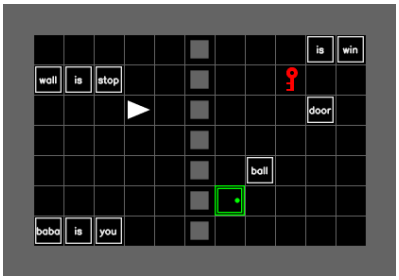
Baba Is AI: Break the Rules to Beat the Benchmark

Nathan Cloos [1] **Meagan Jens** [1] **Michelangelo Naim** [1] **Yen-Ling Kuo** [2] **Ignacio Cases** [1]
Andrei Barbu* [1] **Christopher J. Cueva*** [1]

Abstract

Humans solve problems by following existing rules and procedures, and also by leaps of creativity to redefine those rules and objectives. To probe these abilities, we developed a new benchmark based on the game Baba Is You where an agent manipulates both objects in the environment and rules, represented by movable tiles with words written on them, to reach a specified goal and win the game. We test three state-of-the-art multimodal large language models (OpenAI GPT-4o, Google Gemini-1.5-Pro and Gemini-1.5-Flash) and find that they fail dramatically when generalization requires that the rules of the game must be manipulated and combined.

1. Introduction



Plan: break[**wall is stop**], make[**door is win**], goto[**door**]

Figure 1. **Environment based on the puzzle game Baba Is You.**

Humans demonstrate remarkable abilities in rapid learning and adaptive behavior when faced with novel environments - not only learning and following rules dictated by the environment but altering these rules to enable new outcomes. These abilities leverage two key components that we explore in this paper:

1) The ability to identify and manipulate relevant stimuli in the environment while ignoring distractor objects and rules.

2) The ability to combine previously seen rules in novel ways.

The ability to study how an agent explicitly learns rules, composes them, and crucially, makes or breaks these rules to alter how the environment and agent behaves, prompted us to develop a new benchmark environment based on the puzzle game Baba Is You. In this game, the player often controls a character named "Baba" and must navigate through the grid-based world filled with blocks, objects, and textual rules. We can think of this game as a dynamic environment where the player interacts with various objects and rules to achieve specific goals. A remarkable aspect of Baba Is You is that the rules of the game can be manipulated and rearranged by the player.

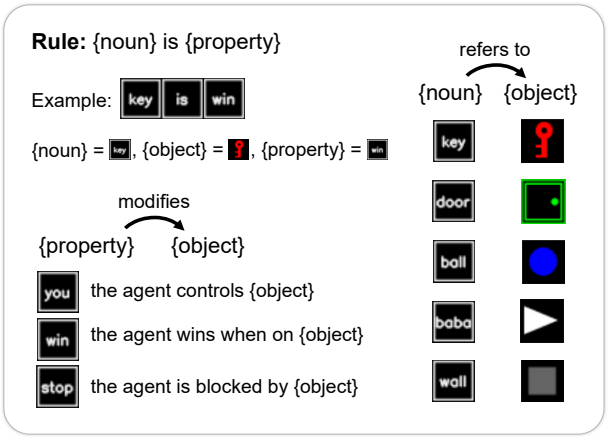


Figure 2. **Active rules in the environment modify the properties of the objects.** A rule is active when it is horizontally aligned and has the form {noun} is {property}.

Figure 1 shows an example game environment. The text blocks [baba is you] indicate the player is controlling the white triangle, i.e. the [baba] object, and can now move this object through the environment. Now let's look for the text blocks that specify how to win the game. The [is win] text blocks in the upper right of the environment are incomplete and so the agent must recognize that there is currently no way to win the game until the winning condition is specified. This is accomplished by moving one of the available text

*Equal senior authors [1]MIT [2]Department of Computer Science, University of Virginia, USA. Correspondence to: Nathan Cloos <nacloos@mit.edu>, Andrei Barbu <abarbu@mit.edu>, Christopher J. Cueva <ccueva@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

