Symmetries and Expressive Requirements for Learning General Policies

Dominik Drexler[1], **Simon Ståhlberg**[2], **Blai Bonet**[3], **Hector Geffner**[2]
[1]Linköping University, Sweden
[2]RWTH Aachen University, Germany
[3]Universitat Pompeu Fabra, Spain
dominik.drexler@liu.se, simon.stahlberg@gmail.com,
bonetblai@gmail.com, hector.geffner@ml.rwth-aachen.de

Abstract

State symmetries play an important role in planning and generalized planning. In the first case, state symmetries can be used to reduce the size of the search; in the second, to reduce the size of the training set. In the case of general planning, however, it is also critical to distinguish non-symmetric states, i.e., states that represent non-isomorphic relational structures. However, while the language of first-order logic distinguishes non-symmetric states, the languages and architectures used to represent and learn general policies do not. In particular, recent approaches for learning general policies use state features derived from description logics or learned via graph neural networks (GNNs) that are known to be limited by the expressive power of C_2, first-order logic with two variables and counting. In this work, we address the problem of detecting symmetries in planning and generalized planning and use the results to assess the expressive requirements for learning general policies over various planning domains. For this, we map planning states to plain graphs, run off-the-shelf algorithms to determine whether two states are isomorphic with respect to the goal, and run coloring algorithms to determine if C_2 features computed logically or via GNNs distinguish non-isomorphic states. Symmetry detection results in more effective learning, while the failure to detect non-symmetries prevents general policies from being learned at all in certain domains.

1 Introduction

Generalized planning is concerned with the problem of obtaining general action strategies for solving classes of instances drawn from a common domain. A classical planning domain ensures that all instances share a structure given by a set of action schemas and predicates. These general strategies, called also general plans or policies, are learned by considering a small set of training instances from the target class $\mathcal{Q}$ (Srivastava, Immerman, and Zilberstein 2011; Jiménez, Segovia-Aguas, and Jonsson 2019; Illanes and McIlraith 2019; Toyer et al. 2020; Yang et al. 2022; Srivastava 2022). General policies that solve the training instances are then expected to generalize to $\mathcal{Q}$. In the symbolic setting, where the learning problem is formulated as a combinatorial optimization problem, this generalization can often be established formally (Bonet, Francès, and Geffner 2019; Francès, Bonet, and Geffner 2021). In the deep learning setting, the algorithms scale up better but do not result in policies that can be understood and proved to be correct (Ståhlberg, Bonet, and Geffner 2022b; Ståhlberg, Bonet, and Geffner 2023).

The computational bottleneck of the symbolic approach is that it considers the complete state space of the training instances, which becomes very large quickly. For example, in the Gripper domain, where the task is to move balls from one room to another, the state space contains more than 2^n reachable states when the number of balls is n. It turns out, however, that many pairs of states in the training set are *symmetric*, meaning that a solution for one state implies a solution for the other. This suggests that the number of states for the training set can be significantly reduced by considering just one representative of each equivalent class of states.

Interestingly, state symmetries play a second important role in generalized planning. Languages and neural architectures that lack the expressive power to distinguish pairs of states that are *not* symmetric may fail to represent general policies at all for certain domains. In particular, recent approaches for learning general policies that use state features derived from description logics or learned via graph neural networks (GNNs) (Francès, Bonet, and Geffner 2021; Ståhlberg, Bonet, and Geffner 2024) are known to be limited by the expressive power of C_2, first-order logic with two variables and counting (Barceló et al. 2020; Grohe 2021).

In this work, we address the problem of detecting symmetries in planning and generalized planning and use the results for two different purposes: to assess the expressive requirements for learning general policies over planning domains, which requires distinguishing non-symmetric states, and to speed up learning, which involves grouping symmetric states together. For detecting symmetries, we map planning states to plain graphs, run off-the-shelf graph algorithms to determine whether two states are isomorphic with respect to the goal, and run coloring algorithms to determine if C_2 features computed logically or via GNNs distinguish non-isomorphic states. The expressive requirements and the performance gains are then evaluated experimentally.

The paper is organized as follows. After discussing related work, we review planning, generalized planning, and relational structures and graphs. Then, we introduce faithful and uniform abstractions, look at the notion of isomorphic relational structures (states) and the computation of such abstractions, carry out experiments, and draw conclusions.

2 Related Work

We discuss briefly three related research threads.

Symmetries. The detection of symmetries in planning has been used to prune the search space (Shleyfman et al. 2015), to define heuristic functions (Edelkamp 2001; Haslum et al. 2007; Helmert et al. 2014; Nissim, Hoffmann, and Helmert 2011), and to transform the problem representation (Riddle et al. 2016). A common thread in these approaches, which contrasts with our approach, is that actions are explicitly considered in the detection of symmetries (Pochter, Zohar, and Rosenschein 2011; Sievers et al. 2019; Sievers et al. 2017).

General policies. The problem of learning general policies has a long history (Khardon 1999; Martín and Geffner 2004; Fern, Yoon, and Givan 2006; Jiménez, Segovia-Aguas, and Jonsson 2019). General, symbolic policies have been formulated in terms of logic (Srivastava, Immerman, and Zilberstein 2011; Illanes and McIlraith 2019), regression (Boutilier, Reiter, and Price 2001; Wang, Joshi, and Khardon 2008; Sanner and Boutilier 2009), and policy rules (Francès, Bonet, and Geffner 2021; Drexler, Seipp, and Geffner 2022; Yang et al. 2022; Srivastava 2023; Silver et al. 2024). General policies have also been learned using deep learning methods (Toyer et al. 2020; Bajpai, Garg, and others 2018; Rivlin, Hazan, and Karpas 2020; Ståhlberg, Bonet, and Geffner 2022a), in many cases using graph neural networks or GNNs (Scarselli et al. 2009; Gilmer et al. 2017; Hamilton 2020).

Expressivity. Interestingly, the expressive limitations of symbolic methods relying on features derived from the domain predicates via description logic grammars (Bonet, Francès, and Geffner 2019; Francès, Bonet, and Geffner 2021) and methods relying on GNNs (Ståhlberg, Bonet, and Geffner 2022b; Ståhlberg, Bonet, and Geffner 2023) are similar. Such methods cannot distinguish states (i.e., relational structures) that cannot be distinguished by C_2, first-order logic with two variables and counting (Barceló et al. 2020; Grohe 2021), or equivalently, by the Weisfeiler-Leman (1-WL) coloring procedure (Cai, Fürer, and Immerman 1992; Morris et al. 2019; Xu et al. 2019). The consequences of this limitation have been analyzed by Ståhlberg, Bonet, and Geffner (2022a), and more recently by Horcík and Sír (2024). We will come back to this work in the discussion section.

3 Background

We review basic notions of planning, generalized planning, relational structures, and graphs.

3.1 Classical Planning

A **planning problem** is a pair $P = \langle D, I \rangle$ where D is a general first-order *domain* containing a set of predicates (or relations) R, each with given arity, and a set of action schemas of the form $\langle pre, eff \rangle$ where pre is an arbitrary first-order formula and eff is an arbitrary effect, and I is specific instance information that contains the set of objects O, and two sets of *ground atoms, Init* and *Goal*, that describe the initial and goal situations, respectively. The problem P defines the state model $S_P^\circ = \langle S, s_I, G, Act, A, f \rangle$ where the states in S are the truth valuations over the ground atoms, where each such valuation is represented by the set of atoms true in the valuation, $s_I = Init$ is the initial state, and $G = \{s \in S \mid Goal \subseteq s\}$ is the set of goal states. The function A maps states s into the set $A(s)$ of ground actions from Act that are applicable in s, and the state transition function f maps states s and actions $a \in A(s)$ into the resulting state $s' = f(s, a)$.

The *unlabeled state model* for the problem P is the tuple $S_P = \langle S, s_I, G, \mathrm{Succ} \rangle$ where the actions are compiled away, and states have a set of possible successor states instead. In this unlabeled model, the first three components are those for S_P°, while $\mathrm{Succ} = \{(s, f(s,a)) \mid a \in A(s)\}$ is the (unlabeled) successor relation.

