CLIMB: Language-Guided Continual Learning for Task Planning with Iterative Model Building

Walker Byrnes[1,2], Miroslav Bogdanovic[4], Avi Balakirsky[3], Stephen Balakirsky[2], Animesh Garg[1,4,5]

***Abstract*— Intelligent and reliable task planning is a core capability for generalized robotics, requiring a descriptive domain representation that sufficiently models all object and state information for the scene. We present CLIMB, a continual learning framework for robot task planning that leverages foundation models and execution feedback to guide domain model construction. CLIMB can build a model from a natural language description, learn non-obvious predicates while solving tasks, and store that information for future problems. We demonstrate the ability of CLIMB to improve performance in common planning environments compared to baseline methods. We also develop *BlocksWorld++* domain, a simulated environment with an easily usable real counterpart, together with a curriculum of tasks with progressing difficulty for evaluating continual learning. Code and additional details for this system can be found at https://plan-with-climb.github.io/.**

I. Introduction

For decades roboticists have pursued the aim of generalized, flexible robots that can solve complex tasks in novel environments. Recent advances in foundation models [1] have shown promising results for their use in world modeling [2–4], task planning [5–8], and motion planning [9–11]. These works leverage the extensive background knowledge present in the foundation model's pre-training to provide incomplete but useful solutions to these challenging tasks. Though incomplete, results can be further refined through repetition, prompt engineering, or post-processing to improve success rates.

While interest in fundamental research has been plentiful, foundation model-guided planners have yet to find regular use in practical application scenarios. This hesitance has been largely derived from the inconsistencies of output for many foundation models. Some research has shown that the direct application of foundational models for task planning yields subpar results [12], where critics argue foundation models provide an "approximate retrieval" of information and are incapable of explicit logical reasoning or planning. This observation suggests that a more sophisticated structure is required in order to leverage the extensive corpus of foundation models effectively.

In this paper, we present CLIMB, Continual Learning for Iteractive Model Building. CLIMB is a hybrid neuro-symbolic planning system that makes use of both foundation models and classical symbolic planners to achieve planning proficiency. Given the limitations of both classical search-based planners and end-to-end learning models, a neuro-symbolic approach is required to address complex planning problems reliably.

[1]Georgia Institute of Technology, [2]Georgia Tech Research Institute,[3]Ohio State University, [4]University of Toronto, [5]Nvidia
Correspondence to: animesh.garg@gatech.edu

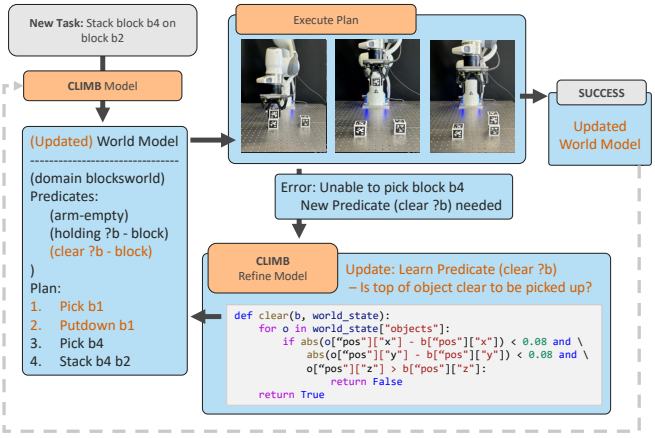

Fig. 1: CLIMB is able to predict common world constraints without ground truth PDDL and can learn domain-specific relationships through the execution of one or more tasks in the domain. The framework can self-improve by storing domain and predicate information across tasks.

Such a hybrid architecture leverages both the structure of symbolic planning and the learning capabilities and extensive knowledge base of foundation models effectively. Additionally, CLIMB incrementally builds a PDDL model of its operating environment while completing tasks, creating a set of world state predicates that function as a representation of the causal structure present in the environment. This continual learning approach enables CLIMB to solve types of problems it has previously encountered without the need to relearn task-specific information and endows it with the ability to expand its environment representation to novel problem formulations. We show CLIMB to be a moderately capable planner independently, but importantly we demonstrate its ability to self-improve, resulting in superior performance once a PDDL model has been established.

The contributions of our work are as follows:

1) We propose CLIMB for learning logical models of the world and accompanying grounding functions, starting from simple domain and task descriptions in natural language and learning and improving through interaction with the environment.
2) We evaluate CLIMB to generate sensible initial domain proposals, to improve through interaction and incremental world model building across several tasks, while simultaneously proposing and correcting grounding functions to connect the logical domain to the continous environments.
3) We propose BlocksWorld++with a curriculum of tasks to evaluate incremental logical world model-building capabilities, both in simulation and in the real world.

