Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2026)

On Sample-Efficient Generalized Planning via Learned Transition Models

Nitin Gupta*, Vishal Pallagani*, John A. Aydin, Biplav Srivastava

University of South Carolina
{niting@email., vishalp@mailbox., jaaydin@email., biplav.s@}sc.edu

Abstract

In this work, we formulate generalized planning as transition-model learning: a neural model approximates the successor-state function and generates plans by rolling out symbolic state trajectories. To achieve size-invariant generalization, we evaluate multiple state representations, including graph embeddings. Our results show that learning explicit transition models yields higher out-of-distribution success than action-sequence prediction in multiple domains, despite significantly fewer training instances and smaller models.

Code — github.com/ai4society/state-centric-gen-planning
Extended version — arxiv.org/abs/2602.23148

Introduction

Classical automated planning is defined over a state-transition system $\Sigma = \langle S, A, \gamma \rangle$, representing the set of states, the set of actions, and $\gamma : S \times A \to S$ the transition function respectively. A planning task $\Pi = \langle \Sigma, s_0, g \rangle$ consists of an initial state $s_0 \in S$ and a goal condition g, typically represented as a set of literals such that a state s satisfies g iff $g \subseteq s$. A solution is an action sequence $\pi = \langle a_1, \ldots, a_n \rangle$ such that $s_{t+1} = \gamma(s_t, a_t)$ and the final state s_n satisfies g. Generalized planning seeks strategies that solve families of such tasks sharing a common γ.

Recent learning-based approaches to generalized planning predominantly model the conditional distribution $p(\pi \mid \Pi)$, for example via an autoregressive factorization $p(\pi \mid \Pi) = \prod_{t=1}^{T} p(a_t \mid \Pi, a_{<t})$ or a policy $\pi_\theta(a_t \mid s_t, g)$, and directly predict action sequences from problem descriptions, as in Plansformer (Pallagani et al. 2022), PlanGPT (Rossetti et al. 2024b), and symmetry-aware Transformers (Fritzsche, Gestrin, and Seipp 2026). This action-centric formulation bypasses explicit modeling of γ: the evolving world state s_t is never directly represented, and long-horizon reasoning relies on implicit correlations between action tokens, leading to state drift in out-of-distribution regimes.

In this work, we model generalized planning as a transition-model learning problem. Rather than predicting the next action, we learn a goal-conditioned neural transition model $\mathcal{T}_\theta$ that predicts the successor state along a plan trajectory, i.e., given the current state s_t and goal g (or problem description Π), the model outputs $\hat{s}_{t+1} = \mathcal{T}_\theta(s_t, g)$. Plans are then obtained by rolling out the predicted state trajectory and recovering the corresponding actions via local symbolic search over applicable operators, by matching $\gamma(s_t, a)$ to $\hat{s}_{t+1}$. This formulation enforces explicit world-state evolution, enables successor validation, and constrains learning to respect frame axioms and causal effects. It is consistent with model-based world modeling in reinforcement learning (Ha and Schmidhuber 2018; Hafner et al. 2019), but applied here to symbolic generalized planning. For clarity, we refer to methods that directly predict actions (e.g., modeling $p(\pi \mid \Pi)$ or $\pi_\theta(a_t \mid s_t, g)$) as *action-centric* learning, and to methods that predict successor states via $\mathcal{T}_\theta(s_t, g)$ as *state-centric* learning. This usage is distinct from heuristic-based state-space learning in existing GP taxonomies (Chen, Trevizan, and Thiébaux 2025) and serves only to distinguish transition-model learning from direct action modeling.

A central challenge in learning $\mathcal{T}_\theta$ for GP is size invariance: the description of a state in S scales with the number of objects. To address this, we systematically evaluate multiple fixed-dimensional state representations, including fixed-size factored encodings and Weisfeiler–Leman (WL) graph embeddings (Chen, Trevizan, and Thiébaux 2024). WL embeddings map variable-sized relational states to fixed-length structural feature vectors, enabling compact models such as LSTMs (Hochreiter and Schmidhuber 1997) and XGBoost (Chen 2016) to generalize from small to large problem instances. We empirically show that relational WL features are critical for size-invariant and sample-efficient generalization. Our contributions, thus, are (i) a transition-model-based formulation of generalized planning via goal-conditioned successor-state prediction; (ii) a systematic evaluation of state representations for size-invariant and sample–efficient generalization; and (iii) an empirical demonstration that compact models achieve competitive GP performance with orders of magnitude fewer parameters and training instances than Transformer-based planners.

