-order logic. Formulas not containing any free variables are called *ground*.

The *initial state specification* I is a conjunction of ground atoms with fluent predicates. The *goal specification* G is a conjunction of ground literals. $\mathcal{O}$ and $\mathcal{A}$ are finite sets of *schematic operators* and *schematic axioms*, respectively. A schematic operator $o = \langle pre(o), \mathit{eff}(o) \rangle$ consists of the *precondition* $pre(o)$, which is a conjunction of literals over $\mathcal{L}$, and the *effect* $\mathit{eff}(o)$. This effect is a conjunction of *universally quantified conditional effects* e of the form $\forall v_1 \dots v_k : cond(e) \rhd \mathit{eff}(e)$, where $v_1, \dots, v_k$ are variables from $\mathcal{L}$, $cond(e)$ is a conjunction of literals, and $\mathit{eff}(e)$ is a literal over $\mathcal{L}$ excluding derived predicates. If $\mathit{eff}(e)$ is a positive literal, we call e an *add effect* of o, otherwise we call it a *delete effect*. A *schematic axiom* a has the form $head(a) \leftarrow body(a)$, where $head(a)$ is an atom with a derived predicate and $body(a)$ is a conjunction of literals. All free variables in $body(a)$ must be free in $head(a)$, and the

set of axioms must be *stratifiable* (Thiébaux, Hoffmann, and Nebel 2005), which is an ordering property that guarantees that the outcome of axiom-evaluation is well-defined.

The notion of free variables is extended to operators and axioms as follows: for an operator effect $e_1 \wedge \cdots \wedge e_n$ the free variables are $\mathit{free}(e_1) \cup \cdots \cup \mathit{free}(e_n)$, where $\mathit{free}(\forall v_1 \dots v_k : \mathit{cond}(e_i) \triangleright \mathit{eff}(e_i)) = (\mathit{free}(\mathit{cond}(e_i)) \cup \mathit{free}(\mathit{eff}(e_i))) \setminus \{v_1, \dots, v_k\}$. For operator o the free variables are $\mathit{free}(\mathit{pre}(o)) \cup \mathit{free}(\mathit{eff}(o))$ and analogously for axiom a $\mathit{free}(\mathit{head}(a)) \cup \mathit{free}(\mathit{body}(a))$. We also refer to the free variables as the *parameters* of the operators and axioms.

Before we can define the semantics of the task, we need to explain what it means to *ground* operators and axioms. During grounding, each schematic operator o gets replaced with a set of *ground operators induced by* o that do not contain any variables. Informally, we can describe this as a two-step process. In a first step, each universally quantified conditional effect gets expanded into a conjunction of conditional effects where the variables are replaced with all possible combinations of constants. Afterwards, all variables in the operator are free. A second step iterates over all possible assignments of constants to the parameters and adds a copy of the operator where all variables are replaced with the assigned constants. Analogously, each axiom $\mathit{head}(a) \leftarrow \mathit{body}(a)$ is replaced with ground axioms by substituting the parameters in all possible ways with constants.

We now describe the semantics of a PDDL task in terms of the induced ground task. A *state* s assigns values TRUE and FALSE to all ground atoms with *fluent* predicates. The corresponding *derived state* $[\![s]\!]$ extends the assignment to *all* ground atoms, evaluating the ground axioms as in stratified logic programming. The *initial state* s_0 of the task assigns value TRUE to all atoms occurring in I, and FALSE to all other fluent ground atoms.

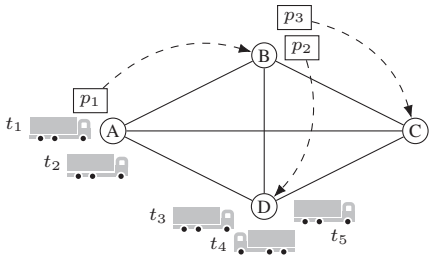
A ground operator o is *applicable* in state s if $[\![s]\!] \models \mathit{pre}(o)$. Ground atom A is true in the successor state if it has been true in s and a has no effect $\varphi \triangleright \neg A$ such that $[\![s]\!] \models \varphi$ or if o has an effect $\varphi \triangleright A$ with $[\![s]\!] \models \varphi$.

A *plan* for the task is a sequence of ground operators whose subsequent application leads from the initial state s_0 to a state $s_\star$ with $[\![s_\star]\!] \models G$. As we are only concerned with reachability, we do not discuss operator costs or plan quality.

Structural Symmetries

A *symbol mapping* σ over language $\mathcal{L}$ (Sievers et al. 2017) is a permutation of $\mathcal{L}$ that permutes the predicates, variables and constants individually (e. g. no predicate is mapped to a constant). We write $\tilde{\sigma}$ for the natural extension of σ to arbitrary expressions over language $\mathcal{L}$. For some expression e over $\mathcal{L}$, $\tilde{\sigma}(e)$ replaces all symbols s from $\mathcal{L}$ with $\sigma(s)$.

A *structural symmetry* of a PDDL task $\langle \mathcal{L}, \mathcal{O}, \mathcal{A}, I, G \rangle$ (Sievers et al. 2017) is a symbol mapping over $\mathcal{L}$ such that $I \equiv \tilde{\sigma}(I)$, for each operator $o \in \mathcal{O}$ there is $\tilde{\sigma}(o) \in \mathcal{O}$, and for each axiom $a \in \mathcal{A}$ there is $\tilde{\sigma}(a) \in \mathcal{A}$ (up to reordering of conjuncts and renaming of variables). Intuitively, applying a structural symmetry to a PDDL task results in a task with semantically equivalent initial state, operators and axioms. The set of all structural symmetries forms a group. In the original definition of structural symmetries for non-ground PDDL tasks, Sievers et al. also require $\tilde{\sigma}(G) \equiv G$, but they already discuss that this stabilization of the goal condition is not necessary for reachability analysis applications.



$$\mathit{load}(p,t,l) = \langle \mathit{pac}(p) \wedge \mathit{truck}(t) \wedge \mathit{loc}(l) \wedge \mathit{at}(p,l) \wedge \mathit{at}(t,l), \neg\mathit{at}(p,l) \wedge \mathit{in}(p,t) \rangle$$

$$\mathit{unload}(p,t,l) = \langle \mathit{pac}(p) \wedge \mathit{truck}(t) \wedge \mathit{loc}(l) \wedge \mathit{in}(p,t) \wedge \mathit{at}(t,l), \mathit{at}(p,l) \wedge \in (p,t) \rangle$$

$$\mathit{drive}(t,l,l') = \langle \mathit{truck}(t) \wedge \mathit{loc}(l) \wedge \mathit{loc}(l') \wedge \mathit{at}(t,l), \neg\mathit{at}(t,l) \wedge \mathit{at}(t,l') \rangle$$

Figure 1: Running example (simple Logistics task).

3 Running Example

We will use the very simple Logistics task from Figure 1 as a running example: there are four locations, three packages and five trucks. The initial state specification states for each constant whether it is a package, a truck or a location, and for each truck and package that it is at the location indicated in the figure. The packages need to be delivered *at* the location indicated by the arrows, using the (schematic) operators for loading/unloading packages to/from trucks and for moving the trucks. In the example, any symbol mapping that only permutes trucks at the same location and packages at the same location is a structural symmetry. The destinations of packages are irrelevant, because we only consider reachability and hence can ignore the goal.