II. Related Work

A. Learning-based Planning in Robotics

Learning methods have long been used to model and represent the object and state interactions in robot environments. Such pursuits have utilized a range of methods including unsupervised clustering [13, 14], supervised learning from demonstration [15], and neural architectures [16]. The level of human setup and interaction significantly varies between approaches, with both heavily supervised interaction-based approaches [4] and fully-unsupervised systems [17] seeing use.

In addition to learning world relationships and predicates, the task of grounding continuous state observations to logical predicates remains a challenge. Both images [18] and synthesized state representations [19] have been used in evaluating world state representations. While both approaches have shown success, the former requires large quantities of annotated data while the latter necessitates extensive engineered systems that do not generalize easily.

B. General Planning Capabilities of LLMs

Recent advances in Large Language Models (LLMs) have shown promising results in their ability to complete logical reasoning tasks, decompose complex goals into sub-goals, and ingest large amounts of data. These capabilities are aided by research into prompting strategies and in-context learning [20–23], evaluating the best methods for interacting with these models. Chain-of-Thought [20], tree-of-thoughts [21], graph-of-thoughts [22], and ReAct [23] examine the relative performance of prompting structures for complex reasoning tasks. While LLMs can effectively generate reasonable unstructured natural language outputs, they have demonstrated poor performance at structured and constrained tasks including planning [12] without additional structure. These limitations also occur in embodied environments. SayCan [24] and Reflexion [25] propose action embedding and verbal RL to surprising success on text-based tasks. Unstructured natural language tasks do not easily translate into most embodied robot paradigms which frequently utilize discrete actions or task policies with structured interpretations.

Additionally, significant attention has been given to LLMs applicability towards embodied reasoning in grounded environments [24, 26]. Recent research demonstrates some ability for LLMs to solve some classes of planning problems directly. However, planning performance of these models is inconsistent at best and insufficient at worst, with state of the art planning architectures still critically failing in some domains (e.g. the *floortile* domain in [27]). This has led to significant debate about whether LLMs are capable of planning in a traditional sense [12]. Hybrid approaches which utilize both foundation models and symbolic planners are gaining in popularity, as they are able to leverage the strengths of both approaches [4, 7, 27–29].

C. Continual/Lifelong Learning for Planning

Lifelong or Continual Learning (CL) presents a significant challenge for training wherein all data is not collected a priori [30]. It is essential for robots operating in complex and loosely structured environments (e.g. in the home) to augment their training sets as they operate. Mendez *et al.*[31] have looked at CL paradigms utilizing diffusion models as samplers in a bi-level TAMP framework. They propose the CL problem as one of compositionality [32], as opposed to most other research which follows more traditional train and test datasets.

III. Problem Statement

CLIMB framework utilizes the following as input:

1) **Domain description:** Given in natural language, in order to provide the general context of the environment the system is operating in.
2) **Logical actions:** A set of predefined low-level primitives that the system can use to interact with the world, defined as python function signatures.
3) **Tasks to complete:** Given in natural language, a set of tasks for the system to complete and represent into its single logical world model. These tasks also serve as the curriculum to enables the logical planner to find a generalized representation that can solve new instances of such tasks.
4) **(optional) Previous world model:** In case the approach has already been run in the same domain, we can take the resulting logical world model and expand it with additional predicates for completing more varied and complex tasks.

Using these inputs, CLIMB achieves two goals:

1) **Solving the planning tasks:** Through repeated execution in the domain, analysis of failures, and fixing errors in generated plans or predicates, the goal of the system is to incrementally solve the entire list of tasks while acquiring knowledge about how the environment functions through the process.
2) **Building an incremental logical world model:** While solving individual tasks the system reuses and expands upon the logical world model produced from previous tasks. Upon completion of all individual tasks, we are left with a model that can be used to planning solutions for new instances of similar tasks. This model (domain and predicates) can be used as a prior in future executions of this pipeline on new sets of tasks with the same robot embodiment.

IV. CLIMB: Model Structure

Overview – The overall architecture for the framework is presented in Figure 2. CLIMB is comprised of modules that generate the PDDL, construct a plan trace for the given problem, observe the robot's performance, and refine the PDDL through observation and queried solutions from the LLM. Each of the LLM modules utilizes the `gpt-4o-2024-08-06` model from OpenAI.

Implementation details and full prompts for each of the modules can be found at https://plan-with-climb.github.io/.