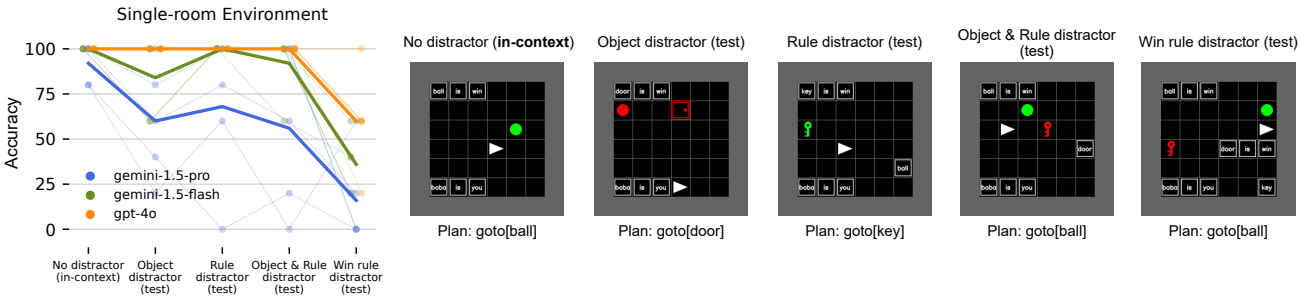


Figure 3. **Accuracy of LLMs across 5 environments testing the ability to generalize in the presence of distractors.** The task is to go to the winning object specified by the text box in the active win rule. Accuracy drops substantially on the final task where both an object and an active rule distractor are present. In this final task the irrelevant win rule does not refer to any of the objects in the environment.

block such as [door] or [ball] to create a rule for winning the game. With this specific environmental layout, a winning strategy is to push the [door] block to create the rule, [door is win], and then move the agent onto the door block, shown in green, to win the game. However, the text blocks [wall is stop] are aligned and so this rule is active and the player cannot move baba through the vertical wall of gray squares to carry out this plan. The player must first push one of the blocks in this rule out of alignment to deactivate the rule [wall is stop]. The final plan to win the game is to first break the rule [wall is stop], then make the rule [door is win], and finally move onto the door object.

As this example illustrates, this is a dynamic environment where the agent must identify the relevant objects and rules in the environment and then manipulate the environment to change or create rules for success (Figure 2). We implemented a simplified version of Baba Is You (Baba Is AI) based on the Gymnasium Minigrid environment (Chevalier-Boisvert et al., 2023).

The goal of the Baba Is AI benchmark is to evaluate the role of systematic compositionality in rule-based generalization. The core component of this benchmark is that the written commands are not only grounded in an environment, but the grounding itself can be manipulated via changing the rules of the environment. This dynamic design allows us to explore a broader notion of generalization compared to the current benchmarks.

We show results for three large language models (LLMs): GPT-4o, Gemini-1.5-Pro (May 2024), and Gemini-1.5-Flash (May 2024) (Team, 2024). We chose GPT-4o and Gemini-1.5-Pro as these models occupy the top two spots on the Chatbot Arena Leaderboard (May 2024) (Chiang et al., 2024). We also include Gemini-1.5-Flash as this model occupies an intriguing spot in the LLM ecosystem with both excellent performance and affordable price, making it an attractive option for many applications. Previous work often convert visual inputs into text before evaluating LLMs (Yao et al., 2023; Carta et al., 2023; Momennejad et al., 2023; Valmeekam et al., 2023a). Here we leverage the multi-modal ability of these models to evaluate them directly on visual inputs of the game.

2. Method

We first prompt LLMs with general text instructions to play the game. This includes a description of the possible objects and textual rule blocks in the environment, and how active rules can change object properties (as illustrated in Figure 2, with the exact prompt in Appendix A). Importantly, we specify that a rule is active only if it follows the form "object is property" and that the three rule blocks must be aligned horizontally in the environment.

Following previous work on LLM-based agents and planners (Ichter et al., 2022; Huang et al., 2022; Park et al., 2023; Valmeekam et al., 2023b; Song et al., 2023; Wang et al., 2024), we ask LLMs to operate at a higher level than the low-level control of actions in the environment. Specifically, we ask LLMs to produce high-level textual plans consisting of the following primitives: breaking an active rule, making a rule active, or moving to a specific object in the environment (see an example plan in Figure 1). We instruct LLMs that these actions can only be taken if the relevant objects and rule blocks are present in the current environment. To generate their plan, LLMs receive as visual input a static image of the initial configuration of the environment.

After providing the game instructions, we present LLMs with 10 example images and corresponding winning plans for in-context learning (Brown et al., 2020). For each example, LLMs are asked to generate reasoning steps to derive the target plan from the given image. Following the in-context examples, LLMs are prompted to describe a general algorithm to solve the environments and to apply it to unseen test environments. The test environments are specifically chosen to assess different type of generalization. We measure accuracy as the exact match between the final response of LLMs and the winning plan of the test environment. LLMs are evaluated on 5 samples for each test environment.