4 Relaxed Reachability

We are interested in the following kind of relaxed reachability of literals:

Definition 1. *The set of* k-reachable *ground literals* ℓ *(*$k \in \mathbb{N}_0$*) is the smallest set that contains literal* ℓ *if*

- ℓ *is true in the initial state, or*
- $k > 0$ *and there is a ground operator* o *such that*
  - o *has an effect* $\varphi \triangleright \ell$*, and*
  - *each literal in* φ *and in* $\mathit{pre}(o)$ *is* $k-1$*-reachable, or*
- *there is a ground axiom* $\ell \leftarrow \psi$ *such that each literal in* ψ *is* k*-reachable.*

This notion is very closely related to the concept of relaxed reachability that is underlying the maximum heuristic (Bonet and Geffner 2001), extending it from atoms to literals by treating them like independent atoms. It relaxes the task in two ways: first, the interaction of literals occurring in conditions is not considered but it is sufficient if the literals

are *individually* relaxed reachable. Second, operator effects that make a literal false are ignored. This corresponds to the well-known delete relaxation.

A ground atom is k-reachable if it occurs in layer k of the Graphplan (Blum and Furst 1997) planning graph in the absence of mutual exclusion relations, which happens naturally in the delete relaxation (Hoffmann and Nebel 2001).

The important property of relaxed reachability is that it is an over-approximation of the actual reachability of literals: if a literal is relaxed unreachable then it will be false in every reachable (derived) state of the planning task.

5 Symmetry-based Reduction and Expansion

The task reduction will be based on sets C of pair-wise symmetric constants, where each symbol mapping that just swaps two elements of C is a structural symmetry of the task. As each permutation of C can be viewed as a sequence of such swaps and the structural symmetries form a group, this is equivalent to the following definition:

Definition 2 (C-permutation, symmetric constant set)**.** *Let Π be a PDDL task over $\mathcal{L}$ with constants $\mathcal{C}$ and let $C \subseteq \mathcal{C}$.*

A C-permutation is a symbol mapping $\sigma : \mathcal{L} \to \mathcal{L}$ with $\sigma(x) = x$ for $x \notin C$.

If each C-permutation is a structural symmetry of Π then C is a symmetric constant set*.*

For a C-permutation σ, the restriction $\sigma|_C$ of σ to C is a permutation of C.

Also, the set of C-permutations, together with the function composition operator, forms a group. In particular, if σ is a C-permutation then also σ^{-1} is a C-permutation.

A subset C' of a symmetric constant set C is a symmetric constant set since each C'-permutation is a C-permutation.

For an illustration in the example task, consider $C_1 = \{t_3, t_4, t_5\}$ and $C_2 = \{t_1, p_1\}$. Symbol mapping σ with $\sigma(t_3) = t_4$, $\sigma(t_4) = t_5$, $\sigma(t_5) = t_3$ and $\sigma(x) = x$ otherwise is a C_1-permutation. Also σ^{-1}, mapping t_3 to t_5, t_5 to t_4, and t_4 to t_3, is a C_1-permutation. It is easy to see that all C_1-permutations are structural symmetries, so C_1 is a symmetric constant set. There are only two C_2-permutations: the identity mapping, and the mapping that only swaps t_1 and p_1. As the latter is not a structural symmetry, C_2 is *not* a symmetric constant set.

While the previous definition considered sets of constants, the following one considers sets of ground literals:

Definition 3 (C-symmetric set of literals)**.** *Let C be a set of constants. A set of ground literals L is* C-symmetric *if for each $\ell \in L$ and C-permutation σ it holds that $\tilde{\sigma}(\ell) \in L$.*

Using C_1 as above, set $\{in(p_1, t_3), in(p_1, t_4), in(p_1, t_5)\}$ is C_1-symmetric, while $\{in(p_1, t_3), in(p_1, t_4)\}$ is not.

The following notion of *reduction* will build the core of our approach. Intuitively, it simply removes all parts from the task that mention a constant that should *not* be preserved.

Definition 4 (Reduction)**.** *Let $\Pi = \langle \mathcal{L}, \mathcal{O}, \mathcal{A}, I, G \rangle$ be a PDDL task. For a set C of constants in $\mathcal{L}$ and $C' \subseteq C$, the* reduction from C to C' *of Π is the PDDL task $R_{C\downarrow C'}(\Pi) = \langle R_{C\downarrow C'}(\mathcal{L}), R_{C\downarrow C'}(\mathcal{O}), R_{C\downarrow C'}(\mathcal{A}), R_{C\downarrow C'}(I), R_{C\downarrow C'}(G) \rangle$, where*

- *$R_{C\downarrow C'}(\mathcal{L})$ removes from $\mathcal{L}$ all constants from $C \setminus C'$,*
- *$R_{C\downarrow C'}(\mathcal{O})$ and $R_{C\downarrow C'}(\mathcal{A})$ remove all operators and axioms mentioning a constant from $C \setminus C'$, and*
- *$R_{C\downarrow C'}(I)$ and $R_{C\downarrow C'}(G)$ remove all conjuncts mentioning a constant from $C \setminus C'$.*

For a set L of literals over $\mathcal{L}$, the reduction $R_{C\downarrow C'}(L)$ consists of all $\ell \in L$ that are also literals over $R_{C\downarrow C'}(\mathcal{L})$.

In our example task, the only change in the reduction from C_1 to $C_1' = \{t_3, t_4\}$ is the removal of $at(t_5, D)$ and $truck(t_5)$ from the initial state specification. If there was an operator that mentioned specifically truck t_5, the entire operator would be removed. For a set $L = \{in(p_1, t_3), in(p_1, t_4), in(p_1, t_5)\}$ of literals, $R_{C_1\downarrow C_1'}(L) = \{in(p_1, t_3), in(p_1, t_4)\}$.

Next we show that reductions preserve all symmetries that do not map preserved constants to removed constants:

Lemma 1. *Let C be a set of constants of task Π with language $\mathcal{L}$ and let $C' \subseteq C$. For every structural symmetry σ of Π with $\sigma(c) \in C \setminus C'$ for all $c \in C \setminus C'$, the symbol mapping $\sigma' := \sigma\big|_{R_{C\downarrow C'}(\mathcal{L})}$ is a structural symmetry of task $R_{C\downarrow C'}(\Pi)$.*

Proof sketch. We need to show that whenever the reduction preserves a component, it also preserves all symmetric ones or, vice versa, whenever it removes a component, it removes all inversely symmetric ones. We show this exemplarily for an operator: let o be an operator that gets removed because o mentions a constant $c \in C \setminus C'$. No $c' \notin C \setminus C'$ has $\sigma(c') \in C \setminus C'$ because then the injective σ could not stabilize $C \setminus C'$. Hence, $\sigma^{-1}(c) \in C \setminus C'$ and therefore also $\tilde{\sigma}^{-1}(o)$ mentions a constant from $C \setminus C'$ and gets removed by the reduction. The restriction of σ to the reduced language is just a formal necessity that removes "unused" mappings. □

Expansions build the counterpart of reductions:

Definition 5 (Expansion)**.** *For a literal ℓ and a set C of constants, the* expansion of ℓ with C *is the set $E_C(\ell) = \{\tilde{\sigma}(\ell) \mid \sigma$ is a C-permutation$\}$. For a set of literals L, it is $E_C(L) = \bigcup_{\ell \in L} E_C(\ell)$.*

For example, the expansion of $P(a, b, a, c)$ with $\{a, b, d\}$ is the set $\{P(a, b, a, c), P(a, d, a, c), P(b, a, b, c), P(b, d, b, c), P(d, a, d, c), P(d, b, d, c)\}$, where each element is obtained from $P(a, b, a, c)$ by applying one of the 3! permutations of $\{a, b, d\}$. For instance, $P(d, a, d, c)$ is obtained from the permutation $\{a \mapsto d, b \mapsto a, d \mapsto b\}$.