A. Language-Model-Guided Domain Generation

CLIMB uses four modules to interface with an LLM:

Domain and Problem Generation – The domain-specific language (DSL) generation module converts a given domain

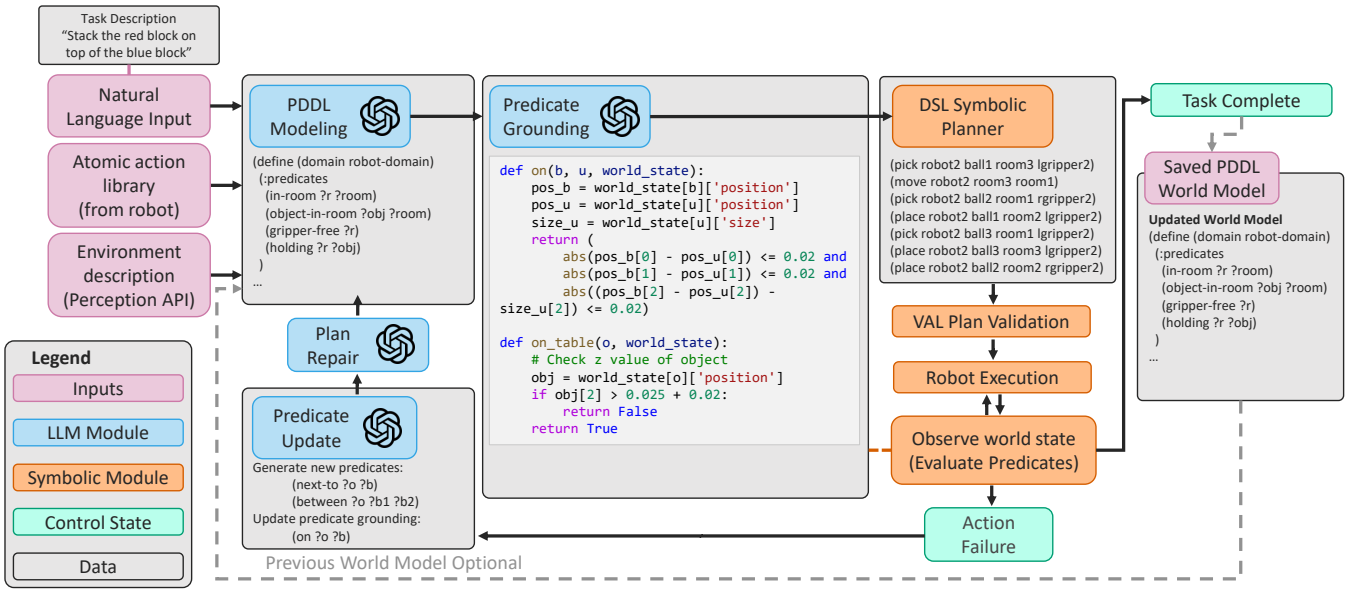

Fig. 2: The CLIMB Planning Framework includes multiple independent modules for problem translation, planning, predicate generation, verification, execution, and perception.

and problem description in unstructured natural language to structured PDDL representations. This generated DSL functions as an estimated representation of a zero-shot world model for the task planner. The initial generation process draws inferences about the logical predicates governing the environment based exclusively on the general world knowledge contained in the LLM and the user's description of the domain.

```
Input:
1) Domain description (natural language)
2) Atomic actions (function signatures and code)
3) (optional) Stored domain from previous tasks
Output:
1) Domain specific PDDL
2) List of domain predicates to implement
```

Predicate Grounding and Debugging – This module is used to generate executable Python functions that convert the continuous world state observed from the perception API into a logical predicate set. Grounding predicates allows us to compare the state of the world after each attempted action to the simulated logical state represented by the PDDL model. The comparison of simulated to perceived logical state serves as the primary error identification mechanism by which previously unseen relationships and constraints can be modeled by the predicate inventor. This module also automatically debugs and corrects syntax errors in the predicate grounding functions through analysis of error messages and performing function regeneration if needed.

```
Input:
1) Domain description (natural language)
2) Continuous state representation (python dict)
3) Existing predicate names and functions
4) Names and descriptions of new predicates
Output:
1) New predicate names and evaluation functions
```

Predicate Update – When an unexpected logical state is observed after executing an action, the Predicate Update module generates plausible explanations for the phenomena in the form of new or modified predicates. This module's inputs include the current predicate set, the problem and plan that is currently being investigated, and the world state in which the error occurred. From this information, the language model is instructed to reason through potential root causes and proposes an updated predicate set to represent dynamics in the domain. Any new or modified predicates are then re-grounded through the Predicate Grounding module.

```
Input:
1) Generated PDDL domain and problem
2) Generated PDDL plan
3) World state (before and after execution)
4) Natural language error message
Output:
1) New predicate names and descriptions to resolve
   error
```