A trajectory seeded at state s_0 in P is a state sequence $s_0, s_1, \ldots, s_n$ such that (s_i, s_{i+1}) is in Succ, $0 \leq i < n$. A state s is reachable in P if there is a trajectory seeded at the initial state s_I that ends in s. For a reachable state s, a plan (resp. optimal plan) for s is a trajectory (resp. trajectory of minimum length) seeded at s that ends in a goal state. The length of an optimal plan for state s is denoted by $V^*(s)$, and referred as the optimal cost of state s.

3.2 Generalized Planning

A **generalized planning problem** is a class $\mathcal{Q}$ of planning problems P for a common domain D (Bonet and Geffner 2018). A **general policy** π for a class $\mathcal{Q}$ is a binary relation on states. A state trajectory $s_0, s_1, \ldots, s_n$ is a π-trajectory seeded at state s_0 if (s_i, s_{i+1}) is a transition that is in both P and π, for $0 \leq i < n$. We say that: (1) π solves state s if each maximal π-trajectory seeded at s reaches a **goal state,** (2) π solves problem P if it solves the initial state of P, and (3) π solves class $\mathcal{Q}$ if it solves each problem P in $\mathcal{Q}$.

In generalized planning, *goals are encoded as part of the state* as follows. For each atom $p(\bar{o})$ that appears in the goal condition G, a new relational symbol p_g of the same arity of p is created. Then, the initial situation I is extended with the atoms $\{p_g(\bar{o}) \mid p(\bar{o}) \in G\}$ which are **static** and thus remain in every reachable state (Martín and Geffner 2004). Adding these "goal atoms" in the state allows general policies/sketches to take the specific goal of the instance into account, so they may generalize not just to instances with different numbers of objects and initial states, but also to instances with different goals.

General policies are often represented in terms of **state features.** A state feature ϕ for class $\mathcal{Q}$ is a function that maps the reachable states s for the problems in $\mathcal{Q}$ into values $\phi(s)$. The feature ϕ is Boolean if its values are Boolean values, and numerical if its values are non-negative integers. If Φ is a set of features, $\Phi(s)$ denotes the vector $(\phi(s))_{\phi \in \Phi}$.

3.3 States, Relational Structures, and Graphs

A (planning) state defines a **relational structure** $\mathfrak{A}^s$ with universe $U^s = O$ for the set of objects O in s, and interpretations $R^s \subseteq (U^s)^k$ for each predicate R of arity k in the planning domain D, where $\langle o_1, o_2, \ldots, o_k \rangle \in R^s$ iff

$R(o_1, o_2, \ldots, o_k)$ is true in s. The *signature* of a relational structure $\mathfrak{A}$ is the set of relational symbols in $\mathfrak{A}$. We assume fully relational structures that contain no functions nor constants (nullary functions). This type of structures are adequate for planning problems described in PDDL.

While a planning state defines a relational structure, relational structures can be encoded by graphs, a mapping that we will use to test state equivalence. Recall that a directed graph, or graph, is a pair $G = (V, E)$ where V is the set of vertices and $E \subseteq V^2$ is the set of edges. An undirected graph is a directed graph G where E is symmetric; i.e., $(v, w) \in E$ iff $(w, v) \in E$. Two graphs $G = (V, E)$ and $G' = (V', E')$ are **isomorphic**, denoted by $G \simeq_g G'$, if there is a bijection $f : V \to V'$ such that $(u, v) \in E$ iff $(f(u), f(v)) \in E'$.

A *vertex-colored graph* is a tuple $G = (V, E, \lambda)$ where (V, E) is a graph, and $\lambda : V \to \mathcal{C}$ maps vertices to the colors in $\mathcal{C}$. Two vertex-colored graphs $G = (V, E, \lambda)$ and $G' = (V', E', \lambda')$ are **isomorphic**, denoted as $G \simeq_g G'$, iff there is a *color preserving isomorphism* f from G to G', i.e., $\lambda(v) = \lambda'(f(v))$ for $v \in V$. If the graphs G and G' are isomorphic via the bijection f, we write $f : G \to G'$.

4 Abstractions

We formalize first the abstraction induced by an **equivalence relation** $\sim$:

Definition 1 (Abstraction). *Let $\mathcal{Q}$ be a class of problems, let $\sim$ be an equivalence relation on the reachable states of the problems in $\mathcal{Q}$, and let P be a problem in $\mathcal{Q}$ with unlabeled state model $S_P = \langle S, s_I, G, Succ\rangle$. The **abstraction** of P induced by $\sim$, denoted by $P/{\sim}$, is the **unlabeled state model** $\tilde{S}_P = \langle \tilde{S}, [s_I], \tilde{G}, \widetilde{Succ}\rangle$ where*

1. *$\tilde{S} \doteq \{[s] \mid s \in S\}$ is the set of equivalence classes for P,*
2. *$[s_I]$ is the equivalence class for initial state s_I of P,*
3. *$\tilde{G} \doteq \{[s] \mid s \in G\}$ is the set of goal classes, and*
4. *$\widetilde{Succ} \doteq \{([s], [s']) \mid (s, s') \in Succ\}$.*

The abstraction $\mathcal{Q}/{\sim}$ is the class of abstractions $\tilde{S}_P$ for the problems P in $\mathcal{Q}$.

The successor relation in $\tilde{S}_P$ is the *existential quantification* of the successor relation in S_P where $([s], [s']) \in \widetilde{\text{Succ}}$ iff there is a transition (t, t') in Succ such that $s \sim t$ and $s' \sim t'$. In particular, the transition (s, s') may not exist in P. Hence, generalized plans that solve the abstraction $\tilde{S}_P$ do not necessarily solve P. In the following, we write $(s, s') \sim (t, t')$ to denote $s \sim t$ and $s' \sim t'$.

Definition 2 (Faithful Abstractions). *Let $\mathcal{Q}$ be a class of problems, and let $\sim$ be an equivalence relation on the reachable states in $\mathcal{Q}$. The abstraction $\mathcal{Q}/{\sim}$ is **faithful** iff*

1. *for any P in $\mathcal{Q}$, any reachable transition (s, s') in P, and any reachable state t in P with $t \sim s$, there is a transition (t, t') in P such that $(s, s') \sim (t, t')$, and*
2. *if $s \sim t$ for reachable states s and t in P, then s is a goal state iff t is a goal state.*

If the abstraction $\mathcal{Q}/{\sim}$ is faithful, the binary relation that associates states s in $\mathcal{Q}$ with their equivalence classes $[s]$ in $\mathcal{Q}/{\sim}$ is a **bisimulation** between the corresponding unlabeled transition systems (Sangiorgi 2012). Indeed,

Theorem 3 (Bisimulation). *Let $\mathcal{Q}/{\sim}$ be a faithful abstraction, and let P be a problem in $\mathcal{Q}$. Then, 1) if $s_0, s_1, \ldots, s_n$ is a trajectory in S_P, then $[s_0], [s_1], \ldots, [s_n]$ is a trajectory in $\tilde{S}_P$, and 2) if $[s_0], [s_1], \ldots, [s_n]$ is a trajectory in $\tilde{S}_P$, for each s'_0 in $[s_0]$, there is trajectory $s'_0, s'_1, \ldots, s'_n$ in S_P with $s'_i \sim s_i$ for $0 \le i \le n$.*

Proof. The first claim is direct by the definition of $\tilde{S}_P$. For the second, notice that $([s_i], [s_{i+1}])$ in $\widetilde{\text{Succ}}$ implies there is a transition (s''_i, s''_{i+1}) with $(s_i, s_{i+1}) \sim (s''_i, s''_{i+1})$, for $0 \le i < n$. We construct the required trajectory in S_P inductively. By faithfulness, there is s'_1 such that (s'_0, s'_1) is in Succ and $s'_1 \sim s''_1$. Hence, $s'_1 \sim s_1$. After constructing $s'_0, s'_1, \ldots, s'_k$, we have $s'_k \sim s_k$. By faithfulness, there is transition (s'_k, s'_{k+1}) with $s'_{k+1} \sim s''_{k+1}$. Thus, $s'_{k+1} \sim s_{k+1}$, and the trajectory can be extended with s'_{k+1}. □

