Learning Hierarchical Domain Models through Environment Interaction

Claudius Kienle[1], Benjamin Alt[2], Oleg Arenz[1] and Jan Peters[1]

I. INTRODUCTION

Autonomous agents must solve long-horizon tasks in open-domain settings while adapting to varying environments. Planning with explicit domain models provides structured, interpretable solutions, but diverse settings make single universal domain models infeasible. Large Language Models (LLMs) can generate context-specific domain models from limited data [2], [3], [4], [5], but their unreliable outputs often contain hallucinated preconditions and effects. Recent work verifies LLM-generated domains through self-critique [2], classical tools [6], [7], [8], [9], or by generating multiple candidates evaluated via similarity metrics or environment interaction [3], [10], [11]. However, prior approaches require extensive human feedback or verbose skill annotations, limiting practical deployment.

We propose LODGE, a fully autonomous framework for learning hierarchical Planning Domain Definition Language (PDDL) domain models with LLMs and environment grounding. To manage complexity, LODGE learns a hierarchical domain model, where abstract operators are decomposed into reusable subdomains. LODGE ensures consistency across hierarchical levels. It autonomously invents PDDL predicates and learns Python-based classifiers from pseudo-labeled interactions. During domain model learning, a *recovery module* detects misalignments between the planner's symbolic state and the observed symbolic state of the environment. We propose a method to resolve misalignments between the two by refining the domain model, resulting in the first automated domain learning method for open-world domains without human feedback. Our contributions are: **(1)** Autonomous domain learning from planning feedback via a recovery module that refines operators based on few environment interactions. **(2)** Hierarchical domain models that split large domains hierarchically, reducing learning complexity and enabling partial re-planning. **(3)** Predicate invention with online classifier learning from pseudo-labeled data to extend the symbolic state autonomously.

II. NOTATION AND PROBLEM STATEMENT

An environment f has continuous state x with objects $\mathcal{O}$ and executable skills Π. A domain model $\mathcal{D} = \langle \Psi, \Omega \rangle$ consists of predicates Ψ encoding discrete properties (e.g., *grasps(?obj)*) and operators Ω with preconditions and effects. Each *state-based* predicate ψ has a classifier $c_\psi : \mathcal{O} \times \mathcal{X} \to \{\text{true}, \text{false}\}$ that evaluates ground atoms (predicates with objects) in state x. An operator $\omega \in \Omega$ instantiated with objects forms an action a. Actions map to one or more skills; hierarchy levels use superscripts (e.g., $\mathcal{D}^{(0)}$). Given a natural language task I and skill library Π, we learn hierarchical domain models including operators, predicates, and classifiers, assuming deterministic, fully observable environments.

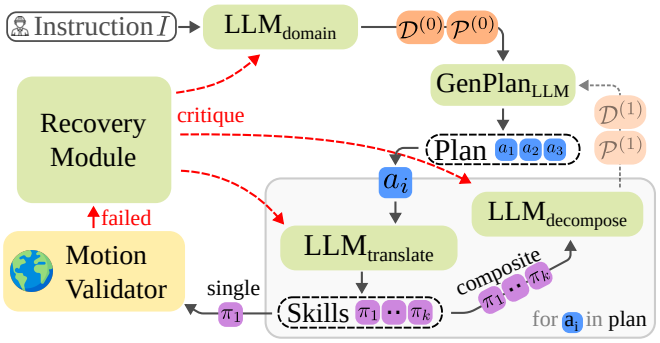

Fig. 1: LODGE: Learning hierarchical domain models with environment grounding.

III. LEARNING HIERARCHICAL DOMAIN MODELS BY PLANNING

LODGE learns hierarchical PDDL domain models by alternating between domain generation, planning, and environment grounding (Fig. 1). Given instruction I, skill library Π, start state x_0, and optional initial predicates Ψ_{init}, LODGE uses an $\text{LLM}_{\text{domain}}$ to induce a preliminary domain $\mathcal{D}^{(0)}$ and goal $g^{(0)}$. The domain includes operators and predicates.

LODGE generates high-level abstract operators instead of one operator per skill. The abstract operators are later progressively decomposed into subdomains. This reduces complexity at each level, improves LLM accuracy, and enables a coarse symbolic state at higher levels. Given the proposed domain $\mathcal{D}^{(0)}$ and goal $g^{(0)}$, LODGE generates a candidate plan $a^{(0)}_{1:m}$ using a classical planner.

For each action $a^{(0)}_i = \omega(o_{1:n})$ in plan $a^{(0)}_{1:m}$, LODGE maps operator ω to a skill sequence with an LLM. Single-skill operators become leaf nodes; multi-skill operators are decomposed. To decompose an operator ω, we define a subproblem from the state transition caused by ω and generate a subdomain $\mathcal{D}^{(1)}$ similiar to $\text{LLM}_{\text{domain}}$. A learned subdomain can later be reused for all instantiations of an operator without additional LLM calls.

*This work was supported by the German Federal Ministry of Education and Research (project RobInTime, grant 01IS25002B).

