Bridging Natural Language and Emergent Representation in Hierarchical Reinforcement Learning

Zihe Ji[1,2], **Sao Mai Nguyen**[1,3], **Mehdi Zadem**[4]
[1]Flowers Team, U2IS, ENSTA Paris, IP Paris & Inria
[2] Shanghai Jiaotong University
[3]IMT Atlantique, Lab-STICC, UMR CNRS 6285
[4]LIX, École Polytechnique,Institut Polytechnique de Paris, France
zihe.ji@ensta-paris.fr, nguyensmai@gmail.com, zadem@lix.polytechnique.fr

Abstract

Hierarchical Reinforcement Learning (HRL) breaks down complex tasks into manageable subtasks, but faces challenges with efficiency and generalization in high-dimensional, open-ended environments. Human In The Loop approaches offer a potential solution to these limitations. In this work, we propose the integration of large language models (LLMs) with HRL, leveraging LLMs' natural language and reasoning capabilities and study how to bridge the gap between human instructions and HRL's learned abstract representations. By translating human demonstrations into actionable reinforcement learning signals, LLMs can improve task abstraction and planning within HRL. Our approach builds upon the Spatial-Temporal Abstraction via Reachability (STAR) algorithm, using a LLM to optimize the hierarchical planning process. Empirical results obtained on continuous control tasks illustrate the potential of LLMs to enhance HRL particularly in environments requiring spatial reasoning and hierarchical control.

1 Introduction

Open-ended learning [Doncieux et al., 2018] is characterised by multi-task learning, where the set of tasks is not known in advance, including interrelated tasks where easy tasks can be composed into more complex tasks. Recent advances in Hierarchical Reinforcement Learning (HRL) have allowed agents to efficiently solve complex, long-horizon problems by decomposing them into easier, more manageable sub-problems. In particular, the STAR [Zadem et al., 2024] algorithm learns, using intrinsic motivation, an abstract goal space that preserves environment dynamics by focusing on reachability relations between sets of states. Unlike other HRL algorithms that use only continuous space or continuous abstract representations, STAR's learned abstraction acts as a discretisation of the state space, where every goal is a set of states that exhibit similar reachability properties in the task. This goal representation is acquired online with the hierarchical policy, while driving sample efficiency and interpretability.

Still, current RL algorithms fail to transpose to robots because they require a large number of real actions to sample the continuous high-dimensional space of states and actions, hence requiring a lot of time and are not scalable for open-ended learning. Therefore, complementary developmental mechanisms need to constrain the complexity of the exploration areas and structure the environment by guiding the learning curriculum toward learnable subspaces. Furthermore, the learned goals are not directly interpretable since it is misaligned with human representations such as in natural language. An important aspect of intelligent systems is to ensure that we can communicate with them intuitively

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

and efficiently. A learning agent should allow users to provide feedback and instructions to guide the learning process just as we teach other humans. Following the principles of a human-centered approach [Boy, 2017], the agent should ground its reasoning in a common language with humans. The Human In The Loop (HITL) [Wu et al., 2021, Retzlaff et al., 2024] Reinforcement Learning paradigm studies how to integrate humans in the different stages of an agent's life cycle. This includes how human demonstrations can be used to enhance the learning process of primitive [Nguyen and Oudeyer, 2012] or sequential [Duminy et al., 2019] tasks, and how humans can instruct RL agents via natural language [Colas et al., 2020]. In this vein, a subfield that has gained attention recently is the integration of Large Language Models (LLMs) in synergy with RL agents. The advances achieved in building LLMs (e.g. OpenAI's GPT, Meta's LLAMA, Anthropic's Claude), have accelerated the creation of language based HITL approaches [Pternea et al., 2024] in different flavors. First, RL can be used in service of training and improving LLMs in natural language tasks such as conversation and question answering. In the vein of approaches, Reinforcement Learning from Human Feedback (RLHF) [Ouyang et al., 2022] has demonstrated how human feedback can be captured by a RL agent and used to fine-tune large language models. Inversely, a LLM can benefit RL agents in improving sample efficiency and injecting a reasoning layer [Du et al., 2023] that would alleviate the need for extensive exploration, especially in the initial training phases. A popular example of such approaches rely on the LLM as a high-level planner, providing instructions to the RL agent [Wong et al., 2023, Ichter et al., 2022, Wu et al., 2023], which can then be used to guide the learning process. Under such architectures, the LLM has to communicate with the RL agent in a common language that allows to express goals. Establishing this common language often resorts to using predefined predicates, reducing the generality for open-ended learning.