Corollary 4. *Let $\mathcal{Q}/{\sim}$ be a faithful abstraction, and let P be a problem in $\mathcal{Q}$. If s and t are reachable states in P with $s \sim t$, then $V^*(s) = V^*(t)$.*

Faithfulness allows us to work with the abstraction, but it does not take into account the form of the policy π. Namely, it can be the case that a transition (s, s') in P belongs to π but not a transition (t, t') with $(t, t') \sim (s, s')$. This will not happen, however, for the large class of *uniform policies*:

Definition 5 (Uniform Policies). *Let $\mathcal{Q}/{\sim}$ be an abstraction, and let Π be a class of policies for $\mathcal{Q}$. A policy π in Π is **uniform over** $\mathcal{Q}/{\sim}$ iff for any problem P in $\mathcal{Q}$, and any pair (s, s') of reachable states in P, if (t, t') is a pair of reachable states in P such that $(s, s') \sim (t, t')$, then (s, s') is in π iff (t, t') is in π. The class Π of policies is uniform over $\mathcal{Q}/{\sim}$ if each policy π in Π is so.*

A uniform policy π over a faithful abstraction $\mathcal{Q}/{\sim}$ generates **well-defined trajectories** $[s_0], [s_1], [s_2], \ldots$ on the abstraction. Let us say that the transition $([s], [s'])$ belongs to π if (s, s') belongs to π. By uniformity, if t and t' are reachable states such that $(s, s') \sim (t, t')$, then $(t, t') \in \pi$. Hence, we can lift the notions of solvability to define when a policy π solves the abstraction $\mathcal{Q}/{\sim}$. We have

Theorem 6 (Solvability). *Let $\mathcal{Q}/{\sim}$ be a faithful abstraction, and let Π be a uniform class of policies for $\mathcal{Q}/{\sim}$. Then, for any policy π in Π: π solves $\mathcal{Q}$ iff π solves $\mathcal{Q}/{\sim}$.*

Proof. Let us assume that π solves $\mathcal{Q}$, and suppose it does not solve $\mathcal{Q}/{\sim}$. That is, there is a P in $\mathcal{Q}$ with initial state s_0, and **maximal** trajectory $[s_0], [s_1], \ldots, [s_n]$ seeded at the initial class $[s_0]$ of $\tilde{S}_P$ that is not goal reaching. By Theorem 3, there is a trajectory $s'_0, s'_1, \ldots, s'_n$ in P such that $s'_i \sim s_i$, for $0 \le i \le n$. By faithfulness and uniformity, such a trajectory is a maximal π-trajectory. On the other hand, the state s'_n cannot be a goal state since $[s_n]$ is not a goal state. Hence, π cannot solve $\mathcal{Q}$, which contradicts the assumption. The other direction is shown similarly. □

In the next section, we define an equivalence relation over states that yields faithful abstractions and uniform policies, and which thus benefits from Theorem 6.

5 Isomorphic Relational Structures (States)

As planning states are relational structures, it is natural to deem two states as equivalent when their relational structures are *isomorphic,* defined as follows:

Definition 7 (Isomorphic Structures)**.** *Two relational structures $\mathfrak{A}$ and $\mathfrak{B}$, over a **common** universe U and **common** signature (without constants), are **isomorphic**, written as $\mathfrak{A} \simeq \mathfrak{B}$, iff there is a permutation σ on U such that for each relation R of arity k, $R^{\mathfrak{B}} = \{\sigma(\bar{u}) \mid \bar{u} \in R^{\mathfrak{A}}\}$, where $\sigma(\bar{u})$ for tuple $\bar{u} = \langle u_1, u_2, \ldots, u_k\rangle$ is the tuple $\langle \sigma(u_1), \sigma(u_2), \ldots, \sigma(u_k)\rangle$. We say that σ maps $\mathfrak{A}$ into $\mathfrak{B}$, and write $\sigma : \mathfrak{A} \rightarrow \mathfrak{B}$.*

Isomorphic structures satisfy the same set of sentences and the same set of formulas under suitable permutations. The following is a standard result.

Lemma 8. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be two relational structures, and let $\varphi(\bar{x})$ be a first-order formula whose free variables are among the ones in $\bar{x}$. If $\sigma : \mathfrak{A} \rightarrow \mathfrak{B}$, then for any tuple $\bar{u}$ of objects of the same length as $\bar{x}$, $\mathfrak{A} \vDash \varphi(\bar{u})$ iff $\mathfrak{B} \vDash \varphi(\sigma(\bar{u}))$. In particular, if φ is a sentence (i.e., it has no free variables), $\mathfrak{A} \vDash \varphi$ iff $\mathfrak{B} \vDash \varphi$.*

In the STRIPS setting where classes $\mathcal{Q}$ consist of problems over a common domain, isomorphism-based equivalence of states yields faithful abstractions:

Theorem 9 (Isomorphism-Based Equivalence)**.** *Let $\mathcal{Q}$ be a class of STRIPS problems over domain D. If $\sim_{iso}$ is the equivalence relation on the reachable states in $\mathcal{Q}$ such that $s \sim_{iso} t$ iff $\mathfrak{A}^s \simeq \mathfrak{A}^t$, then $\mathcal{Q}/{\sim_{iso}}$ is a **faithful abstraction.***

Proof (sketch). Let P be a problem in $\mathcal{Q}$, let (s, s') be a reachable transition in P, and let t be a reachable state in P with $t \sim_{iso} s$. We need to show that there is a transition (t, t') in P with $t' \sim_{iso} s'$. By assumption, $\sigma : \mathfrak{A}^s \rightarrow \mathfrak{A}^t$ for some permutation σ, and there is a ground action $a(\bar{o})$ with $s' = f(s, a(\bar{o}))$. In particular, $\mathfrak{A}^s \vDash pre(\bar{o})$ and thus, by Lemma 8, $\mathfrak{A}^t \vDash pre(\sigma(\bar{o}))$ (i.e. the ground action $a(\sigma(\bar{o}))$ is applicable in t). It is not hard to show that $t' \sim_{iso} s'$ for $t' = f(t, a(\sigma(\bar{o})))$.

Finally, to show the second condition in Definition 2, let P be a problem in $\mathcal{Q}$. As the states in P are assumed to contain the goal atoms for the problem, the **sentence** $\varphi_g = \bigwedge_p \forall \bar{x} \big[p_g(\bar{x}) \rightarrow p(\bar{x})\big]$, where the conjunction is over all predicates p in D, p_g is the goal predicate for p, and the size of $\bar{x}$ is the arity of p, determines whether a state s in P is a goal state; i.e., s is a goal state iff $\mathfrak{A}^s \vDash \varphi_g$. Hence, if s and t are reachable states in P such that $s \sim_{iso} t$, then $\mathfrak{A}^s \vDash \varphi_g$ iff $\mathfrak{A}^t \vDash \varphi_g$; i.e., s is a goal state iff t is a goal state. □

Example. *Let us consider the Gripper domain, where the goal is to move **all balls** from room A to room B with a robot. The robot has two grippers, it can move between the rooms, and it can pick and drop balls with any of the grippers. As the goal is for all balls to be in room B, two states are equivalent if both have the same number of balls in each room, and the robot is in the same room in each state.*

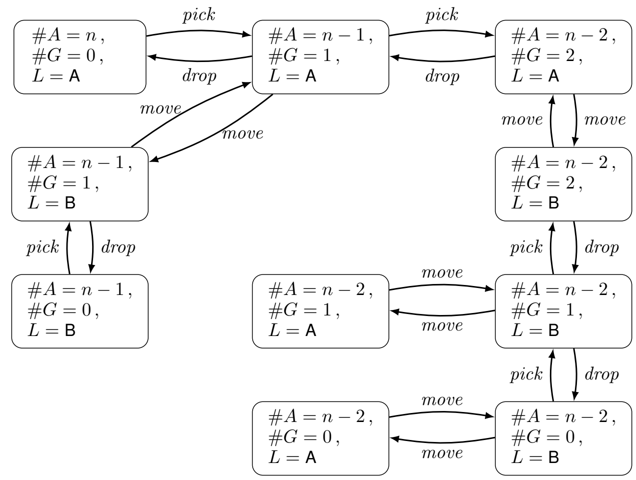

Figure 1: Fragment of the state model $\tilde{S}_P$ for a Gripper instance with n balls. Each equivalence class is identified by the number of balls at room A ($\#A$), the number of balls being held ($\#G$), and the position of the robot (L). For better understanding, we label transition with the action schemas that induce them. The abstraction contains $6n$ abstract states (see text).

