Policy Learning with a Language Bottleneck

Megha Srivastava
megha@cs.stanford.edu
Stanford University

Cédric Colas
ccolas@mit.edu
MIT

Dorsa Sadigh
dorsa@cs.stanford.edu
Stanford University

Jacob Andreas
jda@mit.edu
MIT

Abstract

Modern AI systems such as self-driving cars and game-playing agents achieve superhuman performance, but often lack human-like features such as generalization, interpretability and interoperability with humans. Inspired by the rich interactions between language and decision-making in humans, we introduce *Policy Learning with a Language Bottleneck* (PLLB), a framework enabling AI agents to generate linguistic rules that capture the strategies underlying their most rewarding behaviors. PLLB alternates between a *rule generation* step guided by language models, and an *update* step where agents learn new policies guided by rules. In a two-player communication game, a maze solving task, and two image reconstruction tasks, we show that PLLB agents are not only able to learn more interpretable and generalizable behaviors, but can also share the learned rules with human users, enabling more effective human-AI coordination.

1 Introduction

As AI systems play an increasingly central role in automation, there is a growing need for them to better model and emulate human behavior. Such systems must be interpretable, and both act and generalize in human-predictable ways so that users can effectively interact with them. Unfortunately, many of today's AI systems do not meet these standards. Self-driving cars or game-playing agents may achieve super-human performance but lack interpretability (McIlroy-Young et al., 2020) and often act unpredictably, especially outside their training distribution (Wang et al., 2023). Meanwhile, humans acquire most of their skills and knowledge from others, often using language—via instructions, advice, or explanations that improve their decision-making capabilities (Carruthers & Boucher, 1998; Mesoudi & Thornton, 2018). Language acts as a *communicative* medium, enabling us to teach, learn from, and coordinate with others to solve complex problems. It also supports other *cognitive* functions: it allows us to represent abstract concepts (Hesse, 1988; Lakoff & Johnson, 2008), and plan (Vygotsky, 1965; Clark, 1998); it guides our attention (Waxman, 1994; Yoshida & Smith, 2003), and prompts relational thinking (Gentner & Loewenstein, 2002).

Consider a driver learning novel social conventions (e.g., triangle-shaped stop signs). While adapting to the environment, they might verbalize strategies to themselves (e.g., *If the sign is triangular, I should stop*, a cognitive use), or transmit this convention to others (e.g., telling a friend *In Japan, stop signs are triangles*, a communicative use). Representing knowledge with language helps humans solve problems and transmit solutions by encoding abstract problem structures that facilitate learning and generalization (Boutonnet & Lupyan, 2015; Chopra et al., 2019; Tessler et al., 2021). Although the recent literature contains many examples of AI systems leveraging language-based representations or feedback, these often rely on external supervision: humans in the loop or hard-coded feedback (see related work in Luketina et al., 2019; Colas et al., 2022). This paper aims to develop more human-like AI systems that not only *use* language-based supervision but also *generate their own language-based feedback* to leverage the communicative and cognitive functions of language.

We introduce *Policy Learning with a Language Bottleneck* (PLLB), a framework providing artificial agents the ability to generate linguistic rules that capture the strategies underlying their most

rewarding behaviors. As shown in Figure 1, PLLB alternates between a `gen_rule` step that explains the agent's most rewarding behaviors by prompting a language model (LM) with contrastive episodes, and an `update` step that learns a new policy conditioned on these rules. PLLB helps agents learn more human-like policies, which we examine across four diverse tasks. **They perform better**: in two image reconstruction tasks, PLLB agents generate instructions increasing the listeners' performance compared to non-linguistic baselines (Section 7). **They generalize better**: in a maze task, agents uncover the abstract problem structure and generalize better to similar mazes (Section 6). **They are more interpretable**: in a coordination task, agents converge on the most human-interpretable policy when several optimal ones exist (Section 5). **They are more interoperable**: in maze and image reconstruction tasks, humans achieve achieve better rewards when interacting with PLLB agents (Sections 6 and 7).

2 Background & Related Work

Language facilitates cooperation and coordination between humans and machines via instructions (Hermann et al., 2017; Chevalier-Boisvert et al., 2019), advice (Watkins et al., 2021), explanations (Zhong et al., 2020; Lampinen et al., 2022), or the formation of conventions (Hawkins et al., 2020; Hu & Sadigh, 2023). Such communicative functions increase the fidelity and breadth of cultural transmission — a process of social learning that underlies human ecological success (Mesoudi & Thornton, 2018). Language also augments a learner's cognitive abilities, and helps RL agents represent more abstract goals (Jiang et al., 2019), generalize better (Hill et al., 2020; Colas et al., 2020; Wong et al., 2021), explore more efficiently (Tam et al., 2022; Klissarov et al., 2023) and decompose complex goals into simpler ones (Chen et al., 2021; Ahn et al., 2022; Hu et al., 2022; Sharma et al., 2021; Hu & Clune, 2023). We extend these benefits to agents that learn from *self-generated* linguistic feedback.

Generating linguistic rules for oneself is a form of *inner speech*. Inner speech is seen as the internalization of the social speech generated by caretakers to help children solve problems (Vygotsky, 1965; Luria, 1959). As a result, it is thought to support our capacities for complex, long-term behaviors (Vygotsky, 1965; Luria, 1959; Hermer-Vazquez et al., 2001; Spelke, 2003). Meanwhile, AI agents endowed with forms of inner speech (explanations, descriptions or subgoals) have been found to perform and generalize better than agents trained with purely neural representations (Wong et al., 2021; Lampinen et al., 2022; Roy et al., 2022; Hu & Clune, 2023). Instead, PLLB generates language in an unsupervised way.

Recent works propose guiding LMs' reasoning by prompting them to step through"thoughts" sequences (Wei et al., 2023; Yao et al., 2022; Li et al., 2023b). While Shinn et al. (2024) additionally leverage linguistic "self-reflections" of task feedback, they are limited to language agents that learn without traditional policy or value-based RL. In contrast, PLLB can both increase the interoperability of initially uninterpretable systems (low level policies) and improves the generalization capacities of an RL agent.