Continuing the running example using C_1 and L from above, the expansion of $R_{C_1\downarrow C_1'}(L)$ with C_1 recovers $in(p_1, t_5)$, for instance from $in(p_1, t_3)$ with the permutation that swaps t_3 and t_5. Hence the expansion of the reduction is exactly the original set L. This is no coincidence: for symmetric constant sets, the rest of this section identifies a helpful relationship between reduction and expansion.

Lemma 2. *Let C be a symmetric constant set and $C' \subseteq C$. Let L be a set of literals not mentioning a constant from $C \setminus C'$. Then $L \subseteq R_{C\downarrow C'}(E_C(L))$. Moreover, $L = R_{C\downarrow C'}(E_C(L))$ iff L is C'-symmetric.*

Proof. If $\ell \in L$ then $\sigma = \mathbf{id}$ is a C-permutation that establishes $\ell \in E_C(L)$. Moreover, $\tilde{\sigma}(\ell) = \ell$ contains no constant from $C \setminus C'$ and is therefore preserved by the reduction.

$R_{C\downarrow C'}(E_C(L))$ consists of all $\tilde{\sigma}(\ell)$ with $\ell \in L$ and σ a C-permutation that maps no constant from ℓ to a constant from $C \setminus C'$. This is exactly the set of all $\tilde{\sigma}(\ell)$ with $\ell \in L$ and σ a C'-permutation. This set is equal to L iff L is C'-symmetric. □

Informally, for symmetric constant sets, the reduction of an expansion of a symmetric set is the original set.

Conversely, we can reduce a C-symmetric set of literals to a (sufficiently large) subset $C' \subseteq C$ and re-gain the removed literals by an expansion. A key argument will be that we can swap all constants from C that are not in the preserved set C' with constants from C'.

Lemma 3. *Let C be a symmetric constant set and $C' \subseteq C$. Let L be a C-symmetric set of literals. Let b be the maximal number of different constants from C occurring in any literal from L.*

1. *$E_C(R_{C\downarrow C'}(L)) \subseteq L$.*
2. *If $|C'| \geq b$ then $L = E_C(R_{C\downarrow C'}(L))$.*

Proof. 1. If $\ell \in E_C(R_{C\downarrow C'}(L))$, there is an $\ell' \in R_{C\downarrow C'}(L)$ and a C-permutation σ such that $\ell = \tilde{\sigma}(\ell')$. As also σ^{-1} is a C-permutation and L is C-symmetric, $\ell' = \tilde{\sigma}^{-1}(\ell) \in L$.

2. Consider a literal $\ell \in L$ and let C'' be the set of constants from C that occur in ℓ. As $|C'' \setminus C'| = |C''| - |C'| + |C' \setminus C''|$ and $|C''| \leq b \leq |C'|$, it holds that $|C'' \setminus C'| \leq |C' \setminus C''|$. Thus there exists a symbol mapping σ that swaps each constant from $C'' \setminus C'$ (the constants from ℓ that get removed by the reduction) with a constant from $C' \setminus C''$ (constants in C' that do *not* occur in ℓ) and maps everything else to itself. As σ is a C-permutation and L is C-symmetric, it holds that $\ell' := \tilde{\sigma}(\ell)$ is in L. Since ℓ' contains no constant from $C \setminus C'$, it is in $R_{C\downarrow C'}(L)$. As σ^{-1} also is a C-permutation, ℓ is in the expansion of ℓ'. □

6 Reachability under Symmetry Reduction

We now establish that we can perform relaxed-reachability analysis on a reduced version of the planning task and gain the result for the original task by an expansion.

For a PDDL task defined over $\mathcal{L}$, we will use three bounds b_C^{lit}, b_C^{op} and b_C^{ax} relative to a constant set C for Π, where b_C^{lit} is an upper bound on the number of different constants from C that can occur together in a reachable ground literal over $\mathcal{L}$. For schematic operator o of Π, we can bound the number of constants from C occurring in any ground operator induced by o by the number of the operator parameters plus the number of constants from C occurring in o. With b_C^{op} we denote the maximum such bound over all operators. Analogously, b_C^{ax} is the maximum bound, over all axioms a of Π, of the number of the parameters of a plus the number of constants from C occurring in a.

The following theorem is the key result that establishes the correctness of our approach:

Theorem 1. *Let C be a symmetric constant set for some PDDL task Π and let C' be a subset of C of size $|C'| \geq \max\{b_C^{\text{lit}}, b_C^{\text{ax}}, b_C^{\text{op}}\}$. For $k \in \mathbb{N}_0$, let R_k be the set of k-reachable ground literals of Π and R'_k the set of k-reachable ground literals of $R_{C\downarrow C'}(\Pi)$. Then $R_k = E_C(R'_k)$.*

In principle, this can be shown with one very long proof by induction. As this would be incomprehensible, we split it up into several steps. The first two lemmas cover the case where an axiom makes a literal relaxed reachable.

Lemma 4. *Let C be a symmetric constant set for PDDL task Π defined over $\mathcal{L}$ and let $C' \subseteq C$ have size $|C'| \geq b_C^{\text{ax}}$. Let L be a C-symmetric set of ground literals over $\mathcal{L}$. Then ground literal ℓ over $\mathcal{L}$ can be derived from L with an axiom of Π iff there is an ℓ' that can be derived from $R_{C\downarrow C'}(L)$ with an axiom of task $R_{C\downarrow C'}(\Pi)$ such that $\ell \in E_C(\ell')$.*

Proof. "⇐": Let ℓ' be a ground literal that can be derived from $R_{C\downarrow C'}(L)$ with ground axiom $a' = \ell' \leftarrow \varphi$, induced by axiom A in $R_{C\downarrow C'}(\Pi)$, such that $\ell \in E_C(\ell')$. Let σ be a C-permutation such that $\ell = \tilde{\sigma}(\ell')$. As σ is a structural symmetry of Π, there is an axiom $\tilde{\sigma}(A)$ which induces ground axiom $\tilde{\sigma}(\ell') \leftarrow \tilde{\sigma}(\varphi)$. As all literals ℓ'' in φ are included in L and L is C-symmetric, it also contains all literals in $\tilde{\sigma}(\varphi)$, so we can derive $\tilde{\sigma}(\ell') = \ell$ from L.