If P is an instance with n balls, the number of non-isomorphic states is $6n = 2[(n+1)+n+(n-1)]$: for each of the two possible positions of the robot, there are $n+1$ states with no ball being held, n states with one ball being held, and $n-1$ states with two balls being held. On the other hand, the (plain) state space contains an exponential number of states: when no ball is being held, for example, each ball and the robot can be in either room, for a total of 2^{n+1} states. Thus, abstractions for Gripper are exponentially smaller. Figure 1 shows a fragment of the state model $\tilde{S}_P$ for the abstraction of P, where each "abstract state" is represented with the features $\Phi = \{\#A, \#G, L\}$ where $\#A$ counts the number of balls in room A, $\#G$ counts the number of balls being held, and L is the position of the robot, either A or B. The number of balls in room B is determined by the features $\#A$ and $\#G$. □

The general policies π defined in terms of rules (Bonet, Francès, and Geffner 2019), and GNNs (Ståhlberg, Bonet, and Geffner 2022a) are uniform for the abstraction $\mathcal{Q}/{\sim_{iso}}$, and hence, π solves $\mathcal{Q}$ iff π solves $\mathcal{Q}/{\sim_{iso}}$. To see this, let us say that a policy π is **function-based** if there is a function f that maps reachable states in $\mathcal{Q}$ into a domain Dom_f such that to determine whether a state pair (s, s') is in π, it is sufficient to look at the pair of values $(f(s), f(s'))$. If the function f is invariant under $\sim_{iso}$, any policy π that is based on f is uniform for $\mathcal{Q}/{\sim_{iso}}$. Likewise, policies that select pairs (s, s') by looking at the set $\{(f(s), f(s'')) \mid (s, s'') \in \mathrm{Succ}\}$, like policies that choose pairs (s, s') that greedily minimize the value $f(s')$ over successor states s', are also uniform for $\mathcal{Q}/{\sim_{iso}}$ if f is invariant. Hence, we say that π is an **invariant function-based** policy if π is based on a function f that is invariant under $\sim_{iso}$. For such policies, Theorem 6 implies:

Theorem 10 (Main)**.** *Let $\mathcal{Q}$ be a class of STRIPS problems, and let π be an invariant function-based policy for $\mathcal{Q}$. Then, π solves $\mathcal{Q}$ iff π solves $\mathcal{Q}/{\sim_{iso}}$.*

Proof. Direct from Theorem 6 as $\mathcal{Q}/{\sim_{iso}}$ is a faithful abstraction, by Theorem 9, and π is uniform for $\mathcal{Q}/{\sim_{iso}}$. □

6 Computing The Abstraction

Checking $\sim_{iso}$ on two reachable states can be reduced to a graph-isomorphism test on vertex-colored graphs. These graphs, that we call *object graphs*, encode relational structures as vertex-colored undirected graphs. On the theoretical side, the exact complexity of graph isomorphism is still unknown, but it can be tested in quasi-polynomial time (Babai 2016). However, in practice, the test can be performed efficiently (McKay and Piperno 2014); see discussion in Babai (2016, page 83). Indeed, we use `nauty` (McKay and Piperno 2014) to compute **canonical representations** (i.e. isomorphism-invariant representations) of graphs, that we apply to the object graphs associated with states. `nauty` is a state-of-the-art tool that applies Color Refinement, recursively, using a technique called vertex individualization.

Definition 11 (Object Graphs)**.** *Let $\mathfrak{A}$ be a relational structure with universe U, and relational symbols R_i, each of arity k_i, $0 \le i < n$. The **object graph** for $\mathfrak{A}$ is the **vertex-colored undirected** graph $G(\mathfrak{A}) = (V, E, \lambda)$ where the set V of vertices consists of*

1. *vertices $v = \langle u \rangle$ with color $\lambda(v) = \perp$ for $u \in U$, and*
2. *vertices $v = \langle R_i, j, \bar{u} \rangle$ with color $\lambda(v) = \langle R_i, j \rangle$ for each relation R_i, $1 \le j \le k_i$, and tuple $\bar{u} \in (R_i)^{\mathfrak{A}}$.*

The set of edges E consists of

1. *edges connecting the vertices $\langle u_j \rangle$ and $\langle R_i, j, \bar{u} \rangle$ if $\bar{u} = \langle u_1, u_2, \ldots, u_{k_i} \rangle$, and*
2. *edges connecting the vertices $\langle R_i, j, \bar{u} \rangle$ and $\langle R_i, j+1, \bar{u} \rangle$ for $1 \le j < k_i$.*

The object graph $G(s)$ for a planning state s is the object graph $G(\mathfrak{A}^s)$ of its relational structure.

The vertices of the form $\langle u \rangle$ are called *object vertices*, and vertices of the form $\langle R, j, \bar{u} \rangle$ are called *positional-argument vertices*. The first type of edge connects object vertices to corresponding positional-argument vertices, while the second connects successive positional-argument vertices.

Example. *Figure 2 shows the object graph $G(s)$ for a state s of Gripper where there is a single ball, the robot is at room* B*, and the ball is being held. This graph is isomorphic to $G(t)$ where the state t is like s, except that the other gripper holds the ball.* □

The mapping from relation structures (states) into object graphs preserves all the information in the structures:

Theorem 12 (Reductions)**.** *Let $\mathfrak{A}$ and $\mathfrak{B}$ be two relational structures over a common universe U and signature (with no constant symbols). Then, $\mathfrak{A} \simeq \mathfrak{B}$ iff $G(\mathfrak{A}) \simeq_g G(\mathfrak{B})$.*

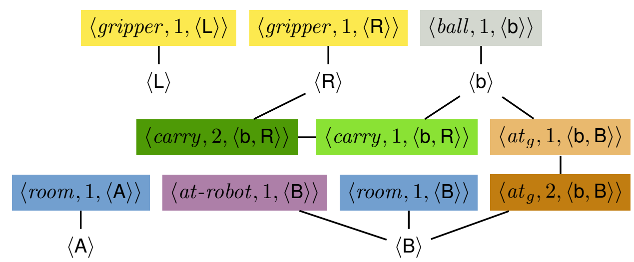

Figure 2: Object graph $G(s)$ for a state s in a Gripper instance with grippers L and R, one ball b, and two rooms A and B. In the state s, the robot is at B, the ball is at gripper R, and the goal is for the ball to be in room B. The state specifies the goal using the goal predicate at_g. This graph is isomorphic to the graph $G(t)$ for a state t that is like s except that the ball is at gripper L.

Proof (sketch). First assume $\mathfrak{A} \simeq \mathfrak{B}$ with $\sigma : \mathfrak{A} \to \mathfrak{B}$. We construct a color-preserving isomorphism f from $G(\mathfrak{A})$ to $G(\mathfrak{B})$: for object vertices, $f(\langle u \rangle) \doteq \langle \sigma(u) \rangle$, while for positional-argument vertices, $f(\langle R, j, \bar{u} \rangle) \doteq \langle R, j, \sigma(\bar{u}) \rangle$. It can be seen that f is an edge-preserving bijection between the vertices of both graphs. Additionally, $\lambda(\langle u \rangle) = \perp = \lambda(\langle \sigma(u) \rangle)$, and $\lambda(\langle R, j, \bar{u} \rangle) = \langle R, j \rangle = \lambda(\langle R, j, \sigma(\bar{u}) \rangle)$. Hence, f is a color-preserving isomorphism.