With the outlook of establishing a language to communicate with a dynamic, emerging symbolic representation in open-ended learning, we propose in this paper some perspectives on how the interpretability of the reachability-aware goal abstraction in STAR can allow for a LLM to reason about abstract goals and boost the planning capabilities of the approach. We argue that this approach is on the one hand allowing humans to instruct the algorithm in natural language, and on the other hand, to allow the algorithm to clearly communicate its behavior. Our main contributions in this work is to explore using LLMs as high-level instructor for the STAR algorithm, and whether it can provide an interpretable and coherent explanation of the agent's behaviour to humans.

2 Spatial-Temporal Abstraction via Reachability (STAR) Algorithm

We base our work on the STAR algorithm, which efficiently partitions the state space. The partitioning data from STAR is used to test the integration of language instructions in the HRL framework.

2.1 Overview of the STAR Algorithm

We consider a goal-conditioned Markov Decision Process $(\mathcal{S}, \mathcal{A}, P, r_{\text{ext}})$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is a continuous state space, $\mathcal{A}$ is an action space, $P\left(s_{t+1} \mid s_t, a_t\right)$ is the transition function, and $r_{\text{ext}} : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward, defined as the negative distance to the goal $g^* \in \mathcal{S}$: $r_{\text{ext}}\left(s, g^*\right) = -\left\|g^* - s\right\|_2$. The objective in multi-task reinforcement learning is to learn a goal-conditioned policy π that maximizes the expected reward by sampling actions $a \sim \pi\left(s_t \mid g^*\right)$ at each timestep.

The goal abstraction is modeled by a function $\mathcal{N} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$ that maps states to sets of states (i.e. $\forall s \in \mathcal{S}, \mathcal{N}(s) \subseteq \mathcal{S}$). We refer to the abstract goal space as $\mathcal{G}_{\mathcal{N}}$. The elements of $\mathcal{G}$ are denoted as G.

The function $\mathcal{N}$ depends on the abstraction method: Mannor et al. [2004] use stochastic partitioning with linear subpolicies, while STAR [Zadem et al., 2024] uses k-step reachability: a state s can reach s' using policy $\pi(., G_j)$ in k steps. Thus, the abstract goal space $\mathcal{G}$ consists of sets of reachable states.

The STAR architecture comprises three hierarchical agents:

- **Navigator**: The high-level agent selects an abstract goal $G \in \mathcal{G}$ to guide the agent towards the task goal g^*: $G_{t+k} \sim \pi_{\text{Nav}}\left(s_t, g^*\right)$.
- **Manager**: The mid-level agent picks subgoals in the state space, conditioned on the Navigator's goal: $g_{t+l} \sim \pi_{\text{Man}}\left(s_t, G_{t+k}\right)$.
- **Controller**: The low-level policy samples actions to reach the subgoal: $a \sim \pi_{\text{Cont}}\left(s_t, g_{t+l}\right)$.

The Manager and Controller use TD3 [Fujimoto et al., 2018] for learning, while the Navigator employs Q-learning. Each agent operates at different timescales: the Navigator selects a goal every k steps, the Manager every l steps (with k a multiple of l), and the Controller at each step. Initially, the abstraction $\mathcal{G}$ is coarse, making direct goal-reaching challenging. The Manager's subgoals serve as intermediate targets, facilitating easier learning for the Controller. This structure allows STAR to guide the agent through large state abstractions while supporting low-level policy learning.

The Manager and Controller use TD3 [Fujimoto et al., 2018] for learning, while the Navigator employs Q-learning. Each agent operates at different timescales: the Navigator selects a goal every k steps, the Manager every l steps (with k a multiple of l), and the Controller at each step. Initially, the abstraction $\mathcal{G}$ is coarse, making direct goal-reaching challenging. The Manager's subgoals serve as intermediate targets, facilitating easier learning for the Controller. This structure allows STAR to guide the agent through large state abstractions while supporting low-level policy learning.