"⇒": Let A be some schematic axiom of Π that induces a ground axiom $a = \ell \leftarrow \varphi$ where all ground literals from φ are true in L (ℓ can be derived with a). Let C'' denote the set of constants from C that occur in a. By an analogous reasoning as for Lemma 3, $|C' \setminus C''| \geq |C'' \setminus C'|$ and we can define a symbol mapping σ that swaps each constant from $C'' \setminus C'$ with a constant from $C' \setminus C''$ and maps all other symbols to themselves. As σ is a C-permutation and therefore a structural symmetry, it holds that $\tilde{\sigma}(A)$ is an axiom of Π. As it does not mention any constant from $C \setminus C'$, $\tilde{\sigma}(A)$ is preserved by the reduction. Furthermore, since all constants occurring in $\tilde{\sigma}(a)$ are preserved by the reduction of the language, $\tilde{\sigma}(A)$ induces a ground axiom $\tilde{\sigma}(a)$ in $R_{C\downarrow C'}(\Pi)$. Its body $\tilde{\sigma}(\varphi)$ contains only literals ℓ'' with $\tilde{\sigma}^{-1}(\ell'') \in L$. As L is C-symmetric and also σ^{-1} is a C-permutation, L contains all literals from $\tilde{\sigma}(\varphi)$, and as they do not mention constants from $C \setminus C'$, these are preserved by the reduction. Therefore, we can derive $\tilde{\sigma}(\ell)$ in the reduction. The claim follows with $\ell \in E_C(\tilde{\sigma}(\ell))$. □

The next lemma establishes that the axiom case of the relaxed reachability definition preserves C-symmetry.

Lemma 5. *Let C be a symmetric constant set for task Π and L be a C-symmetric set of ground literals. Let a be a ground axiom induced by Π. If all literals from $body(a)$ are in L, then for all C-permutations σ, Π induces ground axiom $\tilde{\sigma}(a)$ and all literals from $body(\tilde{\sigma}(a))$ are in L.*

Proof sketch. The ground axiom $\tilde{\sigma}(a)$ exists because all C-permutations are symmetry mappings and the literals of the body are in L because L is C-symmetric. □

The following two lemmas will be used to cover the case where a literal is relaxed reachable due to an operator.

Lemma 6. *Let C be a symmetric constant set for PDDL task Π defined over $\mathcal{L}$ and let $C' \subseteq C$ have size $|C'| \geq b_C^{\text{op}}$. Let L be a C-symmetric set of ground literals over $\mathcal{L}$. Then Π induces a ground operator o with effect $\varphi \triangleright \ell$ and all literals from φ and $pre(o)$ in L iff $R_{C\downarrow C'}(\Pi)$ induces a ground operator o' with effect $\varphi' \triangleright \ell'$ and all literals from φ' and $pre(o')$ in $R_{C\downarrow C'}(L)$ such that $\ell \in E_C(\ell')$.*

Proof. "$\Leftarrow$": Let o' be a ground operator of $R_{C\downarrow C'}(\Pi)$ with effect $\varphi' \triangleright \ell'$ and all literals from φ' and $pre(o')$ in $R_{C\downarrow C'}(L)$ such that $\ell \in E_C(\ell')$. Let σ be a C-permutation such that $\ell = \tilde{\sigma}(\ell')$. From o' also being an operator of Π, σ being a structural symmetry of Π, and grounding being symmetry-preserving, it holds that Π also induces ground operator $\tilde{\sigma}(o')$. As all literals from $pre(o')$ and from φ' are in $R_{C\downarrow C'}(L) \subseteq L$ and L is C-symmetric, also $\tilde{\sigma}(pre(o'))$ and $\tilde{\sigma}(\varphi')$ are in L, and hence $\tilde{\sigma}(o')$ is the required operator.

"$\Rightarrow$": Let O be the schematic operator of Π that induces o. Let C'' denote the set of constants from C that occur in o. We can argue as in Lemma 3 that $|C' \setminus C''| \geq |C'' \setminus C'|$. Therefore there is a variable mapping σ that swaps each constant from $C'' \setminus C'$ with a constant from $C' \setminus C''$ and maps all other symbols to themselves. As σ is a C-permutation and hence a structural symmetry of Π, Π also contains schematic operator $O' = \tilde{\sigma}(O)$. Neither O' nor $\tilde{\sigma}(o)$ contain constants from $C \setminus C'$, so $\Pi' := R_{C\downarrow C'}(\Pi)$ contains O' and all constants occurring in $\tilde{\sigma}(o)$. Therefore, O' induces $\tilde{\sigma}(o')$ in Π'. This operator has effect $\tilde{\sigma}(\varphi) \triangleright \tilde{\sigma}(\ell)$ and all literals from $\tilde{\sigma}(\varphi)$ and $pre(a') = \tilde{\sigma}(pre(a))$ are in L because it is C-symmetric. As they do not contain constants from $C \setminus C'$, they are also preserved in the reduction $R_{C\downarrow C'}(L)$. Moreover, σ^{-1} is a C-permutation and $\ell = \tilde{\sigma}^{-1}(\tilde{\sigma}(\ell))$, so ℓ is in the expansion of $\tilde{\sigma}(\ell)$. □

For the analogon of Lemma 5 for operators, we omit the (analogous) proof for the sake of brevity:

Lemma 7. *Let C be a symmetric constant set for task Π and L be a C-symmetric set of ground literals. If literal ℓ is k-reachable due to an induced ground operator o then for all C-permutations σ literal $\tilde{\sigma}(\ell)$ is k-reachable due to $\tilde{\sigma}(o)$.*

We now combine everything to prove the main theorem:

Proof of Theorem 1. By induction:

$k = 0$: The set L_I of ground literals that are true in the initial state of Π is C-symmetric because all C-permutations are structural symmetries of Π and thus map I onto itself. By the definition of $R_{C\downarrow C'}(I)$, the set L'_I of literals that are true in the initial state of $R_{C\downarrow C'}(\Pi)$ is exactly the set $R_{C\downarrow C'}(L_I)$. For these sets it holds by Lemma 3 that $L_I = E_C(L'_I)$.

All other ground literals can only be 0-reachable due to an axiom. Let A be the set of ground axioms a induced by Π where all literals from $body(a)$ are in L_I. Let A' be the analogous set for $R_{C\downarrow C'}(\Pi)$ and L'_I. We define $L_A = \bigcup_{a \in A} head(a)$ and $L_{A'} = \bigcup_{a \in A'} head(a)$. According to Lemma 4, $L_A = E_C(L_{A'})$. By Lemma 5, $L'_I \cup L_{A'}$ is C'-symmetric, so according to Lemma 2 $L'_I \cup L_{A'} = R_{C\downarrow C'}(L_I \cup L_A)$. Moreover, due to Lemma 5, $L_I \cup L_A$ is C-symmetric. A repeated application of Lemma 4 and analogous arguments leads to $R_0 = E_C(R'_0)$, R_0 is C-symmetric and $R_{C\downarrow C'}(R_0) = R'_0$.