[1]Intelligent Autonomous Systems Group, TU Darmstadt, Germany name.surname@tu-darmstadt.de

[2]AICOR Institute for Artificial Intelligence, University of Bremen, Germany name.surname@uni-bremen.de

This work is based on our previously published paper: [1].

IV. ALIGNING LEVELS OF HIERARCHICAL DOMAIN MODELS

Hierarchical decomposition requires preserving knowledge and ensuring alignment between levels. LODGE retains predicates Ψ and objects $\mathcal{O}$ across all lower levels while encapsulating lower-level predicates from upper levels. When decomposing operator ω with effects $\text{eff}(a)$ into subplan $a_{1:k}$ with joint effects $\text{eff}(a_{1:k})$, misalignments occur when $\text{eff}(a_{1:k}) \supset \text{eff}(a)$. We differentiate predicates used in $\text{eff}(a_{1:k}) \setminus \text{eff}(a)$ into *overshoots* (affecting objects used in a) or *side effects* (affecting objects not used in a). Such misalignments can break the correctness of hierarchical plans. Consequently, we detect misalignments by comparing $\text{eff}(a_{1:k})$ with $\text{eff}(a)$ and prompt $\text{LLM}_{\text{decomp}}$ to realign the abstract operator's effects with $\text{eff}(a_{1:k})$.

V. INVENTING PREDICATES AND LEARNING CLASSIFIERS

LODGE autonomously invents predicates during domain learning and generates Python-based classifiers for state-based predicates using an LLM [12]. These Python classifiers may include hyperparameters (e.g., tolerances) with defaults given by the LLM. To evaluate the correctness of the generated classifiers, we collect environment transitions (x_i, π_i, x_i') during domain learning and generate pseudo-labels s_i' with a VLM [13]. For each classifier c_ψ, LODGE computes F1 scores for all ground atoms and refines based on the lowest F1 score. When F1 $< \tau_{\text{hp}}$ (0.9), we optimize the classifiers' hyperparameters with gradient-free random search to find robust hyperparameters closest to the LLM defaults. When F1 $< \tau_{\text{llm}}$ (0.6), we invoke a LLM self-refinement with the misclassified samples to correct the classifier implementation.

VI. VERIFYING PLANS AND RECOVERING FROM MODEL ERRORS

LODGE executes each planned skill in the environment to compare the planner's symbolic state transition with the observed environment transition. For leaf action $a = \omega(o_{1:n})$ with mapped skill π and current state x_t, LODGE first verifies that the preconditions hold ($\text{pre}(\omega) \subseteq c_{\underline{\Psi}}(x_t)$). It then executes π to obtain x_{t+1}, and verifies that the operator effects hold in x_{t+1}. We propose a *recovery module* that is triggered if the effects do not hold. The *recovery module* analyzes the misalignment and receives past LLM interactions of domain model learning. It identifies which operator to adapt, and starts the refinement, followed by retracting to the affected hierarchy level to continue domain learning.

VII. EXPERIMENTS

We evaluate LODGE on six domains consisting of (1) two IPC domains to learn operators with little predefined information and few environment interactions (2) and four robotics domains to learn complete domain models (predicates, classifiers, operators). We evaluate the generated domains using the Exploration Walk metric [3] and compare LODGE to Guan et al. [2] (no human feedback) and Mahdavi et al. [3] on GPT 4.1 mini. Table I shows all evaluations.

Env	Guan	Mahdavi	LODGE (ours)
Logistics	0.90 / 0.68	0.99 / 0.98	**1.00** / **1.00**
Household	0.06 / 0.00	0.20 / 0.03	**0.69** / **0.44**
FB Lamp	0.26 / 0.06	0.23 / 0.17	**0.78** / **0.81**
FB Round Table†	0.21 / 0.15	0.11 / 0.00	**0.71** / **1.00**
FB Square Table	0.16 / 0.00	0.20 / 0.00	**0.53** / **0.20**
FB Stool & Desk‡	0.11 / 0.00	0.16 / 0.00	**0.52** / **0.14**

TABLE I: Avg@3 (Exploration Walk/Tasks solved) for six domains. †LODGE reuses and refines the domain model learned from FB Lamp. ‡LODGE reuses and generalizes the domain model learned from FB Square Table.

A. Learning Operators for IPC Domains

We evaluate operator learning on the IPC domains *household* and *logistics* (see [2]). LODGE learns operators for the first task in a domain and refines it iteratively for the following tasks. We limit environment interactions to 10 and LLM self-refinements to 20 per task. Table I shows that LODGE outperforms the baseline for the more complex *household* domain, achieving 44% task success.

B. Learning Domain Models for Robotics Domains

We learn the complete domain model for four FurnitureBench domains [14], providing only the goal predicate *assembled(?obj1, ?obj2)* initially. All methods learn the domains' predicates and operators, while LODGE additionally learns predicate classifiers while solving the task.

Table I shows domain model quality and task success for *lamp* and *square-table* assembly, plus generalization to similar tasks (*round-table*, *desk*, *stool*). While Mahdavi [3] performs well on IPC domains, it struggles with FurnitureBench's complexity. LODGE learns accurate domain models and assembles the lamp in 2/3 seeds. The hierarchical domain enables LODGE to invent high-level predicates (*aligned*, *holding*) at the top level and fine-grained predicates (*gripper-around-part*, *touching*) during decomposition.