For the converse, let us assume that f is a color-preserving isomorphism from $G(\mathfrak{A})$ to $G(\mathfrak{B})$. Consider the function $\sigma : U \to U$ defined by $\sigma(u) = u'$ iff $f(\langle u \rangle) = \langle u' \rangle$. As no object vertex has the color of a positional-argument vertex, σ is a U-permutation. We need to show $\sigma : \mathfrak{A} \to \mathfrak{B}$; i.e., for each relation R,

$$R^{\mathfrak{B}} \; = \; \{\sigma(\bar{u}) \mid \bar{u} \in R^{\mathfrak{A}}\} \,. \tag{1}$$

The set of vertices related to the tuple $\bar{u}$ in $R^{\mathfrak{A}}$ is $V(\mathfrak{A}, \bar{u}) = \{\langle u_i \rangle \mid u_i \in \bar{u}\} \cup \{\langle R, j, \bar{u} \rangle \mid 1 \le j \le k\}$. This set induces the subgraph $G(\mathfrak{A}, \bar{u})$ of $G(\mathfrak{A})$. It is not hard to see that (1) holds iff the subgraphs $G(\mathfrak{A}, \bar{u})$ and $G(\mathfrak{B}, \sigma(\bar{u}))$, for all tuples $\bar{u} \in R^{\mathfrak{A}}$, are isomorphic through the (restriction of) f. As this is the case, (1) holds, and $\mathfrak{A} \simeq \mathfrak{B}$. □

By Theorem 12, we can use `nauty` to identify equivalent states. Other state encodings have been proposed that are not aimed at testing structural equivalence but at using standard GNN libraries (Ståhlberg, Bonet, and Geffner 2022a; Chen, Trevizan, and Thiébaux 2023). While the theoretical relationship between GNNs and first-order logics with counting quantifiers C_k is known (Grohe 2021), the relation between logical entailment of such logics over relational structures (i.e., states) and their different encodings (e.g., object graphs) is not clear.

7 Abstractions and Domain Expressivity

Function-based policies, as defined above, such as those captured by GNNs, do not distinguish isomorphic states. On the other hand, such policies often need to distinguish *non-isomorphic* states as they may require different actions.

We focus on two key aspects: whether a pair of non-isomorphic states (s, s') can be distinguished with GNNs, and whether a pair of states (s, s') with different V^*-value can be distinguished with GNNs. Such pairs that cannot be distinguished by any GNN are called **conflict pairs.** If a training set contains conflict pairs of the first type and s is a goal state and s' is not, then no GNN will be able to distinguish goal states from non-goal states. If the conflict is of the second type, no GNN will learn a representation of V^*, even in the training set.

We use the known relations between the counting logics C_k and Weisfeiler-Leman coloring algorithms (Cai, Fürer, and Immerman 1992), and the latter and GNNs (Morris et al. 2019; Xu et al. 2019; Barceló et al. 2020; Grohe 2021), to establish whether a domain contains conflict pairs. More precisely, we use the 1-WL and 2-FWL coloring algorithms over to the object graph $G(s)$ associated with relational structures (states) s.

It is known that if s and s' are two states whose object graphs cannot be distinguished by 1-WL, they will not be distinguished either by formulas in the logic C_2 (first-order logic with counting quantifiers and two variables), or by the embeddings produced by a GNN. And if the graphs for s and s' cannot be distinguished by 2-FWL, they cannot be distinguished by formulas in the logic C_3 or by the embeddings produced by 3-GNNs.

Graphs are compared in terms of their *histograms of colors,* denoted by $\mathsf{Hist}^k(\cdot)$ with $k = 1$ for 1-WL, and $k > 1$ for k-FWL, where such histogram is just the *multiset of colors* for the vertices in the graph. Namely, two states s and s' are distinguished if $\mathsf{Hist}^k(G) \neq \mathsf{Hist}^k(G')$, where $G = G(s)$ and $G' = G(s')$ are the corresponding object graphs.

In the experiments, we obtain the histograms by running 1-WL and 2-FWL over the object graphs for hundreds of training instances of different planning domains. Let D be a STRIPS planning domain, and let $\mathcal{Q}$ be a collection of instances P over D. If $\mathcal{S}$ denotes the set of reachable states across the instances in $\mathcal{Q}$, we want to check whether there is a pair of states (s, s') in $\mathcal{S}$ that is in *conflict* with respect to a coloring algorithm. Formally,

Definition 13 (Conflicts)**.** *Let $\mathcal{S}$ be a set of reachable states for instances over a common domain, where the states are assumed to contain goal atoms. Further, let us consider a coloring algorithm $\times$, such as 1-WL (color refinement), that operates on the* ***object graphs*** *$G(s)$, and let (s, s') be a pair of states in $\mathcal{S}$ that have the* ***same color histogram;*** *i.e.,* $\mathsf{Hist}^\times(s) = \mathsf{Hist}^\times(s')$. *Then,*

1. *(s, s') is an* ***E-conflict*** *if $s \not\sim_{iso} s'$, and*
2. *(s, s') is a* ***V-conflict*** *if $V^*(s) \neq V^*(s')$.*

We say that $\mathcal{S}$ has no conflicts of some type iff there is no pair (s, s') in $\mathcal{S}$ that is a conflict of such type.

Conflicts of the first type imply that GNNs cannot distinguish some pairs of non-isomorphic states, while conflicts of the second type imply that GNNs cannot distinguish some pairs of states that have different costs. The proof of the following theorem follows directly from the known correspondences between 1-WL and GNNs:

Theorem 14 (GNN-based Representation of V^*)**.** *Let $\mathcal{Q}$ be a* ***finite*** *class of problems over a common domain D (where states encode goals with goal atoms), and let $\mathcal{S}$ be the set of reachable states in $\mathcal{Q}$. Then,*

1. *$\mathcal{S}$ has no E-conflicts of type 1-WL iff there is a GNN that identifies the states $[s]$ in the abstraction $\mathcal{Q}/{\sim_{iso}}$, and*
2. *$\mathcal{S}$ has no V-conflicts of type 1-WL iff there is a GNN that represents the value function $V^*(s)$ over $\mathcal{S}$.*

8 Experiments: Domain Expressivity

Experiments are carried out to evaluate the expressivity requirement of various planning domains by looking for E- and V-conflicts. Testing for the equivalence relation $\sim_{iso}$ is implemented in Python using the planning library Mimir (Ståhlberg 2023) and `nauty`, while for computing color histograms we implemented 1-WL and 2-FWL (Drexler et al. 2024). The benchmark set consists of domain and instances from the International Planning Competition (IPC). Code and data are available online (Drexler et al. 2024).

Conflicts are calculated with respect to 1-WL and 2-FWL, and also versions of these algorithms in which multisets are replaced by standard sets.[1] This modification is important because the description logic grammar that is used to generate state features from the planning domain does not use counting quantifiers, and also because some GNN-based approaches use max-aggregation rather than sum-aggregation (Ståhlberg, Bonet, and Geffner 2022a; Ståhlberg, Bonet, and Geffner 2022b; Ståhlberg, Bonet, and Geffner 2023).

Table 1 shows the number of E- and V-conflicts among the reachable states in the benchmark. The table shows, for each domain, the number of instances and their reachable states ($\#\mathcal{Q}$ and $\#\mathcal{S}$), the number of equivalence classes ($\#\mathcal{S}/{\sim_{iso}}$), and the number of E- and V-conflicts ($\#E$ and $\#V$, respectively) for 1-WL and 2-FWL, and for the two versions of the algorithms (multisets and standard sets).

We also tried a slightly different graph encoding to overcome some of the limitations of object graphs in relation to the coloring algorithms. In this encoding, goals are represented using two predicates, $p_{g,T}$ and $p_{g,F}$, rather than a single predicate p_g, that tell whether the goal atom is true or false in the state. This encoding, called *goal marking,* is beneficial in all domains that have conflicts, highlighted in green in the columns "1-WL + G" and "2-FWL + G" in Table 1. In Blocks, for example, $\#V$ drops to 0, while in Ferry, it resolves all conflicts.

The existence of V-conflicts are important when learning a representation of V^*, but E-conflicts give a more general view on the expressivity requirements since V-conflicts are E-conflicts and E-conflicts imply the existence of qualitatively different states that cannot be differentiated. As can be observed on Table 1, and by Theorem 14:

- 1-WL (and hence GNNs) has sufficient expressive power in 11 domains (61%), where there are no conflicts at all.
- In 12 (resp. 14) domains, 1-WL has sufficient expressive power to separate non-isomorphic states (resp. represent V^*) when using goal marking.