Inductive step: Let $L_k \subseteq R_k$ be the set of literals that are k-reachable due to an operator and not $k-1$-reachable, and let L'_k be the corresponding set for $R_{C\downarrow C'}(\Pi)$. By Lemma 6 it holds that $L_k = E_C(L'_k)$. By Lemma 7, $R'_{k-1} \cup L'_k$ is C'-symmetric, so according to Lemma 2 $R'_{k-1} \cup L'_k = R_{C\downarrow C'}(R_{k-1} \cup L_k)$. Also due to Lemma 7, $R_{k-1} \cup L'_k$ is C-symmetric. Hence the premises of Lemma 4 are satisfied and a repeated application of Lemma 4 plus an argumentation as above establishes the claim of the theorem as well as the fact that R_k is C-symmetric and $R_{C\downarrow C'}(R_k) = R'_k$. □

7 Finding Symmetric Constant Sets

In the previous sections, we established the necessary theory to use symmetric constant sets for symmetry-based task reduction and expansion. We now discuss how to find such symmetric constants sets for a PDDL task Π. Sievers et al. (2017) describe how a subset of the structural symmetries of Π can be computed as automorphisms of a graph representation of Π. In an empirical analysis they showed that almost all tasks from the standard benchmark exhibit symmetries and that most of the detected symmetry generators σ have order two, i. e. $\sigma\sigma = \mathbf{id}$. Most commonly, these symmetry generators are *transpositions*, i. e. permutations that just swap two constants.[1]

To determine symmetric constant sets from a set Σ of symmetries, we use the following reasoning: The only C-permutation of a set $C = \{c\}$ is the identity permutation, which is a trivial structural symmetry. Thus, each singleton is a symmetric constant set. If there are two disjoint symmetric constant sets C_1 and C_2 and there exists a symmetry that swaps exactly two constants, one from C_1 and one from C_2, then also $C_1 \cup C_2$ is a symmetric constant set. To see this, let σ be such a transposition that swaps $c_1 \in C_1$ with $c_2 \in C_2$. With σ, we can show that *any* transposition of such constants is a symmetry mapping of Π: let c'_1, c'_2 be any two constants with $c'_1 \in C_1$ and $c'_2 \in C_2$ and let σ' be the transposition that swaps them. For $i \in \{1, 2\}$, let σ_i be the transposition that swaps c_i and c'_i. As C_i is a symmetric constant set and σ_i a C_i-permutation, each σ_i is a symmetry mapping of Π. The desired mapping σ' can be represented as a composition $\sigma' = \sigma_1\sigma_2\sigma\sigma_2\sigma_1$ of symmetry mappings. As the symmetry mappings form a group (Sievers et al. 2017), also σ' is a symmetry mapping of Π. The fact that $C_1 \cup C_2$ is a symmetric constant set, i. e. every $(C_1 \cup C_2)$-permutation is a symmetry mapping of Π, follows because each such permutation can be written as a composition of transpositions of constants from $C_1 \cup C_2$.

These insights give rise to Algorithm 1. If the input transpositions are structural symmetries of the task, then all sets in the computed set $\mathcal{C}$ are symmetric constant sets. Given the symmetry generators of a symmetry group of a PDDL task, Algorithm 1 can thus be run with the subset of these symmetry generators that are transpositions.

[1] In our experiments, 88% of the 18875 detected symmetry generators that only permute constants were transpositions.

Algorithm 1 Partitioning in symmetric constant sets.

Input: Set C of constants, set T of transpositions
  $\mathcal{C} \leftarrow \{\{c\} \mid c \in C\}$
  for $t \in T$ **do**
    $c, c' \leftarrow$ constants swapped by t
    $S \leftarrow$ set from $\mathcal{C}$ with $c \in S$
    $S' \leftarrow$ set from $\mathcal{C}$ with $c' \in S'$
    $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{S, S'\}) \cup \{S \cup S'\}$
  end for
  return $\mathcal{C}$

8 Tightening the Bounds

Theorem 1 uses three bounds $b_C^{\text{lit}}, b_C^{\text{ax}}$, and b_C^{op} that specify the minimum number of symmetric constants that need to be preserved (for a given symmetric constant set C and a PDDL task Π defined over $\mathcal{L}$). A closer examination reveals that these must be an upper bound on the number of different symmetric constants that can occur together in a ground literal, axiom, or operator. Moreover, from the ground axioms only those are relevant where all literals in the body are relaxed reachable. Analogously, b^{op} needs to cover only the operators where all preconditions and relevant effect conditions are relaxed reachable. Also, Lemma 6 only considers individual effects, so it is sufficient if the bound is large enough to cover each possible effect individually (always together with the operator precondition).

We use a simple procedure to tighten these bounds, using a logic program with the following facts, relative to a set of constants C:

- P-k expresses that there is a relaxed reachable ground atom for predicate P where the k-th argument is from C.
- o-x expresses that there can be an instantiation of operator o where parameter x is replaced with a constant from C and where all positive literals from the precondition are relaxed reachable.
- o-e-i expresses that there can be an instantiation of operator o where the i-th quantified variable of effect e is replaced with a constant from C and all positive literals of the effect condition are relaxed reachable.
- a-x expresses that all positive literals in the body of an instantiation of axiom a, where parameter x has been replaced with a constant from C, are relaxed reachable.

From a conceptual perspective, we use several relaxations: we ignore all negative literals in effect and preconditions and in axiom bodies. Furthermore, we only examine parameters of atoms individually and not whether they can occur together. Consider for example operator o with parameters x and y, such that the precondition is not relaxed reachable if both x and y are instantiated with constants from C. Still, both o-x and o-y can be true, so we would (possibly unnecessarily) set $b_C^{\text{op}} \geq 2$.

As determining the exact values of the facts is as hard as computing the set of relaxed reachable literals, the logic program computes an over-approximation of the true values, i. e. no fact may be false in the computed model if the real value would be true, but the opposite is allowed.

We use the following rules in the program for a given constant set C.

1. For each ground atom $P(c_1, \ldots, c_n)$ that is initially true with $c_i \in C$, there is a rule P-i.
2. For each operator o and parameter x of o there is a rule o-x :- φ, where for each atom $P(x_1, \ldots, x_n)$ in the precondition of o with $x = x_i$, φ contains the fact P-i.
3. For each effect $e = \forall v_1 \ldots v_k : \psi \triangleright P(x_1, \ldots, x_n)$ of operator o and $i \in \{1, \ldots, k\}$ there is a rule o-e-i :- φ, where φ contains Q-m for each occurrence of v_i as m-th argument of a positive literal with predicate Q in ψ.
4. For each effect $e = \forall v_1 \ldots v_k : \psi \triangleright P(x_1, \ldots, x_n)$ of operator o and $i \in \{1, \ldots, n\}$ there is a rule P-i :- φ if x_i is a variable or a constant from C. If x_i is a constant then φ is empty. If $x_i = v_j$ for some $j \in \{1, \ldots, k\}$ then $\varphi = o$-e-j. Otherwise, x_i is a parameter of o and $\varphi = o$-x_i.
5. For each axiom a and parameter x of a there is a rule a-x :- φ, where for each atom $P(x_1, \ldots, x_n)$ in the body of a with $x = x_i$, φ contains the fact P-i.
6. For each axiom a with head $P(x_1, \ldots, x_n)$ there is a rule P-i :- φ if x_i is a variable or a constant from C. If x_i is a constant then φ is empty. If x_i is a variable, $\varphi = a$-x_i.

We use an additional enhancement, exploiting static atoms: if a fluent predicate P does not occur as an affected literal of any operator effect then we can analyze whether position i can contain an object from C by scanning the initial state specification. If not, we discard all rules from cases 2, 3, and 5 whose body contains P-i.