3 The Language Bottleneck

PLLB builds on the standard RL framework to train more human-like agents to solve decision-making tasks (e.g., solving a maze), which we formalize as Markov decision processes (S, A, f, R, T) with reward function $R : S \times A \to \mathbb{R}$ (e.g., speed) over states S (e.g., cell coordinates) and actions A (e.g., directions), time horizon T, and a transition function $f : S \times A \to S$ that maps state–action pairs (s, a) to next states s. Standard RL involves training a policy π to maximize expected reward. This is usually done by: (1) collecting data with the current policy: $D \leftarrow \pi_i$ and (2) updating the policy using the data: $\pi_{i+1} \leftarrow \texttt{update}(\pi_i, D)$, where `update` is an RL algorithm (Sutton & Barto, 2018). We introduce a *language bottleneck* between data collection and policy update. We extract linguistic rules describing past rewarding behaviors and then use them to regularize the policy's behavior in the next learning iteration (e.g., *I should go right in blue cells*, see Figure 1). This

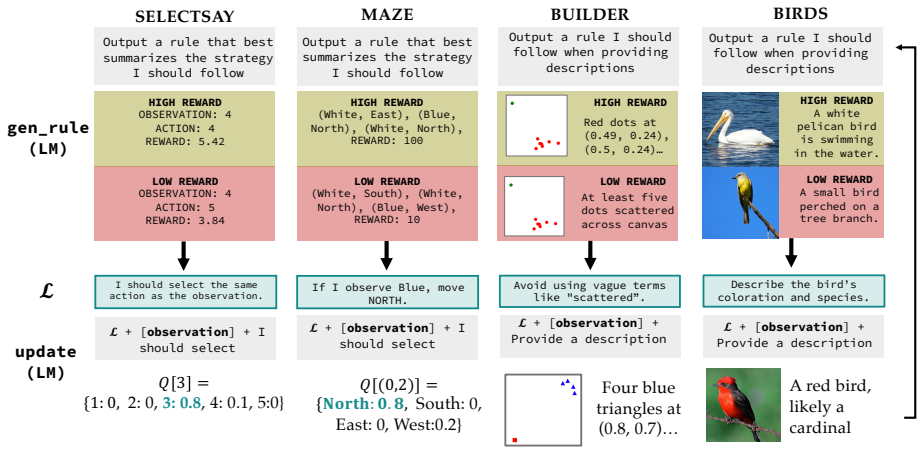

Figure 1: Across both multi-step decision making and visual tasks, PLLB iterates between `gen_rule`, which extracts a linguistic rule ($\mathcal{L}$, blue) by contrasting high and low reward episodes from experience (green/red boxes), and `update`, which updates an agent's policy conditioned on $\mathcal{L}$.

algorithm thus alternates between three steps: (1) data collection, $D_i \leftarrow \pi_i$; (2) language bottleneck generation, $\mathcal{L}_i \leftarrow \texttt{gen\_rule}(D_i)$ and (3) policy updating, $\pi_{i+1} \leftarrow \texttt{update}(\pi_i, D_i, \mathcal{L}_i)$.

Rule Generation (`gen_rule`) Using all the experience D_i collected by the policy π_i in the current iteration, the `gen_rule` step aims at inferring an abstract rule $\mathcal{L}_i$ that best explains the agents' successful behaviors: $\mathcal{L}_i \leftarrow \texttt{gen\_rule}(D)$. Inspired by Dunlap et al. (2023), we prompt an LM with contrastive episodes from D_i (top-N highest vs. top-N lowest total rewards) and ask it to *provide the rule that should be followed to obtain high rewards* (see first row of Figure 1 and Appendix Section A.6 for full prompts).

Rule-Guided Policy Update (`update`) Given a rule $\mathcal{L}_i$, `update` produces a new policy $\pi_{i+1} \leftarrow \texttt{update}(\pi_i, D_i, \mathcal{L}_i)$ that is better aligned with $\mathcal{L}_i$. There exist many methods for leveraging language supervision, typically provided by experts, to modify agents' policies. One approach is InstructRL (Hu & Sadigh, 2023), which regularizes the learned policy with another policy induced by the linguistic rule $\pi_{\mathcal{L}}$. In the maze example from Figure 1, if the rule is *go right on every blue cell*, the induced $\pi_{\mathcal{L}}$ should assign probability 1 to *right* in blue cells, but equal probabilities to all actions in other situations. In the Q-learning algorithm, InstructRL adds a *regularizing term* (orange) to the standard update rule, where λ controls regularization strength:

$$Q^{\theta}(s_t, a_t) \leftarrow r_{t+1} + \gamma Q^{\theta}(s_{t+1}, a_{t+1}), \text{ where } \quad a_{t+1} = \arg\max_a [Q(s_{t+1}, a) + \lambda \log \pi_{\mathcal{L}}(a \mid s_t)].$$

How do we induce $\pi_{\mathcal{L}}$ from $\mathcal{L}$? Since $\mathcal{L}$ is expressed in language, we prompt an LM with both the rule and the agent's current state s_t to generate the next action and obtain a probability distribution over admissible actions. In domains where the policy can be directly implemented in text by an LM, conditioning the policy on the rule $\mathcal{L}$ by adding it to the prompt $\pi_{\mathcal{L}}$ can steer the agent's behavior.

4 Experiment Set-Up

We run experiments on a set of four diverse tasks to showcase the capacity of PLLB to train more human-like agents. Our domains include a simpler two-player communication game called SelectSay (Section 5), a Maze solving task (Section 6), and two collaborative image reconstruction tasks with either synthetic (Builder) or natural (Birds) images (Section 7). For all tasks, we include hyperparameter details (Appendix A.2), the exact prompts used in `gen_rule` and `update` (Appendix A.6), and examples and qualitative analysis of generated rules (Appendix A.7 and A.8).

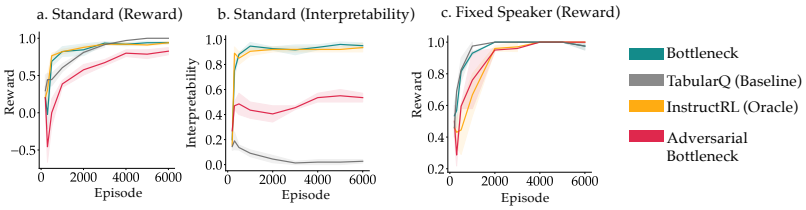

Figure 2: SELECTSAY. a) **Bottleneck** agents learn as fast as **TabularQ** and **InstructRL** and faster than agents using adversarial rules **Adversarial**. b) Unlike **TabularQ**, they learn human-interpretable policies without relying on an external instruction like **InstructRL**. c) When faced with speakers enforcing a non-interpretable policy, **Bottleneck** converges faster than **InstructRL**.

5 SelectSay

As a proof-of-concept, we first consider SELECTSAY, a simple collaborative game introduced in Hu & Sadigh (2023). Here, a *speaker* can see a hidden set of five balls including two blue ones and must help a *listener* find the blue balls by communicating with them only via numbers (Appendix Figure 7). The speaker and listener could converge on any bijective convention between messages and balls, which we observe with RL agents (multiple optimal policies). Humans, on the other hand, prefer the intuitive mapping $1 \rightarrow 1$, $2 \rightarrow 2$, etc. or *human-interpretable policy*. Hu & Sadigh (2023) showed that regularizing the listener's policy with the instruction "I should select the same number as my partner", successfully guided the two RL agents to the human-interpretable policy.