2.2 Integration of LLM

For tasks in real-world environments, humans intuitively understand and navigate them. For instance, navigating a maze, moving from the living room to the kitchen, can be easily communicated using language. To reason and compose symbols grounded in a continuous environment, we take advantage of the discrete representation output by STAR as an intermediary capable of extracting the abstract spatial states of the algorithm and human instructions, then converting them into a format the algorithm can understand, ultimately accelerating the learning process. To achieve this, we propose the conversion of abstract spatial states and goals into a textual representation using LLM.

As the top-level agent, the Navigator only selects the next abstract region $G_{t+k} \sim \pi_{\mathrm{Nav}}(s_t, g^*)$, we propose a translation instruction experiment. In the first experiment, we test the ability of LLMs to perform full route planning based on human-provided instructions, $(G_{t+k}, \ldots, G_{t+nk}) \sim \pi_{\mathrm{LLM}}(X, s_t, g^*)$. Simultaneously, from another perspective, to evaluate the interactivity and alignment of LLM with spatial reasoning, we propose a naming experiment. In the second experiment, we translate abstract regions, $G \in \mathcal{G}$, into natural language descriptions and test whether LLM can support the mapping between continuous spatial regions and symbolic representations.

2.3 Representation of States and Goals

The Ant, adapted from Duan et al. [2016] and Nachum et al. [2018], is a simulated quadrupedal robot with a 30-dimensional state space, including positions, orientations, velocities, and joint angles. The action space is continuous and 8-dimensional, corresponding to forces applied on the joints.

We evaluate two tasks in a 2D environment of size 25 for each dimension: **AntMaze**, where the Ant navigates a $\supset$-shaped maze to the exit, and **AntFall**, which involves crossing a chasm using a movable block as a bridge. These tasks are hierarchical, requiring both low-level movement and high-level navigation. The environment uses Mujoco physics simulator [Todorov et al., 2012]. A training episode lasts up to 500 timesteps. The reward is the negative Euclidean distance to the goal, scaled by 0.1, with success if the distance is smaller than 5.

We use the partitioning from the STAR algorithm's training to test integrating human demonstrations. Human instructions guide the agent in the AntMaze or AntFall environments. To represent partitioning data as prompts for the LLM, we use:

- **Maze layout**: Compressed textual form with marked obstacles and partition regions.
- **Coordination information**: Tracks the agent's current location and the goal.
- **Adjacency list**: Details neighboring relations for each region.

3 Experimental Evaluation

Naming Experiment for Spatial Regions To evaluate the LLMs' ability to generate human-readable descriptions of abstract goals, we utilized four scenarios with Timestep 930000 as a one-shot prompt (see Fig. 1 and Appendix. A). Llama3-8b-instruct and GPT-4o were tested; the former runs on a GPU with more than 16GB of RAM. Completing the STAR program for 5 million timesteps takes about 15 hours, with LLM inference taking 0.6 seconds each. Table. 1 shows the names given

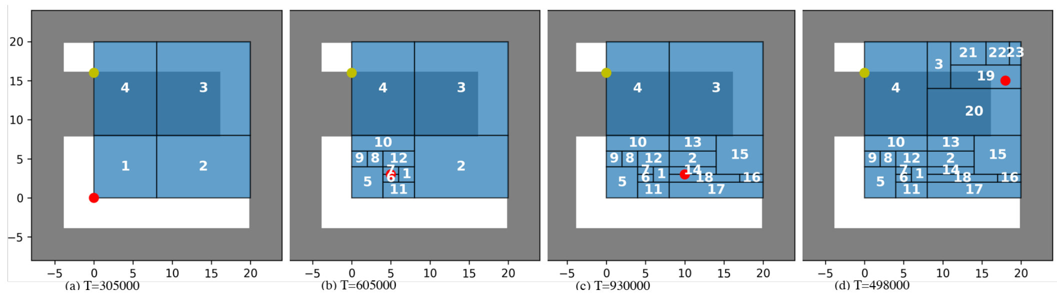

Figure 1: Four situations in the AntMaze environment at Timestep 305000, 605000, 930000, and 4980000. The red point is the agent's current location, and the yellow point is the goal.