Plan Repair – After an execution failure and predicate generation, the Plan Repair module updates the domain representation to include new predicates and constraints. Inputs to this module are the previous domain and problem DSL, the natural language task description, the world state at failure, and the updated predicate set. This module outputs an updated DSL domain which includes any new or modified predicates and an updated problem file to reflect the current world state. By updating the problem DSL, the system can recover from unintended consequences of executing infeasible actions. For example, if a stack of blocks is knocked over during execution, the initial problem state will be modified to reflect the fact that these blocks are no longer stacked.

```
Input:
1) Task description (natural language)
2) Generated domain and problem PDDL
3) List of available actions
4) List of perceived objects in the world
5) Names and descriptions of new predicates to add
Output:
1) Improved PDDL domain and problem utilizing new
   predicates
```

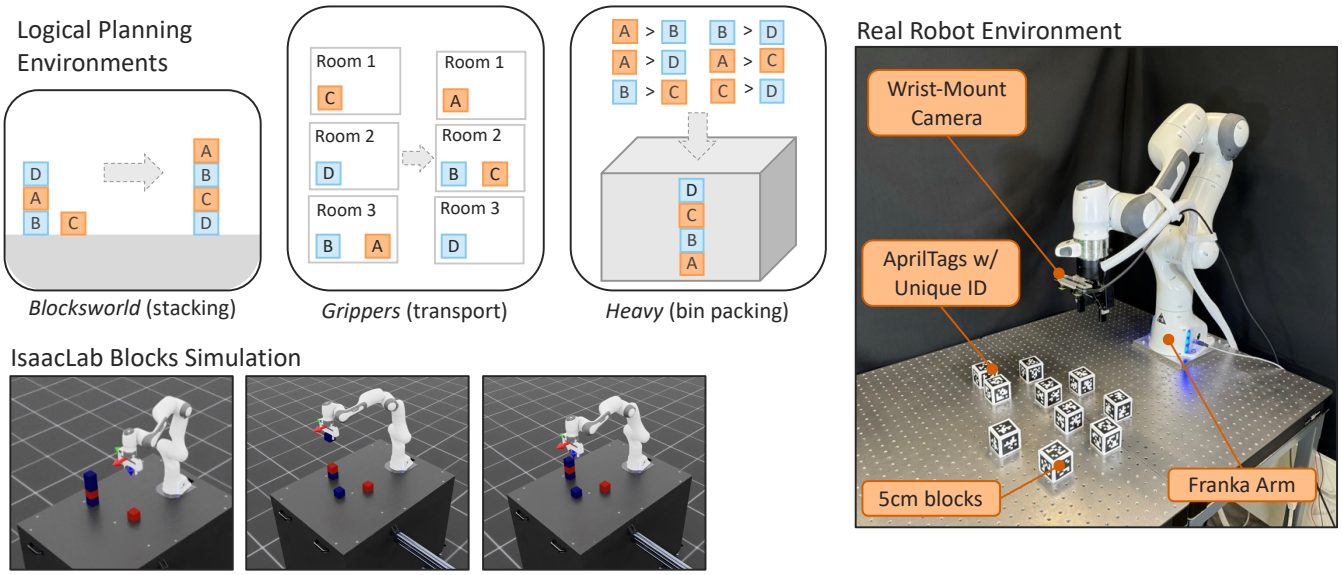

Fig. 3: We evaluate CLIMB across logical, simulated, and real domains. Logical domains (blocksworld, grippers, and heavy) provide a convenient mechanism for comparing performance across a variety of robot embodiments. Experiments in IsaacSim and Real environments extend the block manipulation setting to include more complex stacking and arranging tasks. The simulated and real also serve to evaluate predicate grounding and learning from real experience.

B. Planning & Perception

PDDL Symbolic planner – To generate the plan trace for a given problem, we make use of the FastDownward symbolic planning framework [33] in this study. FastDownward (FD) takes as input the domain and problem PDDL files generated for a specific problem and outputs a plan trace for execution. Internally FD implements a best-first search algorithm with a "causal graph heuristic" which is derived from the recursive decomposition of subtasks towards the goal. It is worth noting that our architecture can make use of any symbolic PDDL planner; it has additionally been tested using the Fast-Forward [34] and Pyperplan [35] planning systems. After a plan has been calculated, we additionally verify syntactic correctness and adherence to the generated PDDL by processing the output on the VAL plan validator [36]. VAL is a logical plan simulator which utilizes the domain and problem file given to evaluate if the plan is valid within the domain and correctly solves the task. While VAL is not able to check the semantic correctness of the domain and problem, it is capable of ensuring the plan's validity for the given domain. We use the Unified Planning library [37] for its implementations of all of the above planners, VAL integration, and simulated plan rollout.

BlocksWorld++ Action Set:
- `pickup(block o)`
- `putdown(block o)`
- `stack(block o, block under)`
- `unstack(block o, block under)`
- `place_between(block o, block a, block b)`
- `stack_on_two(block o, block a, block b)`
- `place_in_front_of(block o, block ref)`
- `place_behind(block o, block ref)`
- `place_to_right_of(block o, block ref)`
- `place_to_left_of(block o, block ref)`

Plan Execution and Skill Library – The plan execution module wraps a library of executable atomic skills and enables the execution of generated plans on the environment. The library of skills is specific to the robot's morphology and to an extent, its domain of operation. However, the skill library should ideally generalize to a variety of tasks within the same environment. By way of example, above are the actions utilized in our *BlocksWorld++* experimental domain.