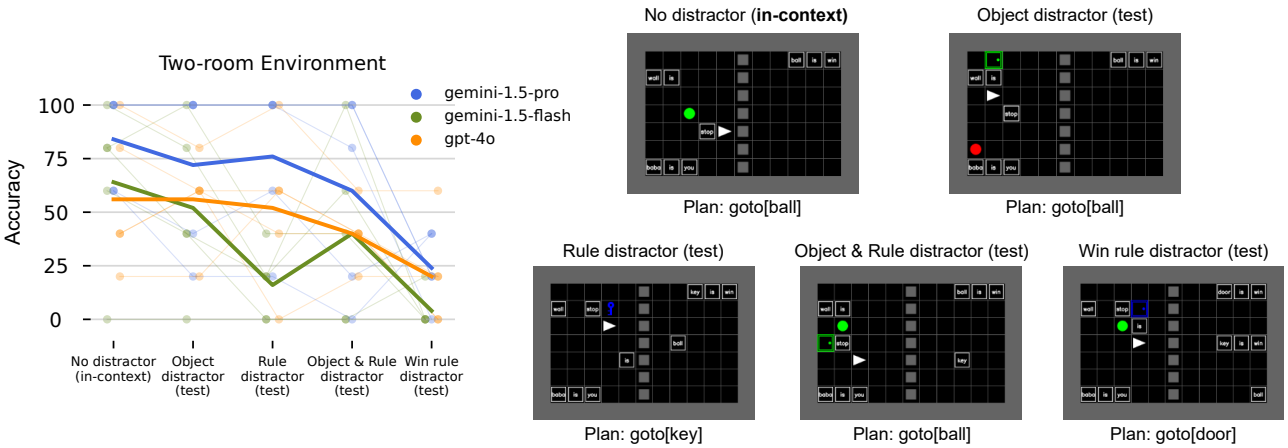


Figure 4. **The mean accuracy for all three models is lower when asked to generalize to distractors in a more complex environment.** This environment introduces a central vertical wall. However, the rule [wall is stop] is initially always inactive and so the wall has no practical impact on the movement of the agent. The task is to go to the object referred to by the active win rule (same as in Figure 3).

This entire process is repeated for 5 random seeds, each corresponding to different in-context and test examples.

3. Results

Our first tests assess the LLMs' ability to extract the most basic rule of the game from in-context examples, namely, go to the winning object, and then apply this rule in novel environments where distractors are present. Complex environments contain not only relevant stimuli but also irrelevant objects or rules; identifying the relevant from irrelevant is a crucial ability that we probe in this set of experiments.

Figure 3 shows the accuracy of the LLMs in five different environments: 1) Environments without a distractor, i.e. new random variations of the environment used during in-context learning. 2) Environments where there are now two objects but one of them is a distractor. In order to win the game, the agent must go to the object specified in the text box with the win rule, e.g. [door is win] requires the agent to go to the door. 3) Environments contain a noun block that is distracting from the active win rule. 4) Environments contain both a distractor object and noun block. 5) Environments contain both a distractor object and a noun block that is part of an active rule. The distractor rule is not relevant for the environment and so should be ignored. For example, the rightmost panel in Figure 3 shows the distractor rule [door is win] but there is no door object in the environment and so the winning strategy is to follow the other rule [ball is win] and navigate to the ball.

Impressively, GPT-4o performs with perfect accuracy on the first four environments, and as a reminder, this is while receiving visual and not textual inputs about the game. Surprisingly, Gemini-1.5-Flash outperforms Gemini-1.5-Pro, with all models showing the same trend downwards in accuracy on the final task that includes both an object and a rule distractor.

The sequence of environments used to test the LLMs in Figure 4 includes the same distractors as in Figure 3, but now all the environments include a gray vertical wall that runs down the center of the environment. The environments are always initialized with the rule [wall is stop] inactive, as the three blocks that form this rule are not horizontally aligned, and so the wall has no practical impact on the movement of the agent. However, these environments now all contain the extra distractor blocks that compose the inactive wall, and blocks about the wall rule. The mean accuracy for all three models is lower under this increased distractor load (compare Figures 3 and 4).