Related Work

Learning-based approaches to generalized planning seek policies or models that transfer across problem instances within a domain. Early neural GP work introduced rela-

*These authors contributed equally.

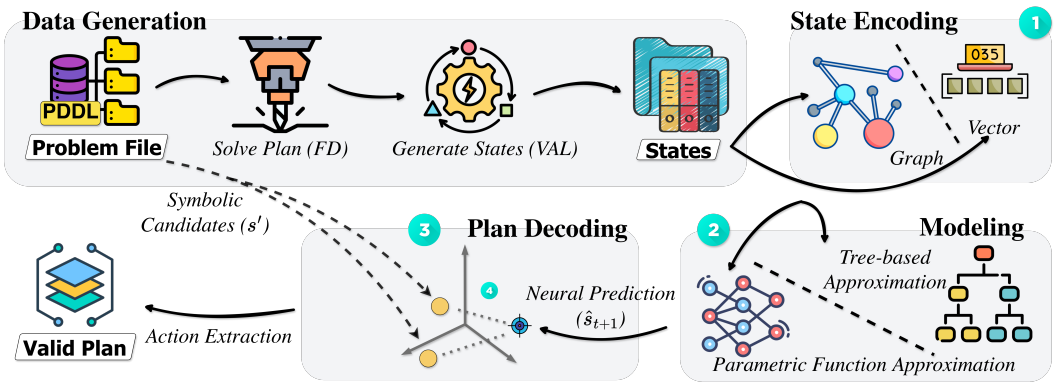

Figure 1: **State-Centric Generalized Planning Pipeline.** From a symbolic planning instance Π, executable plans are generated using a learned transition model. **(1) State Encoding:** Symbolic state–goal pairs (s_t, g) are mapped to fixed-dimensional embeddings $\phi(s_t)$ using either WL graph kernels or fixed-size factored vectors. **(2) Transition Modeling:** A parametric model (LSTM) or a non-parametric model (XGBoost) learns residual state transitions Δ_t to predict successor embeddings. **(3) Neuro-Symbolic Plan Decoding:** The predicted successor embedding $\hat{\phi}(s_{t+1})$ is matched against all valid symbolic successors $\mathrm{Succ}(s_t)$ induced by γ, and the nearest valid successor is selected to recover the executable action. This guarantees symbolic validity while enabling transition-model-based generalization.

tional inductive biases to enable size generalization, including Action Schema Networks (Toyer et al. 2018) and graph-based deep RL for Blocksworld (Rivlin, Hazan, and Karpas 2020), as well as finite-state policy representations (Ståhlberg, Bonet, and Geffner 2022). More recent work formulates GP as sequence prediction: Plansformer (Pallagani et al. 2022) and PlanGPT (Rossetti et al. 2024b) train Transformer architectures to directly generate action sequences, while symmetry-aware Transformers (Fritzsche, Gestrin, and Seipp 2026) introduce architectural and contrastive constraints to improve permutation invariance. However, such action-centric models do not explicitly learn transition dynamics and often exhibit state drift under distributional shift. In parallel, graph-based state representations have been extensively studied for learning heuristics and value functions in planning, including STRIPS-HGN (Shen, Trevizan, and Thiébaux 2020), GOOSE (Chen, Thiébaux, and Trevizan 2023), and domain-independent graph transformations (Chen and Thiébaux 2024). Recent work shows that Weisfeiler–Leman (WL) graph kernels combined with lightweight regressors can match or exceed GNN performance at far lower cost (Chen, Trevizan, and Thiébaux 2024; Chen 2025; Hao et al. 2025). Hybrid neuro-symbolic systems integrate learned components with symbolic solvers or validators, including LLM+P (Liu et al. 2023), symbolic validation for PlanGPT (Rossetti et al. 2024a), LLM-Modulo (Kambhampati et al. 2024), and SayCan for robotics (Ahn et al. 2022). Separately, model-based RL demonstrates the benefits of learning world models for planning (Ha and Schmidhuber 2018; Hafner et al. 2019). In contrast to prior action-sequence and heuristic-centric neural planners, our work adopts a transition-prediction formulation of generalized planning with size-invariant state representations, enabling sample-efficient and robust out-of-distribution generalization using compact models. For a detailed comparison, please refer to the extended work (Gupta et al. 2026).