Table 1: Unique Name given by LLM

Timestep - Region	LLAMA3.1-8B-Instruct	GPT4o
305000 - 2	Rightward Passage	Eastern Pathway
605000 - 1	**Western** Entrance	**Southern** Junction
605000 - 5	Leftward Passage	Western Approach
605000 - 6	**Rightward** Passage	**Northern** Link
605000 - 12	**Southern** Expansion	**Eastern** Border
4980000 - 3	Northern Passage	Northern Access
4980000 - 20	Southern Corridor	Southern Corridor
4980000 - 21	**Eastern** Extension	Northeastern Outlet

by the LLMs, when tasked with naming neighboring regions. The results indicate that the LLMs can generate clear and concise names for each region.

In Table. 1, bold text denotes incorrect region descriptions. The LLMs struggled with directional accuracy, particularly in densely packed situations (e.g., Timestep 605000, with 25% accuracy). However, when focusing on regions adjacent to the agent's location, directional accuracy exceeded 75%, suggesting that representing continuous regions as symbolic names using LLMs is feasible.

Instruction Translation Experiment In Fig. 1.b, a LLM might compose instructions into a complex planning: "Go east, then north past the wall, and finally west to the goal." With region segmentation, this becomes moving through regions ($G_n = (1, 2, 3, 4)$. We tested the LLM's ability to infer this sequence from such instructions (see Appendix. C for the prompts). We report the accuracy, defined as $\text{IoU} = \frac{|G_{\text{LLM}} \cap G_n|}{|G_{\text{LLM}} \cup G_n|}$, where G_{LLM} is the LLM-predicted sequence and G_n is the true sequence. Table. 2 reports the IoU for his ChatGPT-4o and Claude 3.5 Sonnet, with detailed outputs reported in Appendix. D. GPT-4o's errors stemmed from omitting intermediate regions, while Claude added extra ones. Both models achieved over 80% IoU, with 100% accuracy in predicting the next region, indicating effective translation of instructions into abstract regions.

Table 2: IoU Comparison Between GPT-4o and Claude 3.5 Sonnet

Environment	GPT-4o	Claude 3.5 Sonnet
AntMaze	82.1%	90.3%
AntFall	75.0%	100.0%
Total	81.25%	91.43%

4 Discussion

The experiments show that LLMs can enhance HRL tasks, particularly in sequential planning, despite its dynamic abstract representation. This is owing to STAR's emergent symbolic representation. LLMs effectively bridge human instructions and HRL, aiding task abstraction owing to its reasoning capability. Challenges remain in densely packed environments where directional errors occur. Overall, integrating LLMs into HRL can improve complex task performance in hierarchical control and spatial reasoning contexts. Our work hints for interpretable language grounding in open-ended learning through an emergent symbolic abstraction of a continuous multi-dimensional state space.

Acknowledgments and Disclosure of Funding

This work was partially supported by Hi! Paris.

Acknowledgements

The authors would like to thank Sergio Mover for his support.

References

G. A. Boy. *The handbook of human-machine interaction: a human-centered design approach*. CRC Press, 2017.

C. Colas, A. Akakzia, P. Oudeyer, M. Chetouani, and O. Sigaud. Language-conditioned goal generation: a new approach to language grounding for RL. *CoRR*, abs/2006.07043, 2020.

S. Doncieux, D. Filliat, N. Díaz-Rodríguez, T. Hospedales, R. Duro, A. Coninx, D. M. Roijers, B. Girard, N. Perrin, and O. Sigaud. Open-ended learning: A conceptual framework based on representational redescription. *Frontiers in Neurorobotics*, 12, sep 2018.

Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas. Guiding pretraining in reinforcement learning with large language models. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 8657–8677. PMLR, 2023.

Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

N. Duminy, S. M. Nguyen, and D. Duhaut. Learning a set of interrelated tasks by using a succession of motor policies for a socially guided intrinsically motivated learner. *Frontiers in neurorobotics*, 12:87, 2019.

S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K. Lee, Y. Kuang, S. Jesmonth, N. J. Joshi, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu. Do as I can, not as I say: Grounding language in robotic affordances. In *CoRL*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR, 2022.