All rules are Horn clauses, so we can efficiently compute a minimal model M of a program defined for a PDDL task $\Pi = \langle \mathcal{L}, \mathcal{O}, \mathcal{A}, I, G \rangle$ and a subset C of the constants from $\mathcal{L}$. We set b_C^{lit} to $\max_{P \in \text{predicates}(\mathcal{L})} |\{i \mid P\text{-}i \in M\}|$. For the other bounds, we need to consider that schematic operators and axioms can mention constants. For schematic operator o, let $C_{o,\text{pre}}$ denote the constants from C mentioned in the precondition of o and for each effect e, let $C_{o,e}$ denote the constants from C in the effect that do not also occur in the precondition. We can replace the bound b_C^{op} with $\max_{o \in \mathcal{O}}(|C_{o,\text{pre}}| + |\{x \mid o\text{-}x \in M\}| + \max_{\text{effect } e \text{ of } o}(|C_{o,e}| + |\{k \mid o\text{-}e\text{-}k \in M\}|))$. For schematic axiom a, let C_a denote the constants from C mentioned in a. We can tighten b_C^{ax} to $\max_{a \in \mathcal{A}}(|C_a| + |\{x \mid a\text{-}x \in M\}|)$, or 0 if there is no axiom.

In our example, there are three symmetric constant sets: $\{t_1, t_2\}, \{t_3, t_4, t_5\}$, and $\{p_2, p_3\}$. For each of them, we get the bounds $b^{\text{lit}} = b^{\text{op}} = 1$ and $b^{\text{ax}} = 0$.

9 Using Several Symmetric Constant Sets

The analysis in Section 6 only considers reductions for a single symmetric constant set. It would be interesting to exploit several such sets together. We therefore extend our concepts to collections $\mathcal{C}$ of symmetric constant sets.

Ideally, we would want to compute such a collection on a PDDL task Π and then apply Theorem 1 on subsequent reductions to suitable subsets of the symmetric constant sets. For this purpose, we first establish that for disjoint constant

sets, a reduction with one set does not break the symmetry property of the others.

Lemma 8. *Let C be a set of constants of task Π and $C' \subseteq C$. If C'' with $C \cap C'' = \emptyset$ is a symmetric constant set for Π, then C'' is a symmetric constant set for $R_{C\downarrow C'}(\Pi)$.*

Proof sketch. We need to show that each C''-permutation σ'' is a symmetry mapping of $R_{C\downarrow C'}(\Pi)$. As it is a symmetry mapping of Π, it is sufficient to show that if the reduction from C to C' removes a component mentioning a constant from C'', it also removes all symmetric (wrt. C'') components. We do this exemplarily for the operators. Let o be a schematic operator mentioning a constant from C'' that is removed by the reduction from C to C' because it mentions a constant $c \in C \setminus C'$. As σ'' is the identity for all symbols that are not in C'', $\tilde{\sigma}''(o)$ mentions c and therefore also gets removed by the reduction. □

Theorem 1 relies on three bounds b^{lit}_C, b^{ax}_C, and b^{op}_C. In the following, we will indicate the task these refer to as additional subscript and show that a reduction for a disjoint set does not affect them.

Lemma 9. *Let C be a symmetric constant set of task Π and $C' \subseteq C$ be a subset with $|C'| \geq \max\{b^{\text{lit}}_{C,\Pi}, b^{\text{ax}}_{C,\Pi}, b^{\text{op}}_{C,\Pi}\}$. For all constant sets C'' with $C \cap C'' = \emptyset$ it holds that*

- $b^{\text{lit}}_{C'',\Pi} = b^{\text{lit}}_{C'',R_{C\downarrow C'}(\Pi)}$,
- $b^{\text{op}}_{C'',\Pi} = b^{\text{op}}_{C'',R_{C\downarrow C'}(\Pi)}$, *and*
- $b^{\text{ax}}_{C'',\Pi} = b^{\text{ax}}_{C'',R_{C\downarrow C'}(\Pi)}$.

Proof sketch. We again show the lemma exemplarily for the operators. The proof for the literals and axioms is analogous. For an individual operator, the computation of the bound does not change due to the reduction, but in $R_{C\downarrow C'}(\Pi)$ the maximum bound is determined from a subset of the operators. Therefore $b^{\text{op}}_{C'',\Pi} \geq b^{\text{op}}_{C'',R_{C\downarrow C'}(\Pi)}$. The maximum also cannot get lower: let o be an operator that establishes $b^{\text{op}}_{C'',\Pi}$. If it is not removed by the reduction, we are done. If it is removed by the reduction this is because it mentions constants from $C \setminus C'$. As C is a symmetric constant set of Π, the task contains operator $\tilde{\sigma}(o)$ for all C-permutations σ. Analogous to the proof of Lemma 6 we can construct a permutation σ that swaps all constants from $C \setminus C'$ occurring in o with constants from C' not occurring in o. Then $\tilde{\sigma}(o)$ is preserved by the reduction. Because a bound on the constants from C'' in any operator induced by $\tilde{\sigma}(o)$ is also a bound for the operators induced by o, it follows that $b^{\text{op}}_{C'',\Pi} = b^{\text{op}}_{C'',R_{C\downarrow C'}(\Pi)}$. □

Also note that for two disjoint constant sets C_1 and C_2, the order of subsequent reductions or subsequent expansions does not matter, i. e., for $C'_i \subseteq C_i$ ($i \in \{1,2\}$), $R_{C_1\downarrow C'_1}(R_{C_2\downarrow C'_2}(\Pi)) = R_{C_2\downarrow C'_2}(R_{C_1\downarrow C'_1}(\Pi))$, and for a set L of literals, $E_{C_1}(E_{C_2}(L)) = E_{C_2}(E_{C_1}(L))$. This observation enables the following generalization of Theorem 1.

Corollary 1. *Let $\mathcal{C} = \{C_1, \dots, C_n\}$ be a collection of disjoint symmetric constant sets of task Π. For each $C_i \in \mathcal{C}$, let $C'_i \subseteq C_i$ be a subset of size $|C'_i| \geq \max\{b^{\text{lit}}_{C_i}, b^{\text{ax}}_{C_i}, b^{\text{op}}_{C_i}\}$. For $k \in \mathbb{N}_0$, let R_k be the set of k-reachable ground literals of Π and R'_k be the set of k-reachable ground literals of $R_{C_n\downarrow C'_n}(\dots(R_{C_1\downarrow C'_1}(\Pi))\dots)$. Then $R_k = \bigcup_{\ell' \in R'_k} E_{C_n}(\dots(E_{C_1}(\ell'))\dots)$.*

Proof. The claim follows from subsequent applications of Theorem 1, using Lemmas 8 and 9 to preserve the premises of the theorem. □

In a practical implementation, it is not necessary to apply one reduction after the other but we can use a single scan for any constant from the set $\bigcup_{i\in\{1,\dots,n\}}(C_i \setminus C'_i)$.

10 Reachability of Conjunctions

So far, we only considered the relaxed reachability of individual literals. However, for the application of computing so-called mutex groups, this does not suffice. Mutex groups are sets of atoms of which at most one can be true in every reachable state of the task. These mutex groups are the basis of the transformation of a task in finite-domain representation (Helmert 2009), which is for example crucial for the performance of abstraction-based heuristics. To compute mutex groups, we need to determine relaxed (un)reachability of conjunctions $A \wedge A'$.