Transition-Model-Based Generalized Planning

Figure 1 illustrates the complete pipeline of our approach, consisting of symbolic data generation, size-invariant state encoding, transition-model learning, and plan decoding via symbolic verification.

Generalized Planning Setup. A planning instance is $\Pi = \langle \mathcal{O}, \mathcal{P}, \mathcal{A}, s_0, g \rangle$, where $\mathcal{O}$ is a finite object set, $\mathcal{P}$ a predicate vocabulary, $\mathcal{A}$ a set of operators, $s_0 \subseteq \mathcal{P}(\mathcal{O})$ the initial state, and $g \subseteq \mathcal{P}(\mathcal{O})$ the goal condition. Operators induce a deterministic transition function $\gamma : S \times \mathcal{A} \to S$. A plan $\pi = \langle a_1, \ldots, a_T \rangle$ satisfies $s_{t+1} = \gamma(s_t, a_t)$ and $g \subseteq s_T$. Generalized planning seeks a single parameterized model trained on instances from $\mathcal{D}_{\text{train}}$ (small $|\mathcal{O}|$) that generalizes to $\mathcal{D}_{\text{test}}$ with much larger $|\mathcal{O}|$.

Size-Invariant State Representation. Each state-goal pair (s, g) is encoded as a relational instance graph $G_{s,g}$ and embedded using k iterations of WL color refinement. Node color histograms yield a fixed-dimensional embedding $\phi(s, g) \in \mathbb{R}^D$, where D depends only on the domain and is independent of $|\mathcal{O}|$. We overload notation and write $\phi(s)$ and $\phi(g)$ for the state and goal components, respectively. The resulting representation is permutation-invariant, size-invariant, and as expressive as 1-WL message-passing GNNs while enabling lightweight downstream models.

State-Centric Transition-Model Learning. Rather than learning a policy $\pi_\theta(a_t \mid s_t, g)$, we learn a neural transition model $f_\theta : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D$ that predicts state updates in embedding space. We denote this embedding-space transition model as f_θ; the conceptual model $\mathcal{T}_\theta$ from the introduction is realized as $\mathcal{T}_\theta(s_t, g) \approx \phi^{-1}(\phi(s_t) + f_\theta(\phi(s_t), \phi(g)))$, where the inverse is approximated via nearest-neighbor decoding. To exploit the sparsity of STRIPS-style transitions, where most predicates remain unchanged, we adopt a residual formulation: $\hat{\phi}(s_{t+1}) = \phi(s_t) + f_\theta(\phi(s_t), \phi(g))$ where f_θ predicts a delta vector Δ_t. This explicitly encodes frame axioms and improves sample efficiency, particularly for non-

Algorithm 1: *State-Centric* GP with Plan Decoding

Require: Initial state s_0, goal g, operators $\mathcal{A}$, learned model f_θ, embedding ϕ
Ensure: Valid plan π

$t \leftarrow 0, \pi \leftarrow \langle\rangle, s_t \leftarrow s_0$
while $g \not\subseteq s_t$ **do**
  $\mathbf{v}_t \leftarrow \phi(s_t) + f_\theta(\phi(s_t), \phi(g))$
  $\mathrm{Succ}(s_t) \leftarrow \{\gamma(s_t, a) \mid a \in \mathcal{A},\ a \text{ applicable in } s_t\}$
  $s_{t+1} \leftarrow \arg\min_{s' \in \mathrm{Succ}(s_t)} \|\phi(s') - \mathbf{v}_t\|_2$
  $a_t \leftarrow$ unique a such that $\gamma(s_t, a) = s_{t+1}$
  π.append(a_t), $s_t \leftarrow s_{t+1}, t \leftarrow t + 1$
end while
return π

sequential models. We train by minimizing the squared error over expert trajectories: $\mathcal{L} = \sum_t \|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$.