Compositional generalization has been studied in many contexts (Lake et al., 2017; Lake & Baroni, 2017; Ruis et al., 2020), for example, if an agent has learned to solve a task with red circles and green keys then it should generalize to red keys and green circles. In the Baba Is AI environment we can not only study these traditional forms of generalization but probe models under scenarios where the very rules of the game must be manipulated and combined. Figure 5 shows one example scenario where the LLMs are shown environments that each highlight three winning strategies and then are asked to solve a new set of environments that require a novel composition of these previously learned rules.

$$\textbf{In-context:} \begin{cases} \text{goto}\{\text{object}\} \\ \text{make}\{\text{rule}\}, \text{goto}\{\text{object}\} \\ \text{break}\{\text{rule}\}, \text{goto}\{\text{object}\} \end{cases}$$

$$\textbf{Test:} \quad \text{break}\{\text{rule}\}, \text{make}\{\text{rule}\}, \text{goto}\{\text{object}\}$$

The accuracy for all three LLMs is low. We have also alternated the four strategies shown in Figure 5 so a different three are used for in-context training and the remaining is

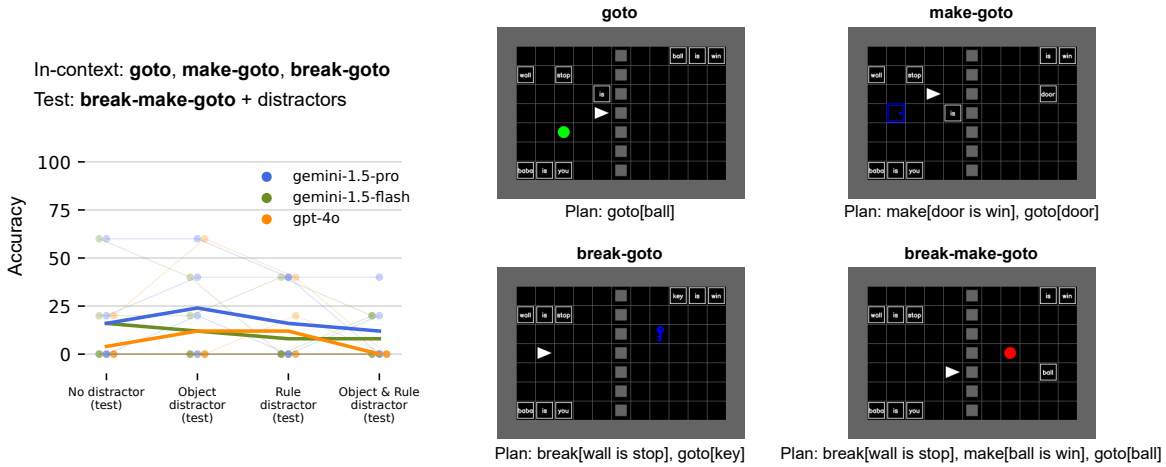


Figure 5. **LLMs generalize poorly under scenarios where the rules of the game must be manipulated and combined.** LLMs are shown environments that each highlight three winning strategies: goto{object}; make{rule} then goto{object}; break{rule} then goto{object}. Then they are asked to solve a new set of environments that require a novel composition of these previously learned rules: break{rule} then make{rule} then goto{object}.

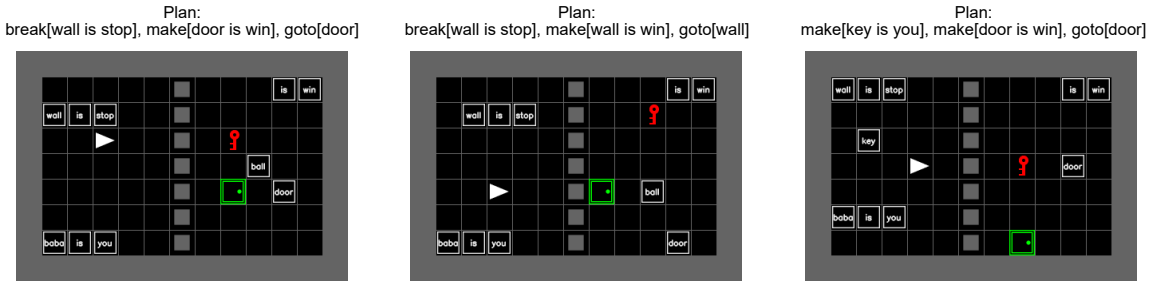


Figure 6. **Despite superficial similarities and identical objects, each environment requires distinct winning solutions, illustrating further challenges in rule manipulation and compositional generalization.**

used for testing (not shown), and accuracy remains low.