For this application, we use the following extended notion of relaxed reachability:

Definition 6. *For $k \in \mathbb{N}_0$, the set M_k of k-*reachable *pairs of ground literals is the smallest set that contains pair $\{\ell, \ell'\}$ if one of the following holds:*

1. *$\ell \wedge \ell'$ is true in the initial state.*
2. *$\ell = \ell'$ and there is a ground axiom $\ell \leftarrow \psi$ such that each pair $\{\ell'', \ell'''\}$ with $\psi \models \ell'' \wedge \ell'''$ is in M_k.*
3. *$\ell \neq \ell'$ and there is a ground axiom $\ell \leftarrow \psi$ such that $\psi \wedge \ell'$ is consistent and each pair $\{\ell'', \ell'''\}$ with $\psi \wedge \ell' \models \ell'' \wedge \ell'''$ is in M_k.*
4. *$k > 0$ and there is a ground operator o such that*
   - *(i) o has effects $\varphi \rhd \ell$ and $\varphi' \rhd \ell'$,*
   - *(ii) $pre(o) \wedge \varphi \wedge \varphi'$ is consistent,*
   - *(iii) every pair $\{\ell'', \ell'''\}$ with $pre(o) \wedge \varphi \wedge \varphi' \models \ell'' \wedge \ell'''$ is in M_{k-1}, and*
   - *(iv) if ℓ is a negative literal then for all effects $\psi \rhd \bar{\ell}$ it holds that $pre(o) \wedge \varphi \wedge \varphi' \not\models \psi$, and analogously if ℓ' is a negative literal.*
5. *$k > 0$, $\ell \neq \ell'$ and there is a ground operator o such that*
   - *(i) o has an effect $\varphi \rhd \ell$,*
   - *(ii) $pre(o) \wedge \varphi \wedge \ell'$ is consistent,*
   - *(iii) for every effect $\psi \rhd \bar{\ell}'$ it holds that $pre(o) \wedge \varphi \not\models \psi$.*
   - *(iv) every pair $\{\ell'', \ell'''\}$ with $\varphi \wedge pre(o) \wedge \ell' \models \ell'' \wedge \ell'''$ is in M_{k-1}, and*
   - *(v) if ℓ is a negative literal then for all effects $\psi \rhd \bar{\ell}$ it holds that $pre(o) \wedge \varphi \wedge \ell' \not\models \psi$, and analogously if ℓ' is a negative literal.*

Case 1 covers the initial state and cases 2 and 3 reachability with axioms for individual literals and pairs, respectively. Case 4 considers the pairs of literals that are reachable together by effects of the same operator. Case 5 covers the situation in which ℓ' is preserved true by an operator that makes ℓ true. Cases 4 (iv) and 5 (v) only tighten the definition, excluding delete effects that are guaranteed to be voided by an add effect in the relevant situations. Note that on *grounded* tasks, we only need to test $\varphi \not\models \psi$ where φ, ψ are conjunctions of ground literals. This can be done by testing that no literal occurs both positively and negatively in φ and that all literals from ψ occur in φ.

This definition of relaxed reachability of pairs of literals is inspired by the reachability in the Π^m-compilation (Haslum 2009). Indeed, on STRIPS tasks, the unreachable pairs of atoms correspond to the mutexes detected by the h^2 heuristic (Haslum and Geffner 2000). These also can be computed by Rintanen's more general invariant synthesis (Rintanen 2008), and our notion is very close to his algorithm for the conjunctions of *two* literals. However, Rintanen does not support axioms and uses slightly different semantics: in the case of conflicting effects, he considers an operator inapplicable, whereas we let the add effect cover the delete effect.

For the transformation to finite-domain variables, we are mainly interested in conjunctions $\ell \wedge \ell'$ where both literals are atoms: if $\ell \wedge \ell'$ is relaxed unreachable, the two atoms can be represented by the same finite-domain variable. In this case, we can significantly speed up the computation with a further relaxation, only requiring all pairs of *atoms* to be relaxed reachable in 2, 3, 4 (iii), and 5 (iv).

To extend our earlier result for the reachability of individual literals to pairs of literals, we naturally extend the definition of expansion to pairs of literals as $E_C(\{\ell, \ell'\}) = \{\{\tilde{\sigma}(\ell), \tilde{\sigma}(\ell')\} \mid \sigma \text{ is a } C\text{-permutation}\}$.

For computing suitable bounds, we can use the same Horn rules as in Section 8. However, as case 4 considers two effects together, we need to compute the operator bound b^{op}_C as $\max_{o \in \mathcal{O}}(|C_{o,\text{pre}}| + |\{x \mid o\text{-}x \in M\}| + \max_{\text{effects } e,e' \text{ of } o: e \neq e'}(|C_{o,e} \cup C_{o,e'}| + |\{k \mid o\text{-}e\text{-}k \in M\}| + |\{k \mid o\text{-}e'\text{-}k \in M\}|))$.

Theorem 2. *Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ be a collection of disjoint symmetric constant sets for a PDDL task Π. For each $C_i \in \mathcal{C}$, let $C'_i \subseteq C_i$ be a subset of size $|C'_i| \geq \max\{b^{\text{lit}}_{C_i}, b^{\text{ax}}_{C_i}, b^{\text{op}}_{C_i}\} + b^{\text{lit}}_{C_i}$. For $k \in \mathbb{N}_0$, let M_k be the set of k-reachable pairs of ground literals of Π and M'_k be the set of k-reachable pairs of ground literals of $R_{C_n \downarrow C'_n}(\ldots(R_{C_1 \downarrow C'_1}(\Pi))\ldots)$. Then $M_k = \bigcup_{\{\ell,\ell'\} \in M'_k} E_{C_n}(\ldots(E_{C_1}(\{\ell, \ell'\}))\ldots)$.*

Proof sketch. The proof follows the same structure and ideas that lead to Corollary 1. There are two key insights:

Firstly, in Definition 6, whenever one of the cases makes $\{\ell, \ell'\}$ k-reachable, the same case also makes $\{\tilde{\sigma}(\ell), \tilde{\sigma}(\ell')\}$ k-reachable for all structural symmetries σ.

Secondly, previously the bounds ensured that there always is a symmetry that maps all constants that are removed by the reduction to those that are not removed and not "used" in the current context. In Definition 6, we need to consider an additional literal, so we need to ensure that even in the worst case where all constants from the literal get removed, there are enough unused constants remaining to map to. We therefore increased the bound accordingly by adding $b^{\text{lit}}_{C_i}$. □

11 Related Work

Over the years, symmetries have been exploited in many different ways in planning.

The work by Fox and Long (1999; 2002) is closely related because they use the same kind of symmetric constant sets as we do (not allowing constants that occur in operators to be part of the sets), albeit in a different way for a different purpose: in a Graphplan-style search, they use one constant from the set as representative, pruning all other constants on failure of the representative in the same search context.

Lin (2004) considers invariants in the situation calculus for entire domains, i. e. during the invariant analysis, the constants in the world are unknown. He uses a rank τ that specifies a sufficient number of constants that need to be considered to analyze the satisfiability and validity of formulas. This rank is distantly related to our bounds b^{X}_C.

Closest to our work is a recent approach by Rintanen (2017) that considers limited grounding for regression-based invariant synthesis, where invariants are represented as disjunctions of at most N literals. In contrast to our work, Rintanen considers a much simpler class of planning tasks with typed variables but without existential preconditions, conditional effects and axioms. Where we use symmetric constant sets, he considers sets of constants that have the same type.

12 Experiments

To evaluate the practical usefulness of our concepts, we analyze the impact of the symmetry-based task reduction on grounding and the computation of mutually exclusive pairs of atoms. We implemented all techniques in the translator component of Fast Downward (Helmert 2006).

To compute structural symmetries, we build on the implementation by Sievers et al. (2017) using the graph automorphism tool Bliss (Junttila and Kaski 2007). We restrict the computation to symmetries that permute constants only and do not stabilize the goal.