Plan Decoding via Neuro-Symbolic Verification. At test time, the true symbolic state s_t is maintained throughout execution. Given s_t, the transition model produces a target embedding $\mathbf{v}_t = \phi(s_t) + f_\theta(\phi(s_t), \phi(g))$. Using the symbolic operators, we enumerate all valid successors $\mathrm{Succ}(s_t) = \{\gamma(s_t, a) \mid a \in \mathcal{A},\ a \text{ applicable in } s_t\}$ and select the successor whose embedding is closest to the neural prediction: $s_{t+1} = \arg\min_{s' \in \mathrm{Succ}(s_t)} \|\phi(s') - \mathbf{v}_t\|_2$. The executed action is the unique a satisfying $\gamma(s_t, a) = s_{t+1}$, which is well-defined under deterministic operators. This decoding step guarantees symbolic validity at every timestep and performs online correction of neural prediction errors. The procedure terminates when $g \subseteq s_t$. The full planning algorithm is summarized in Algorithm 1.

Experimental Setup

We evaluate whether learning an explicit transition model enables sample-efficient and size-invariant generalized planning. Our experiments assess (i) extrapolative out-of-distribution (OOD) generalization to large instances, (ii) whether parametric function approximation is necessary for learning transition dynamics, and (iii) the impact of size-invariant state representations. Additional information is presented in the extended version (Gupta et al. 2026).

Domains and Data. We evaluate on 4 IPC benchmark domains: *Blocksworld*, *Gripper*, *Logistics*, and *VisitAll*. Problem instances are sourced from the Symmetry-Aware Transformer repository, following the same data splits for fair comparison. Symbolic plans are generated using Fast Downward (Helmert 2006) with the landmark-cut heuristic, and complete state trajectories are reconstructed using VAL (Howey, Long, and Fox 2004). Data is partitioned into four splits by object count: *Training* (small instances, e.g., 4–7 blocks), *Validation* (similar or slightly larger sizes, held out), *Interpolation* (unseen configurations within the training size range), and *Extrapolation* (strictly larger than any training instance, e.g., 9–17 blocks). Extrapolation is the primary evaluation axis for size-invariant generalized planning.

State Representations. We compare WL graph embeddings (Chen and Thiébaux 2024), which are permutation- and size-invariant, against Fixed-Size Factored (FSF) encodings. FSF encodings represent states as fixed-dimensional vectors with pre-assigned object slots, deliberately omitting the relational structure of WL to isolate the contribution of invariant representations to OOD generalization (Boutilier, Dearden, and Goldszmidt 2000; Guestrin et al. 2003).

Transition Models. We evaluate two transition-model classes: a parametric neural model (two-layer LSTM) and a tree-based, nonparametric regressor (XGBoost). The LSTM tests whether sequential memory is necessary for trajectory-level transition dynamics, while XGBoost tests whether a local approximation of the transition kernel suffices. This comparison isolates the role of temporal memory. Both models are trained in state-prediction and delta-prediction modes, as defined in previous sections.

Baselines. We compare against published results from Symmetry-Aware Transformers (SymT), which include results on PlanGPT, including applicability-filtered and regrounded variants. We further run inference with Plansformer on our test instances using its publicly released checkpoint; it was trained on Blocksworld but not on Gripper, Logistics, or VisitAll, so its zero-shot cross-domain performance is expected to be limited. PlanGPT results are taken directly from Fritzsche, Gestrin, and Seipp (2026), who train and evaluate PlanGPT on the same data splits and counts as SymT. As a symbolic upper bound, we include Fast Downward with A* and the landmark-cut heuristic, which serves to compare learning paradigms rather than to benchmark against optimized classical planners.

Inference and Metrics. At test time, plans are generated using the neuro-symbolic decoding procedure in Algorithm 1 with beam width 3 and a upper-bound dynamic horizon of $\max(100,\ 10 \cdot |\mathcal{O}|)$ steps. Performance is measured by *satisficing success rate*, i.e., the fraction of instances for which a generated plan is valid under the transition model and reaches a goal state within the horizon limit, as verified by VAL. We additionally report changes in satisficing success across successive rollouts (seeds) to quantify stability under repeated decoding.

Results and Analysis

Table 1 reports satisficing-plan success rates across all domains, splits, representations, model classes, and prediction modes. The primary empirical finding is that explicit transition-model learning combined with size-invariant relational representations yields stronger or matching extrapolation than action-centric prediction in domains with locally factored domains, while remaining insufficient for the Logistics benchmark under size extrapolation. The extended work (Gupta et al. 2026) provides additional details.