S. Mannor, I. Menache, A. Hoze, and U. Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 71, 2004.

O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

S. M. Nguyen and P.-Y. Oudeyer. Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner. *Paladyn*, 3(3):136–146, 2012.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

M. Pternea, P. Singh, A. Chakraborty, Y. D. Oruganti, M. Milletarí, S. Bapat, and K. Jiang. The RL/LLM taxonomy tree: Reviewing synergies between reinforcement learning and large language models. *CoRR*, abs/2402.01874, 2024.

C. O. Retzlaff, S. Das, C. Wayllace, P. Mousavi, M. Afshari, T. Yang, A. Saranti, A. Angerschmid, M. E. Taylor, and A. Holzinger. Human-in-the-loop reinforcement learning: A survey and position on requirements, challenges, and opportunities. *J. Artif. Intell. Res.*, 79:359–415, 2024.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

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. *CoRR*, abs/2312.08566, 2023.

J. Wu, Z. Huang, C. Huang, Z. Hu, P. Hang, Y. Xing, and C. Lv. Human-in-the-loop deep reinforcement learning with application to autonomous driving. *CoRR*, abs/2104.07246, 2021.

Y. Wu, S. Y. Min, Y. Bisk, R. Salakhutdinov, A. Azaria, Y. Li, T. M. Mitchell, and S. Prabhumoye. Plan, eliminate, and track - language models are good teachers for embodied agents. *CoRR*, abs/2305.02412, 2023.

M. Zadem, S. Mover, and S. M. Nguyen. Reconciling spatial and temporal abstractions for goal representation. *arXiv preprint arXiv:2401.09870*, 2024.

A Prompt for Naming Experiment

The system prompt used in the experiment *Naming Experiment for Spatial Regions:*

In this task, You are a navigation assistant, helping agent to reach the goal. Based on the data, name the adjacent regions with a unique name understandable by the user to help him follow your instructions.

The following is the few-shot prompt used in the experiment:

Role: User
Content:
Data:
State: [10, 3], Region 14
Goal: [0, 15], Region 4
Adjacency list:
Region 1: [5, 11, 18]
Region 2: [13]
Region 3: [4]
Region 5: [1, 6, 11]
Region 6: [1, 11]
Region 11: [1, 6, 17]
Region 13: [3, 15]
Region 14: [1, 2, 15, 18]
Region 15: [2, 3, 14]
Region 17: [18]
Region 18: [1, 14]
The top-down view of the maze is shown below, W represents walls, A represents the agent's current position,G represents the goal. The number represents the region number:
4 4 4 4 4 3 3 3 3 3 3
G 4 4 4 4 3 3 3 3 3 3
W W W W W W W W W 3 3
10 10 10 10 10 13 13 13 15 15 15
9 9 8 12 12 2 2 2 15 15 15
5 5 5 7 1 14 14 14 15 15 15
5 5 5 7 1 14 A 14 15 15 15
5 5 5 6 1 18 18 18 18 18 16
5 5 5 11 11 17 17 17 17 17 17

Thinking Process:
1.Determine which regions are directly adjacent to the agent's current location.
2.Assign Names Based on Relative Position: Use directional names (north, south, east, west) or (left, middle, right) based on the region's position relative to the agent or between the adjacent regions.
3.Special Features: If a region has unique features (like being entirely walled off or a sole connection to another important area), this should be reflected in the name.

Role: Assistant
Content:
Based on the maze layout and the adjacency list provided, the agent is currently in Region 14. The regions adjacent to Region 14 are Regions 1, 2, 15, and 18.

1. Region 1

Location: West of Region 14.
Unique Name: Western Gateway
Rationale: It's in the west direction relative to the agent's current position and acts as a connecting hub to other regions (5, 11, 18).