For all applications, we first determine the subset of transpositions and compute disjoint symmetric constant sets C using Algorithm 1. In a second step, we compute the upper bounds b^{lit}_C, b^{op}_C and b^{ax}_C for each constant set C as described in Section 8 and before Theorem 2. Then, we determine the number b_C of constants that we need to preserve for the intended application (c. f. Theorems 1 and 2). We use all sets C with $|C| > b_C$ for the reduction.

In our running example, we detect symmetric constant sets $\{t_3, t_4, t_5\}$, $\{t_1, t_2\}$ and $\{p_2, p_3\}$ of size > 1. For applications exploiting Theorem 1, we get bounds $b^{\text{lit}} = b^{\text{op}} = 1$ and $b^{\text{ax}} = 0$ for all sets, so it is sufficient to preserve one element from each set. For applications exploiting Theorem 2, we get the same individual bounds and, hence, must preserve two elements from each set according to the theorem.

The regular grounding process of Fast Downward instantiates a PDDL task based on a relaxed reachability analysis of atoms. To make this process produce a symmetry-reduced ground task, we modify the logic program that performs the reachability analysis to ignore constants we disregard. With this modification, the standard Fast Downward instantiation yields exactly the symmetry-reduced task.

In our example, the reduced task (as in Theorem 1) has 20 reachable ground atoms and 64 ground actions, compared to 47 and 200, respectively, in the original task.

We can expand the symmetry-reduced task to receive the full grounded task. If we have preserved constants according to Theorem 2, we can alternatively use the grounded symmetry-reduced task for mutex detection.

To compute mutex pairs for a given ground task, we implemented a logic program to determine the relaxed reachability of pairs of atoms according to Definition 6. As described in Section 10, we use the further relaxation that only requires positive literals to be relaxed reachable.[2]

Using the example task, we perform the mutex detection on a task with 160 actions (instead of the original 200). We identify 148 mutex pairs that get expanded to 185 pairs.

We ran experiments on the benchmarks from the sequential tracks of all International Planning Competitions (IPC), removing domains that have been reused in later IPCs of the same track, obtaining a total of 77 domains and 2518 tasks.[3] We set a limit of 30 minutes and 3 GB for each run.

There is a total of 18875 symmetry generators that map constants only, distributed across 65 domains, out of which 16566 are transpositions, distributed across 51 domains.

Using the bounds for symmetry-reduced grounding, we find large enough symmetric constant sets in 1004 tasks of 49 domains. While creating the size-reduced ground task is faster in all applicable domains (by at least an order of magnitude in 3 domains), regular grounding is already so fast that we do not save any computation time if we afterwards expand the resulting task (the approach is even slower in SATELLITE).

We instead focus on the more expensive computation of mutex pairs. We compare the computation of the mutexes on the original (grounded) task with the computation on the symmetry-reduced task and a subsequent expansion. Both approaches compute the same set of mutexes.

Computing mutexes on the original ground task fails 320 times due to reaching the memory limit and 481 times due to reaching the time limit. We can use symmetry reduction for the mutex computation in 610 tasks of 38 domains. This slightly reduces the number of times reaching the memory or time limit to 303 and 448, respectively. To illustrate the gain in runtime, Figure 2 compares the computation time on the 38 applicable domains, highlighting those that exhibit a significant difference.

[2]We also experimented with the computation of mutex *literal* pairs. The results in this scenario are qualitatively the same, however with larger runtime and memory requirements and hence more tasks for which the computation does not complete.

[3]The quantitative results are similar up to a few percentage points if also using duplicate domains.

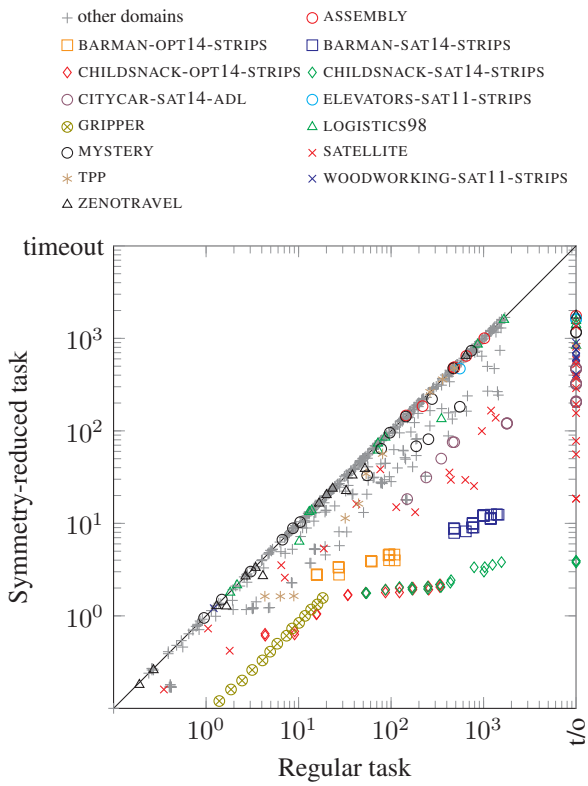


Figure 2: Runtime of regular and symmetry-reduced (including expansion) computation of h^2 mutexes on all applicable domains, highlighting domains where at least one task requires an order of magnitude more/less time.

Our approach works particularly well on the GRIPPER, the CHILDSNACK, and the BARMAN domains, which is not surprising given that these were specifically designed to exhibit a large number of symmetries. But independent from this, we observe that symmetry reduction strictly decreases the required runtime to compute mutexes, including the time to expand the resulting set of mutexes of the reduced task.

13 Conclusion

We explored the potential of symmetry-based task size reductions for relaxed reachability analysis. For this kind of analysis it is often possible to only consider a subset of a collection of pair-wise symmetric constants and to recover the full result in a post-processing step.

Our theory provides sufficient lower bounds on the size of these subsets for two types of relaxed reachability, which are particularly useful for grounding and mutex detection.

We experimentally verified the theoretical findings that the results with a direct reachability analysis and a detour via a task reduction including a post-processing step are equivalent. For grounding, the path via the task reduction gave only minor runtime improvements, mostly because the method used by Fast Downward is already very fast on the original task. In contrast, for mutex detection we observed substantial runtime improvements.

Acknowledgments

This work was supported by the European Research Council as part of the project "State Space Exploration: Principles, Algorithms and Applications" (SSX).

References

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

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

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1):5–33.

Culberson, J. C., and Schaeffer, J. 1998. Pattern databases. *Computational Intelligence* 14(3):318–334.

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.

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

Fox, M., and Long, D. 1999. 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. 2002. Extending the exploitation of symmetries in planning. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, 83–91. AAAI Press.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In Chien, S.; Kambhampati, S.; and Knoblock, C. A., eds., *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*, 140–149. 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.

Haslum, P. 2009. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 354–357. 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.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

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.

Katz, M., and Domshlak, C. 2010. Implicit abstraction heuristics. *Journal of Artificial Intelligence Research* 39:51–126.

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.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL – The Planning Domain Definition Language – Version 1.2. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

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.

Sievers, S.; Röger, G.; Wehrle, M.; and Katz, M. 2017. Structural symmetries of the lifted representation of classical planning tasks. In *ICAPS 2017 Workshop on Heuristics and Search for Domain-independent Planning*, 67–74.

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