Comparison with action-centric planners. Under strict extrapolation, Plansformer and all PlanGPT variants achieve 0.00 success across all four domains. Plansformer further exhibits 0.00 on non-*Blocksworld* domains since they are not present in its training data. SymT attains non-zero extrapolation in *Blocksworld* (0.13), *Gripper* (0.79), and *VisitAll* (0.64), but fails in *Logistics*. The best state-centric models exceed SymT in *Blocksworld* (WL-XGB delta 0.50 vs. 0.13) and *VisitAll* (1.00 vs. 0.64), while SymT remains superior in *Gripper* extrapolation (0.79 vs. 0.42). Notably, these gains are obtained using compact transition models trained

Domain	Split	FD	Plansf.	PlanGPT*			SymT^{E}*	SymT^{ED}*			WL-LSTM		WL-XGB		FSF-LSTM		FSF-XGB	
				greedy	appl.	regr.	greedy	greedy	appl.	regr.	state	delta	state	delta	state	delta	state	delta
Blocks	Val.	1.00	**1.00**$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.44$_{\pm 0.16}$	0.67$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.67$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$
	Interp.	1.00	**1.00**$_{\pm 0.00}$	0.56$_{\pm 0.16}$	0.56$_{\pm 0.16}$	0.00$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.67$_{\pm 0.27}$	0.89$_{\pm 0.16}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.11$_{\pm 0.16}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$
	Extrap.	0.60	0.10$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.05$_{\pm 0.07}$	0.07$_{\pm 0.02}$	0.13$_{\pm 0.05}$	0.00$_{\pm 0.00}$	0.10$_{\pm 0.07}$	0.15$_{\pm 0.07}$	0.25$_{\pm 0.00}$	**0.50**$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$
Gripper	Val.	1.00	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.17$_{\pm 0.24}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.67$_{\pm 0.47}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.17$_{\pm 0.24}$	0.50$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$
	Interp.	1.00	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.44$_{\pm 0.16}$	0.00$_{\pm 0.00}$	0.89$_{\pm 0.16}$	0.67$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.67$_{\pm 0.47}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.44$_{\pm 0.31}$	0.22$_{\pm 0.31}$	0.67$_{\pm 0.00}$	0.00$_{\pm 0.00}$
	Extrap.	0.13	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.02$_{\pm 0.03}$	0.00$_{\pm 0.00}$	0.15$_{\pm 0.06}$	**0.79**$_{\pm 0.16}$	0.42$_{\pm 0.16}$	0.25$_{\pm 0.31}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.04$_{\pm 0.03}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$
VisitAll	Val.	1.00	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.14$_{\pm 0.12}$	0.00$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.33$_{\pm 0.09}$	0.93$_{\pm 0.04}$	0.99$_{\pm 0.02}$	**1.00**$_{\pm 0.00}$	0.79$_{\pm 0.29}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.08$_{\pm 0.07}$	0.53$_{\pm 0.02}$	0.12$_{\pm 0.00}$	0.92$_{\pm 0.00}$
	Interp.	1.00	0.00$_{\pm 0.00}$	0.05$_{\pm 0.04}$	0.67$_{\pm 0.18}$	0.41$_{\pm 0.22}$	**1.00**$_{\pm 0.00}$	0.87$_{\pm 0.01}$	0.99$_{\pm 0.01}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.99$_{\pm 0.01}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.57$_{\pm 0.02}$	0.39$_{\pm 0.03}$	0.86$_{\pm 0.00}$	0.95$_{\pm 0.00}$
	Extrap.	0.50	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.02$_{\pm 0.02}$	0.00$_{\pm 0.00}$	0.42$_{\pm 0.11}$	0.00$_{\pm 0.00}$	0.15$_{\pm 0.05}$	0.64$_{\pm 0.12}$	0.72$_{\pm 0.13}$	0.62$_{\pm 0.39}$	0.15$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.07$_{\pm 0.01}$	0.01$_{\pm 0.00}$	0.16$_{\pm 0.00}$
Logistics	Val.	1.00	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.08$_{\pm 0.12}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	**0.25**$_{\pm 0.35}$	0.08$_{\pm 0.12}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$
	Interp.	1.00	0.00$_{\pm 0.00}$	0.07$_{\pm 0.05}$	0.44$_{\pm 0.09}$	0.19$_{\pm 0.14}$	0.11$_{\pm 0.00}$	0.22$_{\pm 0.31}$	0.26$_{\pm 0.29}$	0.22$_{\pm 0.31}$	**0.85**$_{\pm 0.14}$	0.33$_{\pm 0.16}$	0.11$_{\pm 0.00}$	0.11$_{\pm 0.00}$	0.19$_{\pm 0.10}$	0.11$_{\pm 0.00}$	0.44$_{\pm 0.00}$	0.11$_{\pm 0.00}$
	Extrap.	0.26	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$