Task Overview. As in Hu & Sadigh (2023), the speaker and listener are RL agents trained with Q-Learning that receive positive rewards when the listener collects blue balls, negative rewards otherwise. After training both agents for 200 episodes, we prompt `llama-2-70b-chat` with a contrastive set of high- and low-reward episodes to generate a rule $\mathcal{L}$, e.g., *I should choose action 4 whenever the observation is 2, 3, 4, or 5.* (`gen_rule`, see Figure 1). We generate new rules every 500 training episodes. In between, we regularize the listener's policy with a another policy $\pi_{\mathcal{L}}$ induced from the current rule $\mathcal{L}$. This is done by obtaining a distribution from applying *softmax* on the LM's logits (see method in Section 3) across all possible actions. In the following experiments, we compare our **Bottleneck** method, with an **Adversarial** version (corrupted rule), a **TabularQ** baseline, as well as an **InstructRL** upper bound using the predefined instruction from (Hu & Sadigh, 2023).

5.1 PLLB helps learn human-interpretable policies

We first compare all methods along two dimensions: reward and their *interpretability* measured as the proportional (0 to 1) similarity between the listener's policy and the optimal human-interpretable policy described above. All methods but **Adversarial** quickly solve the task (Figure 2a), showing that corrupted rules hinder learning. **Bottleneck** and **InstructRL** both converge on the human-interpretable policy thanks to their linguistic rules while **TabularQ** — deprived of such rule — does not (Figure 2b). Interestingly, we note that **Adversarial** policies are more interpretable than **TabularQ**, suggesting that even corrupted language may provide biases towards more human-like behavior. We observe that generated rules converge towards describing the human-interpretable policy (e.g., *I should follow the strategy of choosing the same action as Agent 1*, see A.7).

5.2 PLLB can learn counter-intuitive policies

A potential confound in the preceding experiment is whether PLLB helped converge to the human-interpretable policy because this is the *only* behavior an LM is able to describe. To test this hypothesis we fixed the speaker to use a counter-intuitive policy using a random mapping between balls and messages (e.g., $1 \rightarrow 3$, $5 \rightarrow 2$, etc.). Here, a listener following the human-interpretable policy would fail. In Figure 2c we observe that **InstructRL** and **Adversarial**, both regularized by misaligned rules, converge at a slower rate than both **TabularQ** (no rule) and **Bottleneck**.

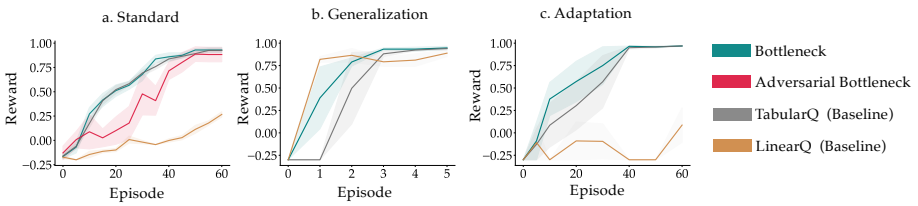

Figure 3: Results in MAZE. a) **Bottleneck** agents learn as fast as the non-linguistic **Baseline agents**, but faster than **Adversarial** and **LinearQ** agents. b) When faced with a new maze with similar structure, **Bottleneck** agents learn faster than **TabularQ** (which does not perceive color) and **LinearQ**(which does, but learns slower). c) When faced with a maze of a different structure, **Bottleneck** agents adapt swiftly while **LinearQ** cannot recover.

6 Maze

We next study the MAZE task, where agents must solve a maze using four directional actions (N/S/E/W), but can leverage generalizable environment hints (e.g color) about the optimal path.

Task Overview. We generate random 7x7 mazes with `gym-maze`[1], where agents are penalized by the number of steps they took to reach the goal. We add structure to all mazes by coloring cells for which the optimal action is *south* (red) or *north then east* (blue) with 50% probability. Agents that leverage color information better generalize to new mazes where the optimal action sequence differs. The first iteration of PLLB correspond to training a standard RL algorithm to obtain π_1, in our case a tabular Q-learning agent (**TabularQ**). After the agent observes two solved mazes, we run `gen_rule` by prompting a `llama-2-70b-chat` LM with contrastive episodes and a task description. This gives us a linguistic rule $\mathcal{L}_1$ (e.g., *Upon observing RED, take SOUTH*) that we can use to `update` the policy. We first induce the regularizing policy $\pi_{\mathcal{L}_1}$ with rule $\mathcal{L}_1$ by obtaining a probability distribution over the 4 actions from the LM, and then run the RL algorithm for 5 episodes to obtain the new policy π_2 (Section 3). We repeat these steps every 5 episodes of interactions with the environment. Baselines include the base tabular Q-learning algorithm without language bottleneck (**TabularQ**) and a variant of **Bottleneck** generating rules from reward-randomized episode samples (**Adversarial**). We additionally evaluate **LinearQ**, an agent learning a linear model Q-function with an additional feature for cell color. We train **LinearQ** with a batch update of size 10 and learning rate 0.001, after performing a hyperparameter sweep, and found it took much longer to converge than other methods.

PLLB learns more generalizable policies Figure 3a shows that learning a valid rule (**Bottleneck**) does not increase learning speed over **TabularQ** (already quite fast), but using a corrupted rule (**Adversarial**) or learning from linear features (**LinearQ**) slows down learning. We next replace the maze by another 7x7 maze sharing the same underlying color semantics, but with different optimal action sequences. For fair comparison, we start with the fully converged policy for each method (requiring 100 episodes for **LinearQ**). Figure 3b shows that **Bottleneck** leverages the learned rule and adapts to the new maze more effectively than both **TabularQ** and a fully converged **LinearQ** agent. While **TabularQ** cannot generalize because it does not perceive colors, **LinearQ** generalizes faster at first, but converges slower. Generated rules improve over time to better capture the structure of the maze (e.g., *if I observe BLUE, then take the NORTH action*; see other examples in Appendix A.7). Across all trials, 100% of the final rules mention the red $\rightarrow$ *south* rule and 60% uncovered the more complex blue $\rightarrow$ *north-then-east* rule. Finally, we find that a policy that does not update Q-values (i.e. purely implemented via the LM) achieves a 0% success rate within the same time limit, even when using the final generated rules from PLLB.

PLLB learns adaptable policies In Figure 3c, we run this same experiment on a maze with a different underlying structure (red now indicates *west* while blue indicates *east then south*). Although the rule **Bottleneck** learned does not apply anymore, it can still adapt faster than the baselines.

[1]https://github.com/MattChanTK/gym-maze

The language bottleneck can indeed quickly capture the new structure of the task and regularize learning to steer adaptation. Here **LinearQ** seems to have overfit to the first maze and struggled to adapt. We find that all trials end with rules capturing the new mapping (100% red → *west*, 50% blue → *east-then-south*). Overall, these results show that the language bottleneck supports human-like cognitive functions by finding a tradeoff between the efficiency of TabularQ and the generalization capabilities LinearQ while remaining more adaptable.