Perception API – While the robot executes the plan trace generated to solve the tasked problem, a domain-specific perception API is utilized to evaluate if the robot accomplished each action successfully. For the real robot system, we utilize AprilTag2 [38] to observe the position of objects in the environment and distinguish between blocks. This perception module can be substituted for any system that tracks and outputs the 6D pose of objects (e.g. FoundationPose [39] for unlabeled objects). After each action is attempted, the perception module collects the state of all objects in the workspace and the proprioception of the robot. This continuous world state representation is then evaluated on the generated predicate functions created by the Predicate Inventor to determine the estimated logical world state.

V. Experiments

The goal of our experiments aims to evaluate the following hypothesis: Can LLMs create robust DSL domains and problems without feedback from human experts? How effective are LLMs at representing common world predicates? Can feedback for predicate learning be generated automatically and interpreted by the LLM without human intervention? Do the environment models generated by CLIMB generalize to new problems within the domain?

We evaluate these questions across *logical*, *simulated*, and *real* domains, each with separate characteristics.

A. Logical Planning Domains

We first evaluate the performance of our architecture on three logical plan-level domains: *blocksworld* [27], *grippers* [27], and *heavy* [8]. These domains provide reasonable

TABLE I: Performance on the Logical Planning Domains CLIMB demonstrates an ability to generate correct zero-shot plans in some cases, but importantly the system is able to improve its representation over successive (maximum of five) attempts to signxificantly improve performance ($N = 60$ for each case).

Dataset	LLM Plan	CLIMB 0-Shot	CLIMB Few-Shot
BLW	0.12	0.40	**0.80**
	(0.05, 0.20)	(0.28, 0.53)	**(0.70, 0.90)**
GRP	0.10	0.53	**0.93**
	(0.03, 0.18)	(0.42, 0.67)	**(0.87, 0.98)**
HVY	**0.68**	0.17	**0.67**
	(0.57, 0.80)	(0.08, 0.27)	**(0.55, 0.78)**

scenarios for generating manipulator plans and provide points of comparison to related work. As these domains are already logical (i.e. state of the world our system has access to is logical), here we are not evaluating the predicate grounding capabilities of our system. Instead, it allows us to evaluate the capabilities of the system to propose the initial domain, improve it based on failure in executing a plan, and incrementally improve the domain across several tasks in the domain.

Domain Description –

- `Blocksworld` This domain tasks the agent to stack and unstack columns of blocks to match specific configurations. A popular foundational planning problem, it is likely some examples of this type of environment are present in the LLM training corpus.
- `Grippers` This domain tasks multiple robots to pick up and transport objects between several different rooms or areas. This domain includes a logical representation for robot position requiring the plan to coordinate multiple robots with one another.
- `Heavy` This domain tasks the agent to pack a box or crate with objects, sorting by weight so that heavier objects are placed below lighter objects.

Correctness of DSL built with Iterative Model Building – Experiments in the logical domain serve to evaluate the ability of LLMs to create world representations in DSL. We evaluate both the zero-shot and few-shot performance of CLIMB incorporating up to five rollouts (i.e. executions) and iterations of feedback correction. In the logical domain, we substitute the execution and perception loop with the VAL plan validator [36] using a ground truth PDDL domain and problem. VAL models and simulates plan execution in the logical predicate space and evaluates both if the overall plan was successful and, if unsuccessful, where a given error occurs.

Table I presents the overall success rate in each of the three evaluated domains along with 95% confidence intervals. Each domain was evaluated over three iterations of 20 problems, in both zero-shot (maximum one rollout) and few-shot (maximum five rollouts) scenarios. The results in Table I demonstrate a moderately capable planner in a zero-shot setting with an overall performance of 37%. However, overall performance increases to 80% when the CLIMB framework can be used to repair and improve the domain from failures.

TABLE II: Blocksworld Predicate Grounding Performance The LLM demonstrates a capability to generate correct grounding functions for common predicates in Blocksworld. Functions were evaluated both for `syntactic` correctness and `semantic` correctness with respect to examples with known ground truth. We find that the relative frequency of syntax errors is generally low, but we are able to debug to improve performance.