Table 1: Coverage rates (%) across all configurations. Values are reported as $Mean_{\pm Std}$. Best result per row (over all non-FD configurations) is bolded and second best is underlined. Light gray columns denote our state-centric implementations. FD = Fast Downward (60s timeout). *Results from Fritzsche, Gestrin, and Seipp (2026).

on unaugmented state trajectories, in contrast to SymT, which relies on large Transformer architectures and extensive symmetry-based data augmentation. This indicates that, under appropriate relational abstractions, explicit transition learning can achieve competitive extrapolation with substantially lower model and data complexity.

Effect of size-invariant relational representations. Across all domains, FSF-based encodings yield negligible extrapolation performance: *Blocksworld* (0.00), *Gripper* (0.00), *VisitAll* ($\leq$ 0.13), and *Logistics* (0.00). In contrast, WL-based models achieve strictly positive extrapolation in three domains. In *Blocksworld*, WL-XGB (delta) reaches 0.50 compared to 0.00 for all FSF variants. In *VisitAll*, WL-XGB (delta) reaches 1.00 while FSF-XGB (delta) reaches only 0.16. This establishes that extrapolation beyond the training object bound requires a permutation- and size-invariant abstraction $\phi : S \to \mathbb{R}^D$. Fixed-slot encodings restrict the hypothesis class to a bounded object universe and therefore fail when $|\mathcal{O}|_{\text{test}} > |\mathcal{O}|_{\text{train}}$.

Effect of residual transition modeling. For tree-based models, residual parameterization consistently improves extrapolation. In *Blocksworld*, WL-XGB improves from 0.25 (state) to 0.50 (delta). In *VisitAll*, it improves from 0.15 to 1.00. This behavior is consistent with STRIPS transition semantics, $\gamma(s, a) = (s \setminus \text{Del}(a)) \cup \text{Add}(a)$, which induces sparse state differences. The delta formulation constrains learning to the subspace of changed fluents, reducing regression variance for non-parametric models. For LSTM, the effect is domain dependent: delta improves *Blocksworld* extrapolation (0.03 $\to$ 0.15) but degrades *Gripper* (0.25 $\to$ 0.17), indicating interaction between residual bias and recurrent state memory.

Sequential versus non-sequential transition learning. Comparing WL-LSTM and WL-XGB isolates the role of temporal memory. In *Gripper* extrapolation, WL-LSTM (state) attains 0.42 while both XGB variants remain at 0.00, indicating that under the chosen abstraction the induced transition kernel $P(\phi(s_{t+1}) \mid \phi(s_t), \phi(g))$ is not well-approximated by a purely local regressor. In contrast, in *Blocksworld* and *VisitAll*, WL-XGB (delta) outperforms WL-LSTM: *Blocksworld* (0.50 vs. 0.15) and *VisitAll* (1.00 vs. 0.72), indicating that the Markovian assumption suffices under relational abstraction in these domains.

Limitations under hierarchical causal coupling. All learned models, including all state-centric variants, achieve 0.00 extrapolation in *Logistics*. Even Fast Downward degrades from 1.00 (validation) to 0.26 (extrapolation) under a 60-second timeout, reflecting the exponential state-space growth of extrapolation instances. The *Logistics* domain exhibits deep multi-layer causal coupling across object types and transport modalities, which is not preserved under local successor matching. This identifies a concrete structural limitation of one-step neural transition prediction under strict size extrapolation for this domain.

Summary of empirical findings. The results support three data-grounded conclusions: (i) size-invariant relational representations are necessary for extrapolation beyond training object bounds; (ii) residual (delta) modeling significantly improves non-parametric transition learning in sparse STRIPS domains; and (iii) the necessity of sequential memory in transition learning is domain dependent. Transition-model learning alone, however, remains insufficient for hierarchical domains under strict extrapolation.