PLLB is more interpretable and inter-operable Can generated rules be useful to humans as well? We asked 50 university students to solve a 7x7 maze in as few steps possible. They could only observe the cells or walls they had already visited or bumped into (see Figure 10 in Appendix). We split them into three groups: a **Control** group receiving no assistance, and two others receiving information about a *similar* but different maze sharing the same color semantics. The **Visual** group is shown a visual representation of the optimal policy (arrows per cell, see Appendix Figure 9), while the **Bottleneck** group is provided a random PLLB rule $\mathcal{L}$, allowing us to evaluate how generalizable the two different aids are to a new maze. Participants using PLLB rules solve the new maze with fewer steps than others and find this aid more useful than the visual one on average (statistically significant at level 0.05 with two-sided Mann-Whitney test after Bonferroni correction, see Figure 6). While the visual aid contained non-transferable information (optimal actions in all non-colored cells), PLLB rules focus on the useful and usable information learned by the previous agent.

7 Collaborative Image Reconstruction

PLLB is not restricted to the training of RL agents. This section introduces two collaborative image reconstruction tasks inspired by the collaborative assembly task of McCarthy (2020). BUILDER and BIRDS both consider two agents: a *speaker*, who can see a hidden target image, and a *listener*, who must accurately reconstruct the target image based on a description from the speaker. Both agents aim to converge to a description style leading to high reconstruction accuracy. We consider both synthetic images built from sequences of discrete actions by the listener (BUILDER), and natural images of birds that the listener reconstructs using a text-to-image generation model (BIRDS).

Dataset. In BUILDER, we construct a dataset of synthetic images with a variable number of shapes (triangle, square, circle) in different colors (magenta, blue, red, green) and 2D-grid locations. Each image is created by a sequence of discrete actions, representing a particular combination of shape, color, and location (e.g., `[ACT12 ACT2 ACT4]`). In BIRDS, we construct a dataset of natural images by selecting 5 images for each of 10 species from the CUB-200-2011 dataset (Wah et al., 2011).

Reward. In BUILDER, because target and reconstructed images are defined by the action sequences that generate them, we measure task success with the Levenshtein similarity between the (sorted) corresponding action sequences. However, the listener in the BIRDS environment outputs images using a text-to-image model that exhibit a variety of properties. We therefore introduce reward functions corresponding to three different properties for BIRDS: **Color**, **Background**, and **Species**. To evaluate a listener's reconstructed image x_r with respect to a particular target image x_t and reward function, we create a *contrast* set C by selecting the three images in CUB-200-2011 dataset that have a different value for the target reward property (e.g., color), but are most similar with respect to values for all other annotated properties. We then define reward r as $r = \sum_{c \in C} d(\mathsf{CLIP}(x_r), \mathsf{CLIP}(x_c)) - d(\mathsf{CLIP}(x_r), \mathsf{CLIP}(x_t))$, where CLIP refers to image embedding method of Radford et al. (2021). For both tasks, reward is reported over a held-out validation set.

Models. We use the llama-2-70b-chat LM as our speaker for BUILDER, representing images in raw text that list all shapes' type, color, and exact coordinates individually in sequence. Wwe use the open source Llava VLM as our speaker for BIRDS (Liu et al., 2023). We implement the listener for BUILDER with BART, a neural sequence-to-sequence model pre-trained on English text by Lewis et al. (2019), which we fine-tune at each episode on a training set of (description, action sequence) pairs using descriptions provided by the speaker. Meanwhile, we use the Stable Diffusion

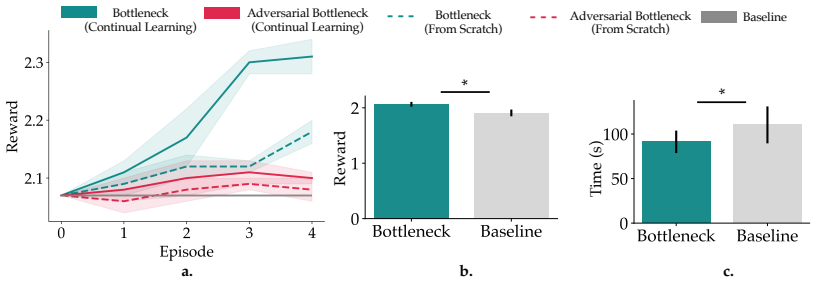

Figure 4: In BUILDER, (a) performance of **Bottleneck** listener improves upon **Baseline** descriptions over time for both **From Scratch** and **Continual Learning**, unlike an (**Adversarial**) listener. In a user study ($n = 20$), human listeners reconstructed target images with higher accuracy (b.), and less time (c.), when given instructions generated using **Bottleneck** vs. original **Baseline** descriptions.

text-to-image diffusion model as the listener for BIRDS. For both tasks, we consider two settings: **From Scratch**, where the listener is initialized using the original pre-trained model weights at every episode, and **Continual Training**, where the listener is continuously updated over time using descriptions by the speaker. We therefore split our datasets into separate held-out sets for training, early stopping, selecting samples for `gen_rule`, and evaluation.

Implementation. We first generate base image descriptions for both tasks by providing the speaker a prompt containing a general task description and target image, but without any rule $\mathcal{L}$. We next evaluate the base descriptions using each task's respective reward functions, and select the (image, description) pairs with the 5 highest and lowest rewards. Using the prompts shown in Section A.6, we implement `gen_rule` using the `llama-2-70b-chat` and `llava` models for BUILDER and BIRDS, respectively. For BIRDS, we observe that output rules $\mathcal{L}$ are reward-specific: an example rule we get for the **Species** reward is: *"Identify the bird's species if possible, and include any distinctive characteristics that set it apart from other birds."*, while an example rule we get for the **Background** reward is: *"Describe background of the image, such as the presence of snow, water, ..."*. We implement `update` for both tasks by appending the output $\mathcal{L}$ to the original prompt to the speaker, as shown in Figure 1, using the speaker's output as the new description for a given target. We repeat this cycle of eliciting rule $\mathcal{L}$, appending $\mathcal{L}$ to the same prompt used to generate the base instructions in order to re-label our data with new instructions, and training and evaluating the listener model for a total of 5 iterations for BUILDER and 3 iterations for BIRDS.

PLLB helps speakers provide more usable instructions For both tasks, PLLB (**Bottleneck**) listeners outperform listeners trained from uninformed (**Baseline**) or misinformed (**Adversarial**) speakers (Figures 4 and 5, top row). This holds true for all three different reward functions in BIRDS, as well as both training settings, with **Continual Learning** enabling **Bottleneck** to have an even stronger improvement over **Baseline** image descriptions by leveraging task experience. **Continual Learning** does not significantly help the **Adversarial** method, showing that the linguistic rules $\mathcal{L}$ in **Bottleneck** capture successful task strategies. The success of the **From Scratch** setting can be interpreted as the speaker generating abstract rules to guide its own learning, leading to improved performance and showing evidence for a cognitive use of language.