[1] Coloring algorithms work with multisets, rather than sets, as multisets provide a means to do restricted forms of counting.

				Multisets								Standard sets							
				1-WL		2-FWL		1-WL + G		2-FWL + G		1-WL		2-FWL		1-WL + G		2-FWL + G	
Domain	$\#\mathcal{Q}$	$\#\mathcal{S}$	$\#\mathcal{S}/{\sim_{iso}}$	$\#E$	$\#V$	$\#E$	$\#V$	$\#E$	$\#V$	$\#E$	$\#V$	$\#E$	$\#V$	$\#E$	$\#V$	$\#E$	$\#V$	$\#E$	$\#V$
Barman	510	115 M	38 M	1,326	537	0	0	1,062	273	0	0	1,326	537	0	0	1,062	273	0	0
Blocks3ops	600	146 K	133 K	50	20	0	0	25	0	0	0	50	20	0	0	25	0	0	0
Blocks4ops	600	122 K	110 K	54	27	0	0	27	0	0	0	54	27	0	0	27	0	0	0
Blocks4ops-clear	120	31 K	3 K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Blocks4ops-on	150	31 K	8 K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Childsnack	30	58 K	5 K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Delivery	540	412 K	62 K	0	0	0	0	0	0	0	0	152	0	0	0	152	0	0	0
Ferry	180	8 K	4 K	36	36	0	0	0	0	0	0	84	84	0	0	0	0	0	0
Grid	1,799	438 K	370 K	42	38	0	0	24	20	0	0	84	80	0	0	44	40	0	0
Gripper	5	1 K	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hiking	720	44 M	5 M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Logistics	720	69 K	38 K	131	131	0	0	94	94	0	0	131	131	0	0	94	94	0	0
Miconic	360	32 K	22 K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reward	240	14 K	11 K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rovers	514	39 M	34 M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Satellite	960	14 M	8 M	5,304	4,226	0	0	1,708	762	0	0	12,908	9,906	0	0	4,372	982	0	0
Spanner	270	9 K	4 K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Visitall	660	3 M	2 M	0	0	0	0	0	0	0	0	27	0	0	0	27	0	0	0

Table 1: The column $\#\mathcal{Q}$ is the number of instances used in our experiments. The columns denoted $\#\mathcal{S}$ and $\#\mathcal{S}/{\sim_{iso}}$ refer to the the total number of states and the total number of partitions in the expanded reachable state spaces. The left part uses a multiset, while the right part uses sets. The suffix "G" indicates that goal atoms are marked as true if they hold true in the state. The number of conflicts that are caused by 1-WL and 2-FWL where the $\#E$ column refers to the total number of conflicts, while the $\#V$ column refers to the number of conflicts in which the two classes differ in V^*.

- In some domains, 1-WL is not expressive enough even with goal marking; this includes the domains Barman, Grid, Logistics, and Satellite.
- Most important, *2-FWL, that has the expressive power of* C_3*, appears to be sufficiently expressive in all domains.*

The table also shows that reducing expressiveness by using sets instead of multisets does not reduce the expressive power needed in most domains. Indeed, the modified 1-WL algorithm with sets creates E-conflicts in Delivery and Visitall (highlighted in orange), but no V-conflicts where they were none. Indeed, it just increases the number of conflicts in Ferry, Grid, and Satellite which was not zero with multisets.

Ståhlberg, Bonet, and Geffner (2023) noted that Logistics requires C_3 features to learn a value function, and similarly for Grid (Ståhlberg, Bonet, and Geffner 2024). *The experiments corroborate these claims, as 1-WL found conflicts in these domains.* However, Ståhlberg, Bonet, and Geffner (2022a) claim that Rovers requires C_3 features, but no conflicts are identified. This finding does not disprove the claim because Rovers contains an important ternary predicate, CAN-TRAVERSE; rather, it likely suggests that our training set is not sufficiently rich.

Barman, Ferry, and Satellite, as far as we know, have not been previously analyzed in this context. The conflicts in Blocks have been studied by Horcík and Sír (2024), where they show that if the goal has a specific structure, then C_2 cannot determine if it is true in a state. Logistics has been investigated by Ståhlberg, Bonet, and Geffner (2023), where they used *derived predicates* to ensure C_2 is sufficient to express a policy. The results suggest that the expressiveness of 1-WL is insufficient for learning a value function. We now study these domains and the conflicts we have identified.

Barman. The objective is to mix cocktails that require exactly 2 ingredients. To create the cocktails, the bartender can fill shot glasses with specific ingredients, pour the shot glasses into a shaker, mix the ingredients with the shaker, and clean the shot glasses and the shaker. A typical plan for creating a cocktail involves pouring the first ingredient into a shot glass, transferring it to the shaker, cleaning the shot glass, pouring the second ingredient into it, then into the shaker, cleaning the shot glass again, shaking the shaker, and finally pouring the cocktail into a shot glass. Figure 3 illustrates two states with different V^* values that cannot be distinguished by 1-WL. There are two different cocktail recipes, c_1 requiring ingredients i_1 and i_3, and c_2 requiring ingredients i_1 and i_2. The goal is to fill shot glass s_1 with c_1 and shot glass s_2 with c_2. In both states, the shaker is on the table, and both shots are being held. The distinction lies in the contents of the shot glasses. In the first state, shot glass s_1 contains i_3 and shot glass s_2 contains i_2, while in the second state, shot glass s_1 contains i_2 and shot glass s_2 contains i_3. In other words, the contents of the shot glasses have been swapped. However, the goal specifies that shot glass s_1 must precisely contain cocktail c_1, so the optimal plan for the second state is first to pour out the contents of s_1 and then clean it, as it contains the wrong ingredient, steps that are unnecessary for the first state.

Blocks. The goal is to arrange all the blocks into a specific configuration by stacking and unstacking them. There are two versions of this domain, one with three action schemas and the other with four action schemas. Remarkably, GNNs have been successfully trained for this domain and exhibit good generalization (Ståhlberg, Bonet, and Geffner 2022b;

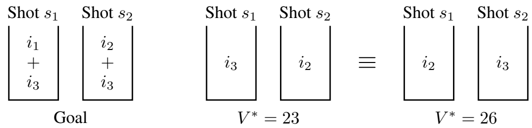

Figure 3: Example of two Barman states with different V^* value from the same instance that are considered isomorphic by 1-WL with respect to the goal. The left (resp. right) one in being held in the left (resp. right) hand, and the shaker (omitted) is on the table. The goal is to have cocktail c_1 in shot glass s_1 and c_2 in s_2. The only difference in both states is that the ingredients in both shots are swapped. However, in the state on the right, the ingredient i_2 in s_1 is wrong and must be removed, resulting in different V^* values.

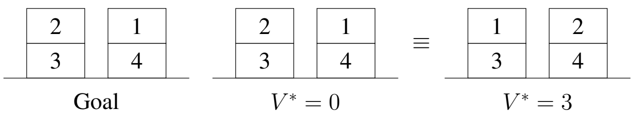

Figure 4: Example of two Blocks states that are considered isomorphic by 1-WL with respect to the goal. In the object graphs, 1-WL cannot determine whether the goal holds.

Ståhlberg, Bonet, and Geffner 2023). However, our results, along with those of others (Horcík and Sír 2024), suggest that GNNs might lack the necessary expressiveness for this domain. Figure 4 illustrates two states and a goal description that cannot be distinguished by 1-WL. In this figure, the two states have distinct values: one is a goal state, and the other is not. The object graph for the state on the left contains two connected components, each forming a 6-gon, and the object graph for the state on the right contains one connected component, forming a 12-gon. These two structures cannot be distinguished by 1-WL.

Ferry. There is only one ferry, capable of carrying a single car. The cars can both board and disembark from the ferry, and the ferry can sail between locations. The goal is to transport cars to their respective destinations, as denoted by a binary predicate. The simplest states where 1-WL fails to differentiate are those where the two cars must be in different locations. One state has both cars at their destinations, while the other has their locations swapped. Consequently, their values differ, with one being a goal state and the other not. By marking goal atoms as true or false, these two states can be distinguished.