Predicate	Zero-Shot	With Syntax Fixing
`on-table`	0.95	**1.00**
`on`	0.25	**0.45**
`holding`	0.50	**0.65**
`arm-empty`	0.10	**0.35**
`clear`	0.60	**0.80**

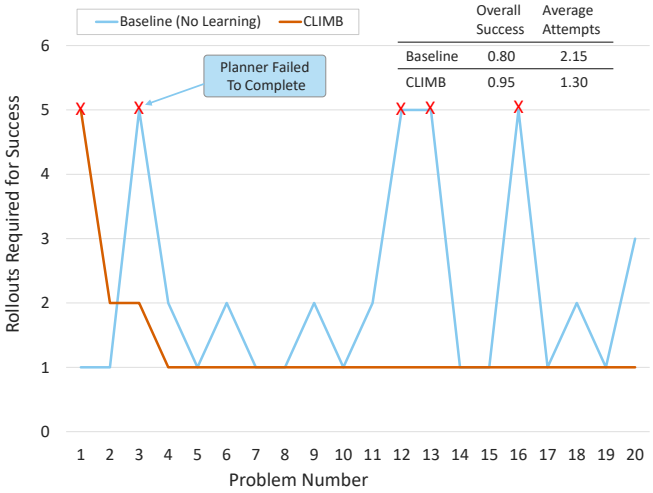

Fig. 4: Evaluation of the continual learning capabilities of the CLIMB framework on the *blocksworld* logical dataset. The baseline in this case is the same planner without saving domain and predicate information between problems. CLIMB with continual learning significantly outperforms the baseline.

Effect of Continual Learning on Performance –

To evaluate the ability of CLIMB to produce generalized domain representations, we conduct a comparison of CLIMB's continual learning with a baseline. In the baseline case the planner is still able to execute and learn predicates as in the full pipeline, however results are not saved between tasks. The results of this generalization experiment is shown in Figure 4. By caching the learned domain and predicates between runs, CLIMB is able to achieve better performance and use 40% less rollouts to accomplish all tasks in the dataset.

B. Simulated Robot World

BlocksWorld++: IsaacLab Block Manipulation Domain – We have developed an implementation and extension of the BlocksWorld problem in Nvidia IsaacLab [40]. This environment enables us to evaluate our approach on the BlocksWorld-type domain, but with continuous state variables and more complex tasks requiring new predicates. We can evalutate whether a correct logical state can be extracted from ground truth continuous state information, and whether this mapping can be iteratively improved based on feedback from

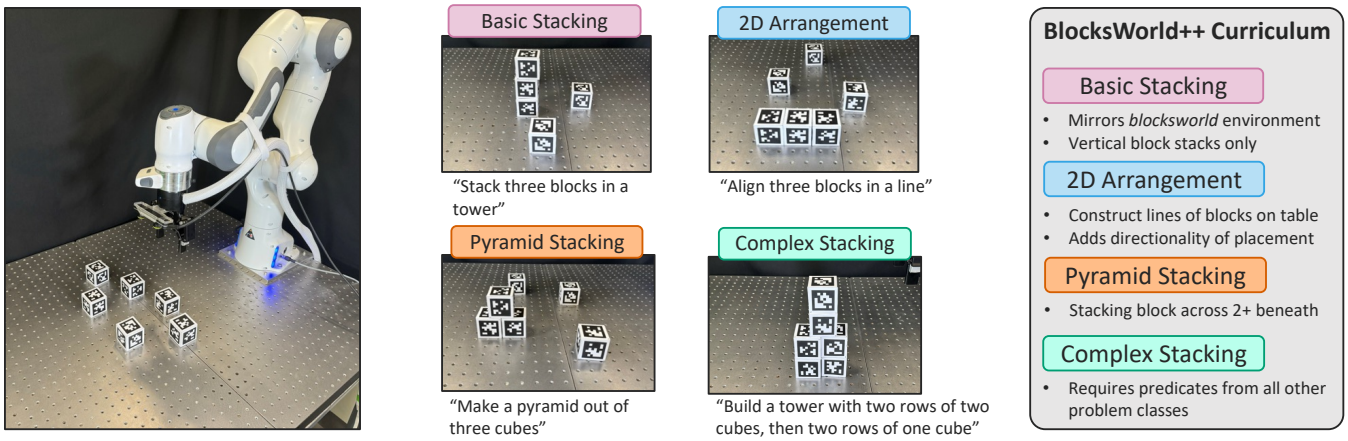

Fig. 5: Expanding on the blocksworld problem formulation, we have developed the *BlocksWorld++* dataset which adds additional levels of complexity to the traditional blocksworld environment. It includes relative 2D placement on the table and stacking across multiple blocks.