These aspects of compositional generalization across rules are particularly unique to the Baba Is AI benchmark, and the poor performance indicates that this benchmark creates meaningful generalization challenges for LLMs.

Model	Accuracy (mean ± std)
gemini-1.5-flash	20.0 ± 29.28
gemini-1.5-pro	14.67 ± 20.66
gpt-4o	17.33 ± 28.15

Table 1. Model accuracies for the environments shown in Figure 6.

4. Discussion

In order for agents to have human-like interactions with the world, they should not only be able to interact with objects but also have the capacity to understand and manipulate the rules of their environment. By defining a static set of rules that an agent must follow, many games and benchmarks have overlooked a critical capability: the ability to understand rules via rule manipulation. Therefore, the Baba Is AI benchmark explores compositional generalization under conditions in which agents can modify the rules of the environment. Figure 6 illustrates some of the further challenges in these environments. All three environments are superficially similar and contain the same objects, yet the winning solutions are different in each case (see text at the top of the figures). For example, the center environment requires the agent to break the [wall is stop] rule, then move the [wall] block to create the rule [wall is win], and finally go to one of the wall blocks to win the game. As a second example, in the environment shown in the rightmost panel of Figure 6 the rule [wall is stop] is located in the corner of the environment and so there is no way to push these blocks out of alignment and break this rule; the agent is initially trapped in the leftmost room of the environment. The agent must break the currently active rule [baba is you] and create [key is you] in order to control the key on the other side of the wall. Then the agent can use the key to create the rule [door is win] and move to the door. The accuracy on these challenging environments is low as shown in Table 1.

The errors that LLMs make in solving the Baba Is AI environments are instructive about future opportunities for

improvements (see Appendix B). LLMs make grounding mistakes: the LLM refers to an object that does not exist in the environment. LLMs make path planning mistakes: the LLM incorrectly asserts that the path to a specific object is blocked by another object, despite the path being clear in the environment.

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y. Grounding large language models in interactive environments with online reinforcement learning. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3676–3713. PMLR, 23–29 Jul 2023.

Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., and Terry, J. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhang, H., Zhu, B., Jordan, M., Gonzalez, J. E., and Stoica, I. Chatbot arena: An open platform for evaluating llms by human preference. *ArXiv*, abs/2403.04132, 2024.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022.

Ichter, B., Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., Kalashnikov, D., Levine, S., Lu, Y., Parada, C., Rao, K., Sermanet, P., Toshev, A. T., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Yan, M., Brown, N., Ahn, M., Cortes, O., Sievers, N., Tan, C., Xu, S., Reyes, D., Rettinghouse, J., Quiambao, J., Pastor, P., Luu, L., Lee, K.-H., Kuang, Y., Jesmonth, S., Jeffrey, K., Ruano, R. J., Hsu, J., Gopalakrishnan, K., David, B., Zeng, A., and Fu, C. K. Do as i can, not as i say: Grounding language in robotic affordances. In *6th Annual Conference on Robot Learning*, 2022.

Lake, B. M. and Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, 2017.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, 2017. doi: 10.1017/S0140525X16001837.

Momennejad, I., Hasanbeig, H., Vieira Frujeri, F., Sharma, H., Jojic, N., Palangi, H., Ness, R., and Larson, J. Evaluating cognitive maps and planning in large language models with cogeval. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 69736–69751. Curran Associates, Inc., 2023.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, 2023.

Ruis, L., Andreas, J., Baroni, M., Bouchacourt, D., and Lake, B. M. A benchmark for systematic generalization in grounded language understanding. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Song, C., Sadler, B. M., Wu, J., Chao, W., Washington, C., and Su, Y. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2023.

Team, G. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.

Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., and Kambhampati, S. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023a.

Valmeekam, K., Marquez, M., Sreedharan, S., and Kambhampati, S. On the planning abilities of large language models - a critical investigation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models, 2023.