Finally, we observe that the generated rules $\mathcal{L}$ encourage the speaker to reduce vague anguage, and describe properties relevant to the corresponding underlying reward function in BIRDS, although this might take several cycles to fully converge (see example rules in Appendix Section A.7). Furthermore, rules $\mathcal{L}$ become more complex over time, and across different trials we observe path dependency where different trials converge to different rules (e.g., (x, y) vs. $x = 0.xx, y = 0.yy$ for coordinates in BUILDER). Overall, these rules help guide the speaker and listener agents in both tasks to iteratively improve image reconstruction over time, which can also be seen qualitatively in Figure 8 (Appendix).

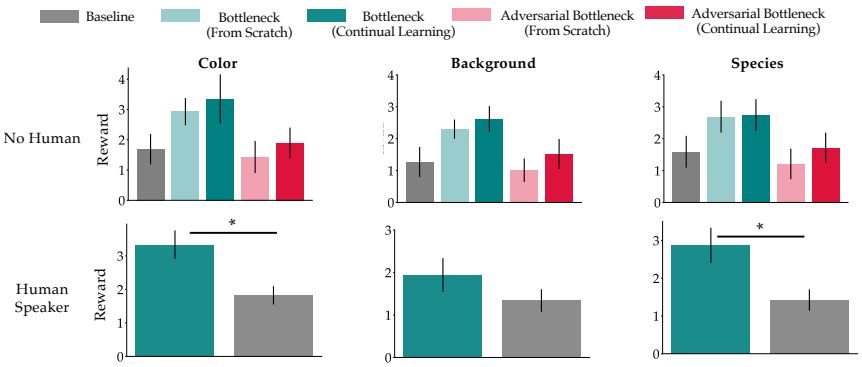

Figure 5: For BIRDS, rules generated after 3 iterations of PLLB help automated (top) and human (bottom) speakers provide stronger descriptions for all three possible reward functions (columns). **Continual learning** (dark teal, red) further improves reward over listener agents learning from scratch (light teal, red), but not enough to help **Adversarial** outperform baseline descriptions.

PLLB can collaborate with human listeners We next conduct a human subject study for BUILDER, where we replace the listener with study participants at the end of one iteration of PLLB, and evaluate how accurately and quickly they can reconstruct images from a held-out test set. We recruit 20 crowdworkers on Prolific to reconstruct 5 images using descriptions from a speaker following a **Bottleneck** rule $\mathcal{L}$, and 5 images using the original **Baseline** descriptions. We provide participants an interface that includes a canvas and buttons controlling shape and colors (Section A.4), which map to the action sequences for a target image. Participants using **Bottleneck** descriptions built target images faster and more accurately than participants using control descriptions (Bonferroni-corrected Wilcoxon signed-rank test, $p < 0.05$, Figure 4). 55% of them preferred **Bottleneck** descriptions when asked in a post-study survey, describing them as *"more direct and less ambiguous"*. 10% answered that both descriptions types were equally easy to follow while remaining participants preferred **Baseline** descriptions because they *they gave more flexibility*, showing a preference not captured in our reward function.

PLLB collaborates with human speakers Can humans benefit from rules generated by PLLB? We asked 12 Prolific crowdworkers to act as speakers and provide descriptions for images in BIRDS. Half of the participants were provided rules inferred for one of the three reward functions ($\mathcal{L}_3$) while the other half were not. Figure 5 (bottom) shows that listeners instructed by human speakers provided with PLLB rules outperform those instructed by uninformed human speakers (**Baseline**) (Bonferroni-corrected t-tests $p < 0.05$ for **species** and **colors**). Participants who were not given a rule provided less focused descriptions e.g., *A barbed wire is an uncomfortable stop for a bird*, while those provided PLLB rules generated more useful descriptions: *Bird on barbed wire. Bright red chest, red at top of head, black wings and beak* for the same target image (see other examples in Appendix Tab. 4). Thus, PLLB agents transmit their experience to humans in the same task.

8 Limitations & Discussion

By inferring the rule explaining its best past behaviors, PLLB becomes more interpretable, generalizable, and inter-operable with humans, which are important properties as AI systems become increasingly embedded in society. However, in PLLB, rule generation requires converting agent histories into representations the LM can handle (e.g., text or images), which may prevent the use of long-horizon sensorimotor trajectories (e.g., robotic systems). Rule generation can also fail when the LM overfits, and is subject to well-studied issues of biases in modern LMs. An interesting direction for future work is to apply PLLB to tasks with complex reward functions that include hard-to-articulate human preferences. In applications like image captioning for accessibility (Nie et al., 2020) and personalization of language models (Li et al., 2023a), linguistic rules might improve transparency and predictability even in cases where users find it difficult to describe their own goals.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. 2022.

Bastien Boutonnet and Gary Lupyan. Words jump-start vision: A label advantage in object recognition. *Journal of Neuroscience*, 35(25):9329–9335, 2015. doi: 10.1523/JNEUROSCI.5111-14.2015. URL https://www.jneurosci.org/content/35/25/9329.

Peter Carruthers and Jill Boucher. *Language and Thought*. Cambridge University Press, 1998.

Valerie Chen, Abhinav Gupta, and Kenneth Marino. Ask your human: Using human instructions to improve generalization in reinforcement learning. *Proc. of ICLR*, 2021.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *Proc. of ICLR*, 2019.

Sahil Chopra, Michael Henry Tessler, and Noah D Goodman. The first crank of the cultural ratchet: Learning and transmitting concepts through language. *Proc. of CogSci*, 2019.

Andy Clark. Magic Words: How Language Augments Human Computation. In Peter Carruthers and Jill Boucher (eds.), *Language and Thought*. Cambridge University Press, 1 edition, 1998. ISBN 978-0-521-63108-2 978-0-521-63758-9 978-0-511-59790-9.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter F. Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity driven exploration. *Proc. of NeurIPS*, 2020.

Cédric Colas, Tristan Karch, Clément Moulin-Frier, and Pierre-Yves Oudeyer. Language and culture internalization for human-like autotelic ai. *Nature Machine Intelligence*, 4(12):1068–1076, 2022.

Lisa Dunlap, Yuhui Zhang, Xiaohan Wang, Ruiqi Zhong, Trevor Darrell, Jacob Steinhardt, Joseph E. Gonzalez, and Serena Yeung-Levy. Describing differences in image sets with natural language, 2023.

Dedre Gentner and Jeffrey Loewenstein. *Relational Language and Relational Thought*. Erlbaum, 2002.

Robert D Hawkins, Minae Kwon, Dorsa Sadigh, and Noah D Goodman. Continual adaptation for efficient machine communication. *Proc. of ACL*, 2020.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. Grounded language learning in a simulated 3d world. 2017.