Table 3: Region status at different timesteps

Region	Timestep 305000	Timestep 605000	Timestep 930000	Timestep 4980000
Region 1	Top Left	Top-Left	Top Left Corner	One
Region 2	Top Middle	Top-Middle	Not Available	Two
Region 3	Top Right	Top-Right	Not Available	Three
Region 4	Bottom	Goal	Goal Region	Goal
Region 5	-	Bottom-Left	Top Middle	Five
Region 6	-	Bottom-Middle	Top Right	Six
Region 7	-	Bottom-Right	Not Available	Seven
Region 8	-	Not Found	Not Available	Eight
Region 9	-	Not Found	Not Available	Nine
Region 10	-	Not Found	State Region	Ten
Region 11	-	Bottom-Right Corner	Upper Middle	Eleven
Region 12	-	Not Found	Not Available	Twelve
Region 13	-	-	Down Left	Thirteen
Region 14	-	-	Current Region	Fourteen
Region 15	-	-	Down Middle	Fifteen
Region 16	-	-	Not Available	Sixteen
Region 17	-	-	Down Right	Seventeen
Region 18	-	-	Bottom Right	Current
Region 19	-	-	-	Nineteen
Region 20	-	-	-	Twenty
Region 21	-	-	-	Twenty-One
Region 22	-	-	-	Twenty-Two
Region 23	-	-	-	Twenty-Three

2. Region 2:

Location: Directly above Region 14 in the maze.
Unique Name: Northern Corridor
Rationale: It is located to the north of the agent's current position and appears to be a narrow passage leading to Region 13.

3. Region 15:

Location: Right next to Region 14 on the east side.
Unique Name: Eastern Gateway
Rationale: This region connects to multiple other regions (Regions 2, 3, and 14) and serves as a gateway towards the eastern part of the maze.

4. Region 18:

Location: Directly below Region 14 in the maze.
Unique Name: Southern Expansion

Rationale: It extends to the south and also connects to Region 1 and 17, offering a larger area or expansion from the agent's current position.

B Zero-shot Naming Experiment

Table. 3 shows the result of the zero-shot naming experiment. In this experiment, no shots were given to the LLM, and the LLM was asked to name all the regions, instead of concentrating on the regions adjacent to the agent's current location. The results show that the LLM give up when facing too complex tasks, such as naming all the regions in the maze. Therefore, we asked the LLM to focus on naming the regions adjacent to the agent's current location in the few-shot naming experiment.

Table 4: Region Information Across Timesteps

Timestep - Region	Location Description	Unique Name
	LLAMA3.1-8B-Instruct	
305000 - 2	Directly right of Region 1 in the maze	Rightward Passage
605000 - 1	West of Region 7	Western Entrance
605000 - 5	Directly left of Region 7	Leftward Passage
605000 - 6	Directly right of Region 7	Rightward Passage
605000 - 12	Below Region 7 in the maze	Southern Expansion
4980000 - 3	Directly above Region 19 in the maze	Northern Passage
4980000 - 20	Directly below Region 19 in the maze	Southern Corridor
4980000 - 21	Right next to Region 19 on the east side	Eastern Extension
	GPT4o	
305000 - 2	Directly east of Region 1 in the maze	Eastern Pathway
605000 - 1	South of Region 7	Southern Junction
605000 - 5	West of Region 7	Western Approach
605000 - 6	North of Region 7	Northern Link
605000 - 12	East of Region 7	Eastern Border
4980000 - 3	North of Region 19 in the maze	Northern Access
4980000 - 20	Below Region 19 in the maze	Southern Corridor
4980000 - 21	Northeast of Region 19	Northeastern Outlet

C Prompts for Instruction Translation Experiment

The system prompt used in this experiment: In this task, You are a navigation assistant, helping agent to reach the goal. Based on the data and the instruction given, tell the route to the agent to reach the goal.

The following is the few-shot prompt used in the experiment:

> *User:*
> *Data:*
> *State: [10, 3], Region 14*
> *Goal: [0, 15], Region 4*
> *Adjacency list:*
> *Region 1: [5, 11, 18]*
> *Region 2: [13]*
> *Region 3: [4]*
> *Region 5: [1, 6, 11]*
> *Region 6: [1, 11]*
> *Region 11: [1, 6, 17]*
> *Region 13: [3, 15]*
> *Region 14: [1, 2, 15, 18]*
> *Region 15: [2, 3, 14]*
> *Region 17: [18]*
> *Region 18: [1, 14]*
> *The top-down view of the maze is shown below, W represents walls, A represents the agent's current position,G represents the goal. The number represents the region number:*
> *4 4 4 4 4 3 3 3 3 3 3*
> *G 4 4 4 4 3 3 3 3 3 3*
> *W W W W W W W W W 3 3*
> *10 10 10 10 10 13 13 13 15 15 15*
> *9 9 8 12 12 2 2 2 15 15 15*
> *5 5 5 7 1 14 14 14 15 15 15*
> *5 5 5 7 1 14 A 14 15 15 15*

5 5 5 6 1 18 18 18 18 18 16
5 5 5 11 11 17 17 17 17 17 17

Instruction:
Go straight to east till the end, then go north till pass the wall. Finally go west and reach the goal.