Conclusion and Future Work

We presented a state-centric formulation of generalized planning in which models learn to predict successor states rather than action sequences. When combined with size- and permutation-invariant relational embeddings, this approach enables compact transition models ($\sim$1–2M parameters, no data augmentation) to achieve strong extrapolation performance in locally factored domains, matching or exceeding Transformer baselines ($\sim$25–220M parameters) that rely on orders of magnitude more data and parameters. Empirically, our results show that explicit transition modeling provides a stronger inductive bias for extrapolation than architectural scale alone, though limitations remain in domains with hierarchical and long-range dependencies. The neuro-symbolic decoding interface further improves robustness by enforcing symbolic validity at every planning step. Future work will target hierarchical and long-range dependency domains, where one-step state prediction fails under strict extrapolation. We will extend the framework to multi-step or abstract transitions while preserving symbolic verification.

Acknowledgments

This work is partially supported by NSF Awards #2454027 and NAIRR250014, and Faculty Award by JP Morgan Research.

References

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2000. Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1-2): 49–107.

Chen, D.; and Thiébaux, S. 2024. Graph learning for numeric planning. *Advances in Neural Information Processing Systems*, 37: 91156–91183.

Chen, D. Z. 2025. Weisfeiler-Leman Features for Planning: A 1,000,000 Sample Size Hyperparameter Study. *arXiv preprint arXiv:2508.18515*.

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. 2024. Return to tradition: Learning reliable heuristics with classical machine learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 68–76.

Chen, D. Z.; Trevizan, F.; and Thiébaux, S. 2025. Learning for Generalised Planning. Tutorial, International Conference on Automated Planning and Scheduling (ICAPS). https://l4p-tutorial.github.io/slides.pdf.

Chen, T. 2016. XGBoost: A Scalable Tree Boosting System. *Cornell University*.

Fritzsche, M.; Gestrin, E.; and Seipp, J. 2026. Symmetry-Aware Transformer Training for Automated Planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(43): 36236–36244.

Guestrin, C.; Koller, D.; Parr, R.; and Venkataraman, S. 2003. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19: 399–468.

Gupta, N.; Pallagani, V.; Aydin, J. A.; and Srivastava, B. 2026. On Sample-Efficient Generalized Planning via Learned Transition Models. *arXiv preprint arXiv:2602.23148*.

Ha, D.; and Schmidhuber, J. 2018. World models. *arXiv preprint arXiv:1803.10122*, 2(3).

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, 2555–2565. PMLR.

Hao, M.; Chen, D. Z.; Trevizan, F.; and Thiébaux, S. 2025. Effective data generation and feature selection in learning for planning. In *European Conference on Artificial Intelligence (ECAI-25)*.

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

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Howey, R.; Long, D.; and Fox, M. 2004. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *16th IEEE International Conference on Tools with Artificial Intelligence*, 294–301. IEEE.

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L. P.; and Murthy, A. B. 2024. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Pallagani, V.; Muppasani, B.; Murugesan, K.; Rossi, F.; Horesh, L.; Srivastava, B.; Fabiano, F.; and Loreggia, A. 2022. Plansformer: Generating symbolic plans using transformers. *arXiv preprint arXiv:2212.08681*.

Rivlin, O.; Hazan, T.; and Karpas, E. 2020. Generalized planning with deep reinforcement learning. *arXiv preprint arXiv:2005.02305*.

Rossetti, N.; Tummolo, M.; Gerevini, A. E.; Olivato, M.; Putelli, L.; and Serina, I. 2024a. Enhancing GPT-based planning policies by model-based plan validation. In *International Conference on Neural-Symbolic Learning and Reasoning*, 328–337. Springer.

Rossetti, N.; Tummolo, M.; Gerevini, A. E.; Putelli, L.; Serina, I.; Chiari, M.; and Olivato, M. 2024b. Learning general policies for planning through GPT models. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 500–508.

Shen, W.; Trevizan, F.; and Thiébaux, S. 2020. Learning domain-independent planning heuristics with hypergraph networks. In *Proc. of the International Conference on Automated Planning and Scheduling*, volume 30, 574–584.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022. Learning generalized policies without supervision using gnns. *arXiv preprint arXiv:2205.06002*.

Toyer, S.; Trevizan, F.; Thiébaux, S.; and Xie, L. 2018. Action schema networks: Generalised policies with deep learning. In *Proc. of the AAAI Conference on Artificial Intelligence*, volume 32.