Grid. In this domain, an agent needs to move keys to specific cells by picking them up and placing them down. However, there are locked doors, and the cells might be positioned behind one. Each locked door can only be opened by keys with the corresponding shape, i.e., both locks and keys have shapes associated with them. An example illustrating when 1-WL is insufficient for distinguishing non-isomorphic states is shown in Figure 5. In these states, the positions of two keys have been swapped, resulting in different V^* values. However, 1-WL cannot determine which key should be placed in which location.

Logistics. This domain involves cities, trucks, airplanes, and packages. In each city, there are several locations where

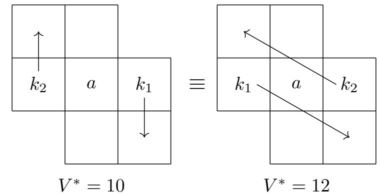

Figure 5: An example of two Grid states that are considered isomorphic by the 1-WL algorithm with respect to the goal. The goal is to move the keys k_1 and k_2 to specific cells, as the arrows indicate. All keys and locks have the same shape. The agent a is in the center of the grid. In the left state, 10 actions are needed to solve the instance, while 12 actions are needed in the right state.

trucks can move between, as well as pick up and deliver packages. There is also an airport in each city, from which airplanes can load and unload packages. The goal is to deliver each package to a specific location within some city. A plan for a single package typically involves a truck that picks it up and unloads it at the airport, then an airplane is used to move it to the correct city, after which a truck is used to deliver it to the destination. Two states with different V^* values that 1-WL cannot discriminate are as follows: There are two cities, c_1 and c_2, each consisting of a single location, which we refer to using the city name. There is a single truck in each location, t_1 at c_1 and t_2 at c_2. There are also two airplanes, a_1 at c_1 and a_2 at c_2. The goal is to deliver two packages, p_1 to c_1 and p_2 to c_2. In one state, p_1 is inside t_1 and p_2 is inside t_2, while in the other state, p_1 is inside t_2 and p_2 is inside t_1. The V^* value of the first state is 2 as the trucks have to unload the packages, while the value is 8 in the second state as they need to be transported to the other city. Here, 1-WL is unable to determine whether the correct packages are inside the trucks.

Satellite. In this domain, there are satellites equipped with instruments to capture specific images. Each satellite can calibrate the equipment to various targets, but not necessarily to all possible targets. The typical goal is to capture images of various phenomena using specific instruments. We found states with different values that are identified as isomorphic by 1-WL. One example is an instance where the goal is to capture a spectrograph image of a phenomenon, and there are two satellites capable of capturing such an image. However, only one satellite can calibrate the instrument to the phenomenon; thus, said satellite has to capture the image. The only difference between the two states is that, in the first state, one satellite is pointing to the ground station and the other is pointing to a star related to the phenomenon, whereas in the second state, their orientations have been swapped. This means that in one state, one satellite must first turn to the star to calibrate the instrument. However, 1-WL is unable to determine whether the correct satellite points to the star – only that one satellite does.

9 Experiments: Learning on Abstractions

The next set of experiments evaluates the impact of replacing the states in the training set when learning general poli-

Domain	without equivalence-based reduction				with equivalence-based reduction					
	M	T_{pre}	T_{learn}	$\#\mathcal{Q}_{\mathcal{T}}$	M	T_{pre}	T_{learn}	Speedup	$\#\mathcal{Q}_{\mathcal{T}}/{\sim_{iso}}$	Factor
Blocks3ops	**9**	**103**	28,781	145,680	11	213	**11,020**	**2.65**	**4,901**	29.72
Blocks4ops-clear	1	3	5	30,540	1	3	**3**	**1.33**	**86**	355.12
Blocks4ops-on	3	**30**	**177**	30,540	**2**	33	195	0.47	**249**	122.65
Delivery	3	107	427	411,720	**2**	**65**	**260**	**1.64**	**3,346**	123.05
Ferry	1	**13**	**56**	8,430	1	19	72	0.76	**265**	31.81
Gripper	1	2	**3**	1,084	1	2	4	0.83	**90**	12.04
Miconic	1	**8**	**30**	32,400	1	14	44	0.66	**12,339**	2.63
Reward	1	**5**	15	13,394	1	6	**8**	**1.43**	**7,026**	1.91
Spanner	1	**3**	4	9,291	1	4	4	0.88	**283**	32.83
Visitall	**2**	**22**	**55**	476,766	3	36	59	0.98	**402,880**	1.18

Table 2: Learning general policies with and without equivalence-based reductions. The table shows the memory in GiB (M), the wall-clock times in seconds for preprocessing (T_{pre}), the time in seconds for grounding, solving the ASPs, and validation (T_{learn}), the total number of states in the training set ($\#\mathcal{Q}_{\mathcal{T}}$), and the reduced training set ($\#\mathcal{Q}_{\mathcal{T}}/{\sim_{iso}}$), and ratios for the speedup in time and number of states for the reduced training set. Boldface figures denote the winner in the pairwise comparison, i.e., the one with strictly fewer resources needed.

cies with symbolic methods (Drexler, Seipp, and Geffner 2022) with their abstractions. For both training sets, the learned policies are aimed to generalize to a much larger (infinite) class of instances. The impact on performance for symbolic learning mainly results from reducing in the number of states, although some extra preprocessing is needed to implement the reduction, which, for the easiest cases, increases overall times. If $\mathcal{Q}_T$ denotes the set of states used for training, then $\mathcal{Q}_T/{\sim_{iso}}$ denotes the reduced set of states obtained in the equivalence-based abstraction where every pair of isomorphic states in $\mathcal{Q}_T$ are mapped to the same abstract state.

Learning is done on two Intel Xeon Gold 6130 CPUs with 32 cores, 96 GiB of memory, and a time budget of 24 hours. Since the reductions are significant, we use training instances with up to 10,000 states instead of the 2,000 used by Drexler, Seipp, and Geffner (2022), and we tested generalization of the learned policies on significantly larger instances.

Table 2 shows a summary of the times required for preprocessing (that includes the tests for $\sim_{iso}$) and the learning of the general policies. The sizes of the plain and reduced training sets, $\#\mathcal{Q}_{\mathcal{T}}$ and $\#\mathcal{Q}_{\mathcal{T}}/{\sim_{iso}}$ respectively, are shown, as well as the reduction factors with respect to time (Speedup) and the number of states (Factor). Notice that there is only a single state in $\mathcal{Q}_{\mathcal{T}}/{\sim_{iso}}$ for every equivalence class across all instances. As it can be seen, the total overhead incurred by testing $\sim_{iso}$ (i.e., the difference between the two figures for T_{pre}) is small.

Policy learning is done iteratively by solving a Clingo program (ASP) over a *subset* of the training set that is grown at each iteration until the resulting policy correctly solves (i.e., verifies) *all* the instances in the training set. Table 2 shows that the learning time increases for the easiest cases due to the overhead but reduces for the most difficult domains, Blocks3ops and Delivery. Our policy learning code is not optimized as it is implemented on top of the code for learning sketches (Drexler, Seipp, and Geffner 2022), a task that requires further bookkeeping. We expect better speedups by using specific code only for policy learning because they do not require computing the complete abstraction mapping and, therefore, can better exploit the reduction in abstract states.

10 Discussion

In recent work, developed independently, Horcík and Sír (2024) analyze the expressive power of a number of GNN architectures over a number of planning domains. For this, they map state pairs s and s' from a domain instance into graphs, and run GNNs with random weights to compute scalars $g(s)$ and $g(s')$.[2] The equality $g(s) = g(s')$ is a strong indication that the GNNs cannot distinguish s from s', and if the actual costs $V^*(s)$ and $V^*(s')$ are different, the pair (s, s') is marked as a conflict; an indication that GNNs lack expressive power to capture V^* in the domain. In our case, rather than using GNNs with random weights, we run 1-WL, and rather than using different types of graphs, we use a map from states (relational structures) to graphs that is invariant under state isomorphism. In addition, we see if 1-WL distinguishes non-isomorphic pairs of states and not just states with different V^* values. This is important because E-conflicts (s, s'), as we call them, may become V-conflicts when the goals encoded in s and s' change. Yet, while results over the various domains are quite different, the reasons for these differences may be elsewhere. Horcík and Sír (2024) consider large training instances but sample the state pairs that are considered; we consider small training instances and consider all possible state pairs. The result is that we observe conflicts in domains such as Barman, Blocks, Logistics, and Satellite, but not in Rovers, while they observe conflicts in Rovers but not in the first four domains.