Thinking Process:
1. Identify the agent's current region and the goal region.
2. Interpret the Instruction: Understand the directional commands provided in the instruction and translate them into movements between regions.
3. Plan the Route: Based on the adjacency list and the maze layout, determine the sequence of regions the agent should traverse to follow the given instructions and reach the goal.

Assistant:

Current State: The agent is located in Region 14, at coordinates [10, 3].
Goal: The target is in Region 4, at coordinates [0, 15].

Interpretation of Instructions:
Go straight to the right till the end: Starting from Region 14, moving east would involve transitioning through adjacent regions towards the eastern edge of the maze.
Then go north till pass the wall: After reaching the furthest right point accessible, the agent should then move northwards until they pass the wall.
Finally go west and reach the goal: The agent needs to move left from the northernmost point they reached, continuing until they arrive at the goal in Region 4.

Route Plan Based on Regions and Their Adjacencies:
Starting at Region 14: Move eastward within the maze. Adjacent regions from 14 are [1, 2, 15, 18]. The furthest east the agent can go from here within the maze layout while still adhering to the instruction to move to the right and then up includes moving to Region 15.

Move to Region 15: After reaching Region 15, the next step is to go north. From Region 15, the agent can continue to Region 3 via Region 13 (15 $\rightarrow$ 13 $\rightarrow$ 3). Region 13 doesn't pass the wall, so the agent should continue to Region 3.
Move to Region 3: On reaching Region 3, the agent would then move to the left, as instructed.
Region 3 directly connects to Region 4, which is the goal region.

Final Route Summary:
Region 14 $\rightarrow$ Region 15 $\rightarrow$ Region 13 $\rightarrow$ Region 3 $\rightarrow$ Region 4 (Goal)

D Detailed Results of Instruction Translation Experiment

Table. 5 shows the detailed instructions given and the routes planned by ChatGPT-4o and Claude 3.5 Sonnet. The results show that both models succeed in choosing the first region in the sequence by interpreting the instruction correctly.

Table 5: Route Planning Results Comparison between ChatGPT-4o and Claude 3.5

Case	Instruction	ChatGPT-4o Route	Claude 3.5 Route
1	Go east until pass the wall, then go north until passing the wall, turn west and reach the goal.	$7 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ (goal)	$7 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ (goal)
2	Go north a little bit until pass the wall, then go west till the end, and the goal is on the left.	$19 \rightarrow 3 \rightarrow 4$ (goal)	$19 \rightarrow 21 \rightarrow 3 \rightarrow 4$ (goal)
3	Go east until pass the wall, then go north till the end, the goal is on the left.	$9 \rightarrow 6 \rightarrow 10 \rightarrow 2 \rightarrow 16$ (goal)	$9 \rightarrow 8 \rightarrow 11 \rightarrow 2 \rightarrow 3 \rightarrow 15 \rightarrow 14 \rightarrow 16$ (goal)
4	Go straight to the west, the goal is at the north.	$20 \rightarrow 12 \rightarrow 11$ (goal)	$20 \rightarrow 12 \rightarrow 5 \rightarrow 1 \rightarrow 11$ (goal)
5	Go east til pass the wall, then go south until the end, finally go west and reach the goal.	$4 \rightarrow 3 \rightarrow 19 \rightarrow 15 \rightarrow 14 \rightarrow 1 \rightarrow 5$ (goal)	$4 \rightarrow 3 \rightarrow 19 \rightarrow 20 \rightarrow 15 \rightarrow 2 \rightarrow 12 \rightarrow 8 \rightarrow 5$ (goal)
6	Go straight to east till the end, then go north to push the movable block.	$10 \rightarrow 2 \rightarrow 15$ (block)	$10 \rightarrow 5 \rightarrow 2 \rightarrow 15$