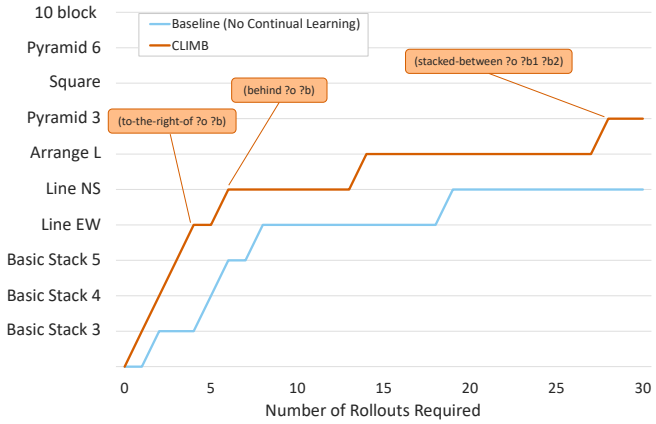

Fig. 6: Evaluation of CLIMB's ability to generalize to related tasks in the *BlocksWorld++* dataset. We label enabling predicates which allow the planner to succeed on new categories of problem. Over 40 iterations, CLIMB is successfully able to solve 7 of the 10 *BlocksWorld++* problems.

plan execution in the domain.

Predicate Grounding Function Generation – To evaluate our predicate generator, we construct grounded predicates from the correct ground truth PDDL domain. We then initialize the IsaacSim environment, extract the scene information, and evaluate our generated predicates to construct a perceived logical world state. This perceived world state is compared to the ground truth PDDL problem initial state for accuracy. We evaluate predicate grounding in isolation by asking it to generate grounding functions for a list of predicates from a given domain and then comparing the values to the ground truth logical values along a single task execution. Table II we show results for 20 independent predicate set generations. Automated syntax error fixing provides only a minor benefit in performance. The majority of errors are caused by semantic, rather than syntactic issues.

Domain Generalization on Blocksworld++ Dataset – Figure 6 demonstrates the ability for CLIMB to expand it's predicate set to include knowledge relevant to different classes of problems with the same robot embodiment. Evaluating on the *BlocksWorld++* dataset, CLIMB is able to quickly learn simple 1D arrangement in lines on the table. Though more iterations are required, the system is also able to learn spanning across two blocks for the 3 block pyramid problem and more complex horizontal arrangements including an L shape (combination of line NS and line EW).

C. Physical Robot in the Real-World

Finally, we demonstrate the system in the real world using the *BlocksWorld++* domain. Figure 5 showcases the *BlocksWorld++* curriculum for continual learning evaluation on real hardware. This environment can serve as a benchmark to evaluate CLIMB along with other continual learning paradigms. Moving from simulation to real, the additional challenges are perception and stochastic motion control. To address the challenge of perception, we integrate AprilTag2 [38] markers on the cubes and a camera mounted on the robot gripper. By returning to a home position after each action, we can gather full pose information for all objects in the scene. We show examples using this system to evaluate *BlocksWorld++* tasks at https://plan-with-climb.github.io/.

VI. Conclusion

In this paper we presented CLIMB, a system for incremental learning of logical domains and continous-state grounding functions through interaction. It requires only a general description of the domain and tasks that need to be solved and access to a set of low-level primitives. CLIMB generates an initial planning domain and logical representation of the task, uses a symbolic planner to solve it, and then learns new world constraints through observation of discrepencies in the logical state expected and observed through the predicate grounding functions.

We evaluate the method capabilities across several logical domains, showing excellent performance given the limited information given as a prior. That, through iterative world interaction and refinement, CLIMB can learn non-intuitive predicates and world constraints to improve performance across successive attempts. Furthermore, we develop the *BlocksWorld++* dataset, enabling evaluation of continual learning frameworks in a unified manner across logical, simulated, and real domains.

References

[1] Y. Hu, Q. Xie, V. Jain, J. Francis, J. Patrikar, N. Keetha, S. Kim, Y. Xie, T. Zhang, S. Zhao, Y. Q. Chong, C. Wang, K. Sycara, M. Johnson-Roberson, D. Batra, X. Wang, S. Scherer, Z. Kira, F. Xia, and Y. Bisk, "Toward general-purpose robots via foundation models: A survey and meta-analysis," 2023. [Online]. Available: https://arxiv.org/abs/2312.08782

[2] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," 2023. [Online]. Available: https://arxiv.org/abs/2306.15724

[3] L. Wong, J. Mao, P. Sharma, Z. S. Siegel, J. Feng, N. Korneev, J. B. Tenenbaum, and J. Andreas, "Learning adaptive planning representations with natural language guidance," 2023. [Online]. Available: https://arxiv.org/abs/2312.08566

[4] M. Han, Y. Zhu, S.-C. Zhu, Y. N. Wu, and Y. Zhu, "Interpret: Interactive predicate learning from language feedback for generalizable task planning," 2024. [Online]. Available: https://arxiv.org/abs/2405.19758

[5] Z. Wang, S. Cai, G. Chen, A. Liu, X. Ma, and Y. Liang, "Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents," 2024. [Online]. Available: https://arxiv.org/abs/2302.01560

[6] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," 2023. [Online]. Available: https://arxiv.org/abs/2305.16291

[7] 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," 2024. [Online]. Available: https://arxiv.org/abs/2407.09792

[8] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. Kaelbling, and M. Katz, "Generalized planning in pddl domains with pretrained large language models," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 18, pp. 20 256–20 264, Mar. 2024. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/30006

[9] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," 2023. [Online]. Available: https://arxiv.org/abs/2209.07753

[10] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 2086–2092.