Linda Hermer-Vazquez, Anne Moffet, and Paul Munkholm. Language, space, and the development of cognitive flexibility in humans: The case of two spatial memory tasks. *Cognition*, 79(3):263–299, 2001.

Mary Hesse. The Cognitive Claims of Metaphor. *The Journal of Speculative Philosophy*, 1988.

Felix Hill, Andrew K. Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L. McClelland, and Adam Santoro. Emergent systematic generalization in a situated agent. *Proc. of ICLR*, 2020.

Hengyuan Hu and Dorsa Sadigh. Language instructed reinforcement learning for human-AI coordination, 2023.

Hengyuan Hu, David J Wu, Adam Lerer, Jakob Foerster, and Noam Brown. Human-AI coordination via human-regularized search and learning, 2022.

Shengran Hu and Jeff Clune. Thought cloning: Learning to think while acting by imitating human thinking. *Proc. of NeurIPS*, 36, 2023.

Yiding Jiang, Shixiang Gu, Kevin Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. *Proc. of NeurIPS*, 2019.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. 2023.

George Lakoff and Mark Johnson. *Metaphors We Live By*. University of Chicago press, 2008.

Andrew K. Lampinen, Nicholas A. Roy, Ishita Dasgupta, Stephanie C. Y. Chan, Allison C. Tam, James L. McClelland, Chen Yan, Adam Santoro, Neil C. Rabinowitz, Jane X. Wang, and Felix Hill. Tell me why! – explanations support learning of relational and causal structure. *Proc. of ICML*, 2022.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

Belinda Z Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. Eliciting human preferences with language models. 2023a.

Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia, and Brian Ichter. Chain of code: Reasoning with a language model-augmented code emulator. 2023b.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. 2019.

Aleksander R Luria. The directive function of speech in development and dissolution. *Word*, 15(2), 1959.

William McCarthy. Emergence of compositional abstractions in human collaborative assembly. 2020. URL https://api.semanticscholar.org/CorpusID:233179096.

Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '20. ACM, August 2020. doi: 10.1145/3394486.3403219. URL http://dx.doi.org/10.1145/3394486.3403219.

Alex Mesoudi and Alex Thornton. What is cumulative cultural evolution? *Proceedings of the Royal Society B*, 285(1880):20180712, 2018.

Allen Nie, Reuben Cohn-Gordon, and Christopher Potts. Pragmatic issue-sensitive image captioning, 2020.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

Nicholas A Roy, Junkyung Kim, and Neil Rabinowitz. Explainability via causal self-talk. *Advances in Neural Information Processing Systems*, 35:7655–7670, 2022.

Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. *Proc. of ACL*, 2021.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Proc. of NeurIPS*, 2024.

Elizabeth S Spelke. What makes us smart? core knowledge and natural language. *Language in mind: Advances in the study of language and thought*, pp. 277–311, 2003.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

Allison Tam, Neil Rabinowitz, Andrew Lampinen, Nicholas A Roy, Stephanie Chan, DJ Strouse, Jane Wang, Andrea Banino, and Felix Hill. Semantic exploration from language abstractions and pretrained representations. *Proc. of NeurIPS*, 2022.

Michael Henry Tessler, Jason Madeano, Pedro A. Tsividis, Brin Harper, Noah D. Goodman, and Joshua B. Tenenbaum. Learning to solve complex tasks by growing knowledge culturally across generations, 2021.

L.S Vygotsky. *Thought and Language*. MIT Press, 1965.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, 2011.

Tony T. Wang, Adam Gleave, Tom Tseng, Kellin Pelrine, Nora Belrose, Joseph Miller, Michael D. Dennis, Yawen Duan, Viktor Pogrebniak, Sergey Levine, and Stuart Russell. Adversarial policies beat superhuman go ais, 2023.

Olivia Watkins, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Jacob Andreas. Teachable reinforcement learning via advice distillation. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Proc. of NeurIPS*, 2021.

Sandra R Waxman. The development of an appreciation of specific linkages between linguistic and conceptual organization. *Lingua*, 92:229–257, 1994.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

Catherine Wong, Kevin Ellis, Joshua B. Tenenbaum, and Jacob Andreas. Leveraging language to learn program abstractions and search heuristics. *Proc. of ICML*, 2021.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. 2022.

Hanako Yoshida and Linda B Smith. Sound symbolism and early word learning in two languages. In *Proc. of CogSci*, 2003.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. RTFM: Generalising to new environment dynamics via reading. In *Proc. of ICLR*, 2020.

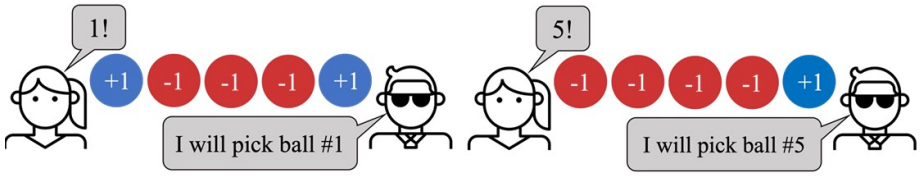

Figure 7: Overview of SELECTSAY game, reproduced from Hu & Sadigh (2023).

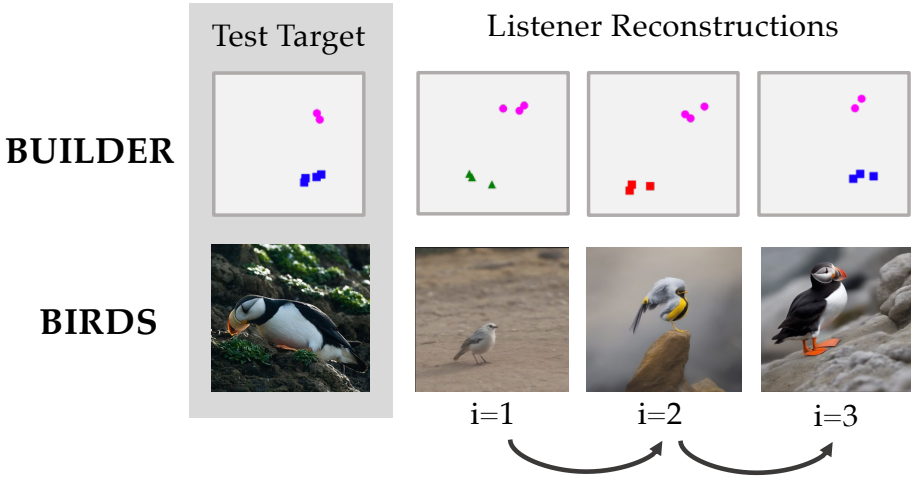

Figure 8: BUILDER and BIRDS tasks consist of speaker and listener agents. At test time, the speaker provides a language description to the listener that helps them recreate the image accurately. For both tasks, PLLB helps improve listener accuracy over time.