While the presence of V-conflicts in a domain is a strong indication that GNNs will not be able to represent the optimal value function, even over the training instances, the lack of V-conflicts does not ensure that the GNNs will rep-

[2]Other mappings from states into graphs are considered by Chen, Trevizan, and Thiébaux (2023) and Chen, Thiébaux, and Trevizan (2023).

resent the optimal value function or suitable approximation of it over the test set (as in Rovers). Also, GNNs may fail to represent V^* over the training set and yet accommodate non-optimal policies. Likewise, in certain cases, this limitation can be addressed by using slightly different state encodings, as shown in the case of Blocks and Ferry where goal and state predicates p_g and p are composed. Other ways for extending the state representations are addressed by Ståhlberg, Bonet, and Geffner (2024).

11 Conclusions

State symmetries play two key roles in generalized planning. On the one hand, symmetric states can be pruned, speeding up the learning process with no information loss. On the other hand, non-symmetric states need to be distinguished by the languages and neural architectures used to represent and learn value functions and policies. Indeed, languages and architectures that lack the expressive power to make these distinctions may fail to accommodate general policies for certain planning domains at all. These two roles of symmetries and non-symmetries have been studied through a number of experiments that illustrate the expressive power required by some common planning domains and the performance gains obtained in the symbolic setting for learning general policies. In the future, we want to explore how these results can be sharpened and made more broadly useful by learning general policies for domains that remain out of reach for current techniques.

Acknowledgments

This work has been supported by the Alexander von Humboldt Foundation with funds from the Federal Ministry for Education and Research. It has also received funding from the European Research Council (ERC), Grant agreement No 885107, the Excellence Strategy of the Federal Government and the NRW Lander, Germany, and the Knut and Alice Wallenberg (KAW) Foundation under the WASP program. The computations were enabled in part by the supercomputing resource Berzelius provided by National Supercomputer Centre at Linköping University and the KAW Foundation.

References

Babai, L. 2016. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, 684–697. Association for Computing Machinery.

Bajpai, A. N.; Garg, S.; et al. 2018. Transfer of deep reactive policies for mdp planning. In *Proc. NeurIPS 2018*, 10965–10975.

Barceló, P.; Kostylev, E.; Monet, M.; Pérez, J.; Reutter, J.; and Silva, J.-P. 2020. The logical expressiveness of graph neural networks. In *Proc. ICLR 2020*.

Bonet, B., and Geffner, H. 2018. Features, projections, and representation change for generalized planning. In *Proc. IJCAI 2018*, 4667–4673.

Bonet, B.; Francès, G.; and Geffner, H. 2019. Learning features and abstract actions for computing generalized plans. In *Proc. AAAI 2019*, 2703–2710.

Boutilier, C.; Reiter, R.; and Price, B. 2001. Symbolic dynamic programming for first-order MDPs. In *Proc. IJCAI 2001*, 690–700.

Cai, J.-Y.; Fürer, M.; and Immerman, N. 1992. An optimal lower bound on the number of variables for graph identification. *Combinatorica* 12(4):389–410.

Chen, D. Z.; Thiébaux, S.; and Trevizan, F. 2023. Goose: Learning domain-independent heuristics. In *NeurIPS 2023 Workshop on Generalization in Planning*.

Chen, D. Z.; Trevizan, F.; and Thiébaux, S. 2023. Graph neural networks and graph kernels for learning heuristics: Is there a difference? In *NeurIPS 2023 Workshop on Generalization in Planning*.

Drexler, D.; Ståhlberg, S.; Bonet, B.; and Geffner, H. 2024. Code and data for the paper titled "symmetries and expressive requirements for learning general policies". https://doi.org/10.5281/zenodo.13285981.

Drexler, D.; Seipp, J.; and Geffner, H. 2022. Learning sketches for decomposing planning problems into subproblems of bounded width. In *Proc. ICAPS 2022*, 62–70.

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

Fern, A.; Yoon, S.; and Givan, R. 2006. Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *Journal of Artificial Intelligence Research* 25:75–118.

Francès, G.; Bonet, B.; and Geffner, H. 2021. Learning general planning policies from small examples without supervision. In *Proc. AAAI 2021*, 11801–11808.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *Proc. ICML 2017*, 1263–1272.

Grohe, M. 2021. The logic of graph neural networks. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 1–17.

Hamilton, W. 2020. *Graph Representation Learning*, volume 14 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proc. AAAI 2007*, 1007–1012.

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.

Horcík, R., and Sír, G. 2024. Expressiveness of graph neural networks in planning domains. In *Proc. ICAPS 2024*, 281–289.

Illanes, L., and McIlraith, S. A. 2019. Generalized planning via abstraction: Arbitrary numbers of objects. In *Proc. AAAI 2019*, 7610–7618.

Jiménez, S.; Segovia-Aguas, J.; and Jonsson, A. 2019. A review of generalized planning. *The Knowledge Engineering Review* 34:e5.

Khardon, R. 1999. Learning action strategies for planning domains. *Artificial Intelligence* 113:125–148.

Martín, M., and Geffner, H. 2004. Learning generalized policies from planning examples using concept languages. *Applied Intelligence* 20(1):9–19.

McKay, B. D., and Piperno, A. 2014. Practical graph isomorphism, ii. *Journal of Symbolic Computation* 60:94–112.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proc. AAAI 2019*, 4602–4609.

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In *Proc. IJCAI 2011*, 1983–1990.

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

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.

Rivlin, O.; Hazan, T.; and Karpas, E. 2020. Generalized planning with deep reinforcement learning. In *ICAPS Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL)*, 16–24.

Sangiorgi, D. 2012. *Introduction to Bisimulation and Coinduction*. Cambridge University Press.

Sanner, S., and Boutilier, C. 2009. Practical solution techniques for first-order MDPs. *Artificial Intelligence* 173(5-6):748–788.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20(1):61–80.

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

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.

Sievers, S.; Röger, G.; Wehrle, M.; and Katz, M. 2019. Theoretical foundations for structural symmetries of lifted PDDL tasks. In *Proc. ICAPS 2019*, 446–454.

Silver, T.; Dan, S.; Srinivas, K.; Tenenbaum, J. B.; Kaelbling, L. P.; and Katz, M. 2024. Generalized planning in PDDL domains with pretrained large language models. In *Proc. AAAI 2024*, 20256–20264.

Srivastava, S.; Immerman, N.; and Zilberstein, S. 2011. A new representation and associated algorithms for generalized planning. *Artificial Intelligence* 175(2):393–401.

Srivastava, S. 2022. Hierarchical decompositions and termination analysis for generalized planning. *jair* 77:1203–1236.

Srivastava, S. 2023. Hierarchical decompositions and termination analysis for generalized planning. *Journal of Artificial Intelligence Research* 77:1203–1236.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022a. Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits. In *Proc. ICAPS 2022*, 629–637.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022b. Learning generalized policies without supervision using GNNs. In *Proc. KR 2022*, 474–483.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2023. Learning general policies with policy gradient methods. In *Proc. KR 2023*, 647–657.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2024. Learning general policies for classical planning domains: Getting beyond c2. arXiv:2403.11734 [cs.AI].

Ståhlberg, S. 2023. Lifted successor generation by maximum clique enumeration. In *Proc. ECAI 2023*, 2194–2201.

Toyer, S.; Thiébaux, S.; Trevizan, F.; and Xie, L. 2020. ASNets: Deep learning for generalised planning. *Journal of Artificial Intelligence Research* 68:1–68.

Wang, C.; Joshi, S.; and Khardon, R. 2008. First order decision diagrams for relational MDPs. *Journal of Artificial Intelligence Research* 31:431–472.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *Proc. ICLR 2019*.

Yang, R.; Silver, T.; Curtis, A.; Lozano-Pérez, T.; and Kaelbling, L. P. 2022. PG3: policy-guided planning for generalized policy generation. In *Proc. IJCAI 2022*, 4686–4692.