[11] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, "Autotamp: Autoregressive task and motion planning with llms as translators and checkers," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 6695–6702.

[12] S. Kambhampati, K. Valmeekam, L. Guan, M. Verma, K. Stechly, S. Bhambri, L. Saldyt, and A. Murthy, "Llms can't plan, but can help planning in llm-modulo frameworks," 2024. [Online]. Available: https://arxiv.org/abs/2402.01817

[13] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018.

[14] S. James, B. Rosman, and G. Konidaris, "Autonomous learning of object-centric abstractions for high-level planning," in *Proceedings of the The Tenth International Conference on Learning Representations*, 2022.

[15] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. Kaelbling, and J. B. Tenenbaum, "Predicate invention for bilevel planning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 10, 2023, pp. 12 120–12 129.

[16] A. Ahmetoglu, M. Y. Seker, J. Piater, E. Oztop, and E. Ugur, "Deepsym: Deep symbol generation and rule learning for planning from unsupervised robot interaction," *Journal of Artificial Intelligence Research*, vol. 75, pp. 709–745, 2022.

[17] S. Nguyen, O. Oguz, V. Hartmann, and M. Toussaint, "Self-supervised learning of scene-graph representations for robotic sequential manipulation planning," in *Conference on Robot Learning*. PMLR, 2021, pp. 2104–2119.

[18] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," 2019. [Online]. Available: https://arxiv.org/abs/1904.12584

[19] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[20] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[21] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," 2023. [Online]. Available: https://arxiv.org/abs/2305.10601

[22] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, and T. Hoefler, "Graph of thoughts: Solving elaborate problems with large language models," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, p. 17682–17690, Mar. 2024. [Online]. Available: http://dx.doi.org/10.1609/aaai.v38i16.29720

[23] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," 2023. [Online]. Available: https://arxiv.org/abs/2210.03629

[24] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do as i can, not as i say: Grounding language in robotic affordances," 2022. [Online]. Available: https://arxiv.org/abs/2204.01691

[25] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," 2023. [Online]. Available: https://arxiv.org/abs/2303.11366

[26] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," 2022. [Online]. Available: https://arxiv.org/abs/2209.11302

[27] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+p: Empowering large language models with optimal planning proficiency," 2023. [Online]. Available: https://arxiv.org/abs/2304.11477

[28] 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," 2023. [Online]. Available: https://arxiv.org/abs/2305.14909

[29] 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, p. 981–997, Aug. 2023. [Online]. Available: http://dx.doi.org/10.1007/s10514-023-10133-5

[30] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," 2019. [Online]. Available: https://arxiv.org/abs/1907.00182

[31] J. Mendez-Mendez, L. P. Kaelbling, and T. Lozano-Pérez, "Embodied lifelong learning for task and motion planning," 2023. [Online]. Available: https://arxiv.org/abs/2307.06870

[32] J. A. Mendez and E. Eaton, "How to reuse and compose knowledge for a lifetime of tasks: A survey on continual learning and functional composition," *arXiv preprint arXiv:2207.07730*, 2022.

[33] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, p. 191–246, July 2006. [Online]. Available: http://dx.doi.org/10.1613/jair.1705

[34] J. Hoffmann, "Ff: The fast-forward planning system," *AI magazine*, vol. 22, no. 3, pp. 57–57, 2001.

[35] Y. Alkhazraji, M. Frorath, M. Grützner, M. Helmert, T. Liebetraut, R. Mattmüller, M. Ortlieb, J. Seipp, T. Springenberg, P. Stahl, and J. Wülfing, "Pyperplan," https://doi.org/10.5281/zenodo.3700819, 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3700819

[36] R. Howey, D. Long, and M. Fox, "Val: automatic plan validation, continuous effects and mixed initiative planning using pddl," *16th IEEE International Conference on Tools with Artificial Intelligence*, pp. 294–301, 2004. [Online]. Available: https://api.semanticscholar.org/CorpusID:3098522

[37] AIPlan4EU, "Unified-planning," https://github.com/aiplan4eu/unified-planning, 2021.

[38] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4193–4198.

[39] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "Foundationpose: Unified 6d pose estimation and tracking of novel objects," 2024. [Online]. Available: https://arxiv.org/abs/2312.08344

[40] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.