A Appendix

A.1 Additional Figures

A.2 Hyperparameters

1. SELECTSAY: We use the same default parameters used in InstructRL (Hu & Sadigh, 2023), including setting the regularization strength $\lambda = 0.25$. Additionally, each time we invoke `gen_rule`, we create an ensemble of 3 rules, which we aggregate over when construct the probability distribution over actions during `update`.

2. MAZE: We use the InstructRL objective with a tabular Q-learning agent, but introduce a ϵ_{LM} parameter that controls whether the regularization strength λ is 0 or 1 at each time-step in an episode. Although we did not observe a strong effect on modifying ϵ_{LM}, we did not explore values larger than $\epsilon_{LM} = 0.4$ as that led to increased inference cost and experiment latency. Finally, each time we invoke `gen_rule` we create an ensemble of 4 rules, which we aggregate over when construct the probability distribution over actions during `update`.

3. BUILDER: For the listener agent we finetune BART for a maximum of 100 epochs at each iteration of PLLB, employing early stopping on a held-out validation set and using the default arguments provided by the HuggingFace Transformers library (Lewis et al., 2019). Each time we invoke `gen_rule`, we same 3 rules and select the rule with the highest aggregate likelihood across all tokens. Likewise, for each image description (the speaker's action), we

sample 3 possible descriptions under the given rule and select the one with the highest probability.

4. BIRDS: For the fine-tuned version of the listener agent, we finetune StableDiffusion for 1000 steps on a separate finetuning dataset. When generating images, we simply sampe one image per description, using 10 inference steps and a guidance scale of 7.5.

A.3 Compute Resources

We use the Together and Replicate API services to conduct all inference and finetuning jobs across all experiments, except for the listener model in BUILDER, which was implemented using the HuggingFace Transformers library implementation of BART (Lewis et al., 2019). All tasks were run using 1 NVIDIA A40 GPU.

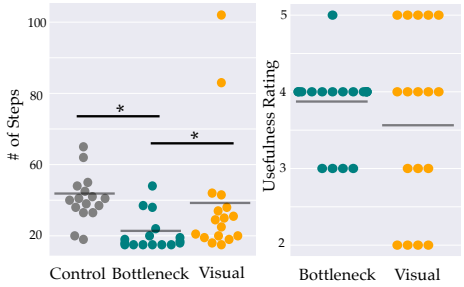

Figure 6: Participants provided PLLB rules solve new mazes faster than those given either visual or no aid, and reported linguistic more useful than visual.

A.4 Maze

See Figures 9 and 10.

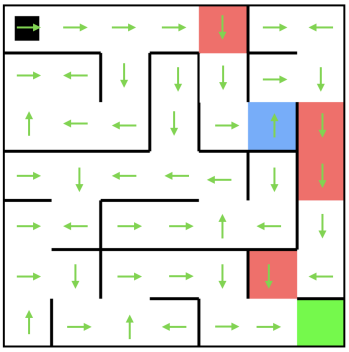

Figure 9: Visual aid provided to participants in the human subject study for the MAZE task.

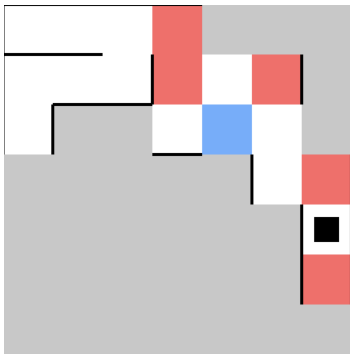

Figure 10: Interface used in the human subject study for the MAZE task. Cells not visited by the participants are hidden in gray.

A.5 Dataset Details for Birds.

We consider images of the following bird species in the CUB-200-2011 dataset from Wah et al. (2011)

for the BIRDS task: `[Indigo Bunting, Cardinal, Yellow Breasted Chat, American Crow, Vermillion Flycatcher, California Gull, Blue Jay, Tropical Kingbird, White Pelican, Horned Puffin]`.

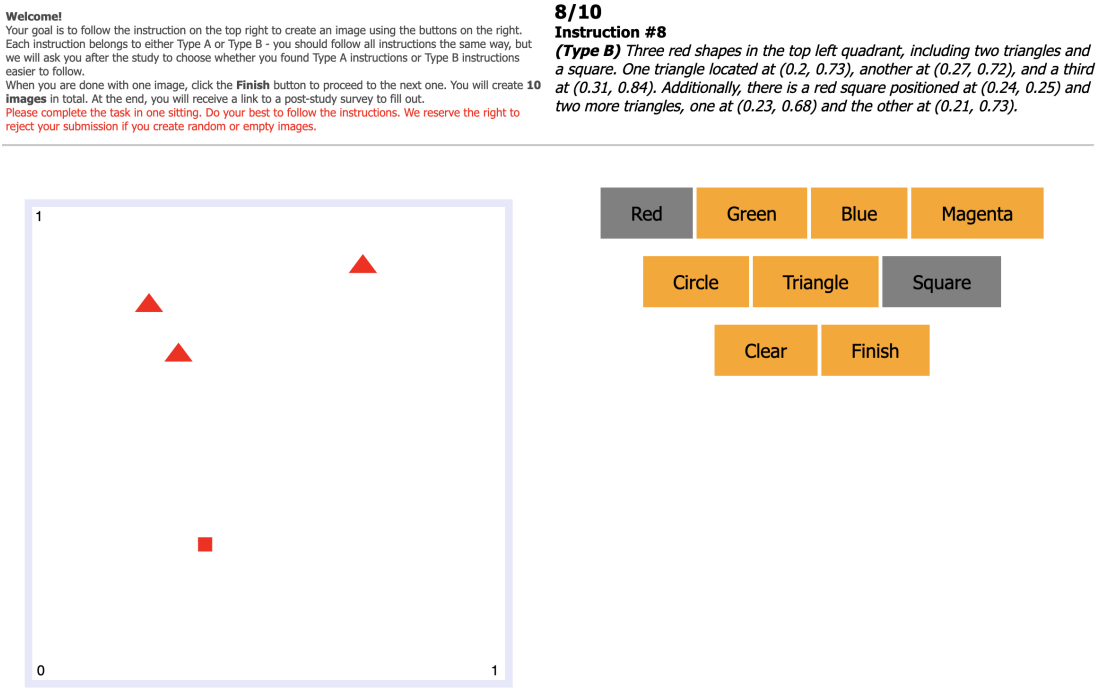

Figure 11: Interface used in the human subject study for the BUILDER task.

A.6 Full Prompts

We first provide the full prompts used for `gen_rule`. For space constraints, we do not include example `samples`, but note they follow the format shown in Figure 1.