A. Prompt

You will be playing a gridworld game. In this game you can change the rules of the game by moving blocks around. Blocks in the grid can be objects or rules.
A rule is active if the rule blocks are aligned horizontally and they form a valid rule. A valid rule has the form "<object> is <prop>", where <object> is the object to which the rule applies, and <prop> is the property that the object gains. Activated rules modify the rules of the objects in the grid.
- Objects: the possible objects are "baba" (white triangle), "wall", "key", "door", "ball". Objects are recognizable by their shape and color.
- Rule blocks: the rule blocks are "is", "win", "you", "stop", and rule blocks with the names of the objects. Rule blocks are represented by white boxes with white text inside.

Rule properties:
- "is": the rule block that connects the object and the property.
- "win": the rule block that indicates the object that the agent needs to reach to win the game.
- "you": the rule block that indicates the object that the agent controls.
- "stop": the rule block that indicates the object that blocks the agent from moving through it.

By default objects are not blocking, but if the rule "object is stop" is active, the object will block the agent from moving through it. Rule blocks are always pushable by the agents.
The game is always solvable and there is always a unique winning plan to solve the game.

A winning plan is composed of one or more actions (comma separated). An action can be one of the following:
- goto[<obj>]: move the agent towards the object <obj>.
- make[<obj> is <prop>]: create a rule that makes the object <obj> have the property <prop>. This action can only be taken if both the object rule block and the property rule block are in the grid and the rule is not already active.
- break[<obj> is <prop>]: remove the rule that makes the object <obj> have the property <prop>. This action can only be taken if the rule is active in the environment (i.e. the rule blocks <obj>, is, <prop> are horizontally aligned).

N x

Example:
<image>
Winning plan: goto[door]

Explain what are the reasoning steps to find the winning plan based on the given gridworld. Be as concise as possible and avoid unnecessary details.

Model: [...]

Based on the examples and the reasoning steps you have given, describe an algorithm to find the winning plan for a given gridworld. It's really important that another person can understand your algorithm and apply it to find the winning plan. Be as concise as possible and avoid unnecessary details.

Model: [...]

Apply the algorithm steps by steps to the following gridworld.
<image>

Model: [...]

Give your final answer using exactly this format: 'Winning plan: <your plan>'.

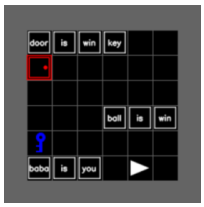
Model: [...]

Figure 7. **Prompt template.** The first part of the prompt includes general instructions describing the game and the task for the LLMs. After providing the game instructions, we present LLMs with N = 10 example grid images and corresponding winning plans, and ask LLMs to generate reasoning steps to derive the plan from the grid image. We then ask LLMs to describe an algorithm to solve the environments and to apply to new environments that systematically differ from the in-context environments.

B. Error cases

Prompt:

[...]
Apply the algorithm steps by steps to the following gridworld.



Model:

Let's apply the algorithm step by step to the given gridworld.

Step 1: Identify the Agent
- Scan the grid for the rule "baba is you".
- The rule "baba is you" is active, indicating that the white triangle (baba) is the agent controlled by the player.
- Position of baba: (5, 4).

Step 2: Identify the Winning Condition
- Scan the grid for any active rule of the form "<object> is win".
- There are two active rules: "door is win" and "ball is win".
- Positions:
  - Door (red square): (0, 0).
  - Ball (blue circle): (3, 2).

Step 3: Plan the Movement
- Check if there are any obstacles between the agent and the winning objects.
- The path to the door (red square) is blocked by the key (blue key) at (1, 1).
- The path to the ball (blue circle) is clear.

Step 4: Execute the Plan
- Since the path to the ball is clear, plan a direct path to the ball.

Generate the Plan
- The action to achieve the winning condition is to move the agent towards the ball.

Output the Plan
- Winning plan: `goto[ball]`.

Correct plan: goto[door]

Figure 8. **Two common types of mistakes observed in the reasoning of LLMs**, illustrated here for GPT-4o when tested on the single-room environment with an additional win rule distractor. (i) Grounding mistake: the LLM refers to an object that does not exist in the environment. In this example, the LLM mentions a ball (Step 3, third bullet point in the model's answer), specifying that is it a blue circle in parenthesis, even though no such ball is present. (ii) Path planning mistake: the LLM incorrectly asserts that the path to a specific object is blocked by another object, despite the path being clear in the environment. In this instance, the LLM claims that the path to the door is blocked by the key, even though it is not the case.