We compare our method of predicate classifier learning to InterPreT [12] and Pix2Pred [13] for the *lamp* domain. Pix2Pred achieves an average F1 score of 0.84 using VLM-based classification, while InterPreT's Python classifiers show variable accuracy (F1 score of 0.71) due to coding errors or unsuitable hyperparameters. LODGE's refinement using pseudo-labels achieves an F1 score of 1.00, outperforming both baselines.

VIII. CONCLUSION

We presented LODGE, a framework for learning hierarchical PDDL domain models through hierarchical decomposition, autonomous predicate invention with classifier learning, and environment-grounded operator refinement. Our evaluation on six domains demonstrates higher task success and domain model accuracy than existing methods while eliminating the need for human feedback or skill annotations. This enables practical domain learning for real-world robotic applications in open, unstructured environments.

REFERENCES

[1] C. Kienle, B. Alt, O. Arenz, and J. Peters, "Learning Hierarchical Domain Models Through Environment-Grounded Interaction," Oct. 2025, arXiv:2505.13497 [cs] version: 3. [Online]. Available: http://arxiv.org/abs/2505.13497

[2] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati, "Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 79 081–79 094, Dec. 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/f9f54762cbb4fe4dbffdd4f792c31221-Abstract-Conference.html

[3] S. Mahdavi, R. Aoki, K. Tang, and Y. Cao, "Leveraging Environment Interaction for Automated PDDL Translation and Planning with Large Language Models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 38 960–39 008, Dec. 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/hash/44af065477781e7f8a8589b14a62c489-Abstract-Conference.html

[4] M. Tantakoun, X. Zhu, and C. Muise, "LLMs as Planning Modelers: A Survey for Leveraging Large Language Models to Construct Automated Planning Models," Mar. 2025, arXiv:2503.18971 [cs]. [Online]. Available: http://arxiv.org/abs/2503.18971

[5] J. Oswald, K. Srinivas, H. Kokel, J. Lee, M. Katz, and S. Sohrabi, "Large Language Models as Planning Domain Generators," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 34, pp. 423–431, May 2024. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/31502

[6] P. Smirnov, F. Joublin, A. Ceravola, and M. Gienger, "Generating consistent PDDL domains with Large Language Models," Apr. 2024, arXiv:2404.07751 [cs]. [Online]. Available: http://arxiv.org/abs/2404.07751

[7] A. Ishay and J. Lee, "LLM+AL: Bridging Large Language Models and Action Languages for Complex Reasoning About Actions," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 23, pp. 24 212–24 220, Apr. 2025. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/34597

[8] G. Chen, L. Yang, R. Jia, Z. Hu, Y. Chen, W. Zhang, W. Wang, and J. Pan, "Language-Augmented Symbolic Planner for Open-World Task Planning," July 2024, arXiv:2407.09792 [cs]. [Online]. Available: http://arxiv.org/abs/2407.09792

[9] Y. Ding, X. Zhang, S. Amiri, N. Cao, H. Yang, A. Kaminski, C. Esselink, and S. Zhang, "Integrating action knowledge and LLMs for task planning and situation handling in open worlds," *Autonomous Robots*, vol. 47, no. 8, pp. 981–997, Dec. 2023. [Online]. Available: https://link.springer.com/10.1007/s10514-023-10133-5

[10] Z. Yu, Y. Yuan, T. Z. Xiao, F. F. Xia, J. Fu, G. Zhang, G. Lin, and W. Liu, "Generating Symbolic World Models via Test-time Scaling of Large Language Models," May 2025, arXiv:2502.04728 [cs]. [Online]. Available: http://arxiv.org/abs/2502.04728

[11] M. Hu, T. Chen, Y. Zou, Y. Lei, Q. Chen, M. Li, Y. Mu, H. Zhang, W. Shao, and P. Luo, "Text2World: Benchmarking Large Language Models for Symbolic World Model Generation," Feb. 2025, arXiv:2502.13092 [cs]. [Online]. Available: http://arxiv.org/abs/2502.13092

[12] M. Han, Y. Zhu, S.-C. Zhu, Y. N. Wu, and Y. Zhu, "InterPreT: Interactive Predicate Learning from Language Feedback for Generalizable Task Planning," May 2024, arXiv:2405.19758 [cs]. [Online]. Available: http://arxiv.org/abs/2405.19758

[13] A. Athalye, N. Kumar, T. Silver, Y. Liang, J. Wang, T. Lozano-Pérez, and L. P. Kaelbling, "From Pixels to Predicates: Learning Symbolic World Models via Pretrained Vision-Language Models," June 2025, arXiv:2501.00296 [cs]. [Online]. Available: http://arxiv.org/abs/2501.00296

[14] M. Heo, Y. Lee, D. Lee, and J. J. Lim, "FurnitureBench: Reproducible real-world benchmark for long-horizon complex manipulation," *The International Journal of Robotics Research*, p. 02783649241304789, 2023, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/02783649241304789