SelectSay: You will be given a list of (OBSERVATION, ACTION, REWARD) examples collected from two agents learning to solve a task. Possible ACTIONS an agent can take are: 1, 2, 3, 4, 5, and quit. Each OBSERVATION describes the ordered sequence of actions that AGENT 1 picks, and each ACTION describes the ACTION that AGENT 2 picks based on the given OBSERVATION. The examples are separated into HIGH REWARD and LOW REWARD examples.+`[samples]`+Output a language rule that best summarizes the strategy AGENT 2 should follow to receive HIGH REWARD, not LOW REWARD, based on the examples. Start the instruction with the prefix 'I should'.

Maze: You will be given a list of example (OBSERVATION, ACTION) trajectories collected from an AGENT learning to solve a maze. Each trajectory receives a REWARD. Possibles OBSERVATIONS an agent see are: WHITE, RED, BLUE Possible ACTIONS an agent can take are: NORTH, SOUTH, EAST, WEST. The examples are separated into HIGH REWARD and LOW REWARD examples + `[samples]` + Output a language rule that best summarizes the strategy the AGENT should follow when picking a sequence of ACTIONS to solve the maze and receive HIGH REWARD, not LOW REWARD, based on the examples. Start the instruction with the prefix 'I should'.

Builder: There are two agents. The goal of Agent 1 is to provide instructions to Agent 2 that helps Agent 2 to successfully recreate the image. You will be given a list of (ORIGINAL, AGENT 1 INSTRUCTION, REWARD) values where ORIGINAL is the original description of an image,

INSTRUCTION is the instruction provided by Agent 1 to Agent 2, and REWARD is the reward Agent 2 receives when trying to re-create the image (higher is better). The examples are separated into HIGH REWARD and LOW REWARD examples. + `[samples]`+

Based on the examples above, output a list of 2 RULES for Agent 1 to follow when generating INSTRUCTION in order to receive HIGH REWARD, instead of LOW REWARD. Write the rules after the prefix RULES:

Birds: The top row of three images have the following HIGH REWARD descriptions:+`high reward samples`+The bottom row of three images have the following LOW REWARD descriptions:+`low reward samples`+Provide a rule I should follow in order to provide image descriptions with HIGH REWARD, not LOW REWARD. Provide the rule after the prefix RULE:"

We next provide the full prompts used in `update` for each task.

1. SelectSay: [$\mathcal{L}$]+Agent 1 selected `[observation]`. So I should select
2. Maze: You are an agent solving a maze following a provided RULE. You will be given a list of PREVIOUS ACTIONS and the CURRENT OBSERVATION. Follow the RULE to select your NEXT ACTION (East, West, South, North):

   RULE: + [$\mathcal{L}$]+ PREVIOUS ACTIONS: + [$\tau_{1...t-1}$] + CURRENT OBSERVATION: + `[observation]` +

   What is the NEXT ACTION you should take? Output one of (East, West, South, North) after the prefix NEXT ACTION:.
3. Builder: You will be given a DESCRIPTION of an image. Your goal is to use this description to provide a short INSTRUCTION to help someone else, who cannot see the image, accurately reconstruct it. You will also be given a list of RULES you must follow when providing the instruction.

   DESCRIPTION: + `observation` +

   RULES: +$\mathcal{L}$ +

   Please provide a short instruction following the prefix INSTRUCTION:
4. Birds: Provide a one-sentence description of this image, using the following RULES:+$\mathcal{L}$

A.7 Examples of Generated Rules for All Environments

See Tables 1, 2, 3, and 4.

A.8 Analysis of Generated Rules for All Environments

In our experiments, we did not observe a significantly strong quantitative difference in performance when **`gen_rule`** is instantiated with models of different sizes (e.g. llama-2-70b-chat vs. llama-2-13b-chat). However, we did notice interesting qualitative differences across samples that are likely due to the additional fine-tuning step using reinforcmenet learning from human feedback (RLHF, see https://llama.meta.com/llama2/ for more information). We briefly describe these differences per environment below.

1. SelectSay: Because of the simplicity of this environment, it is possible for a rule $\mathcal{L}$ to summarize the full optimal human-interpretable policy as a sequence of if-statements (e.g. *"If the current state is '1', I should take action 1."*) - we observed that smaller language models (e.g. llama-2-7b-chat and llama-2-13b-chat) always did this, while larger models (e.g. llama-2-70b-chat and mistral-8x-7b-instruct) were better able to generalize and use more efficient language, such as *"I should take the same action as the observation"*. We also noticed smaller models often included superfluous language, such as *"I should always take the action that leads to the highest reward"*.

Setting	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_5$
Standard	I should follow the strategy of choosing action 4 whenever the observation is 2, 3, 4, or 5.	I should follow the strategy of choosing actions based on the observation provided by AGENT 1. If the observation is 1, I should choose action 1. If the observation is 2, I should choose action 3. If the observation is 3, I should choose action 3. If the observation is 4, I should choose action 4. If the observation is 5, I should choose action 5.	I should follow the strategy of choosing the same action as AGENT 1 for observations 1, 2, 3, 4, and 5.
Fixed Speaker	I should choose action 1 when observation is 1 or 2 or 4 or 5. I should choose action 2 when observation is 3.	I should choose action 1 when observation is 1, 2, 3 or 5. I should choose action 2 when observation is 4.	I should choose action 1 when observation is 1. I should choose action 4 when observation is 2. I should choose action 5 when observation is 3. I should choose action 2 when observation is 4. I should choose action 3 when observation is 5.

Table 1: Example $\mathcal{L}$ rules generated for the SelectSay environment, for the Standard setting (both Listener and Speaker agents are RL agents trained from random initialization) and a Fixed Speaker agent.

Size	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_5$	$\mathcal{L}_8$
7x7	I should start by moving north, then continue moving north until I reach a wall, then turn east and move until I reach a wall, then turn south and move until I reach a wall, then turn west and move until I reach the end of the maze.	I should prioritize moving North, East, and South over moving West, as the HIGH REWARD EXAMPLES show a higher frequency of these actions. 2. I should avoid moving West unnecessarily, as it is rarely seen in the HIGH REWARD EXAMPLES and often leads to LOW REWARD EXAMPLES	I should take the EAST action when I see WHITE, and then repeat the EAST action until I see RED, then take the SOUTH action, and then repeat the EAST action until I see BLUE, then take the NORTH action, then take the EAST action until I see RED again, then take the SOUTH action, and finally take the WEST action.	I should take the EAST action when I observe WHITE, and keep taking EAST actions until I observe RED, then take the SOUTH action. After taking the SOUTH action, I should take the EAST action again until I observe BLUE, then take the NORTH action. After taking the NORTH action, I should take the EAST action until I observe RED again, then take the SOUTH action.

Table 2: Example $\mathcal{L}$ rules generated for the Maze environment for two different maze sizes.

2. Maze: As discussed in Section 6, the majority of generated rules captured the underlying color semantics of the maze, enabling generalization. However, we did observe that smaller model sizes (e.g. llama-2-7b-chat) resulted in more superfluous language (e.g. *"I should always prioritize taking actions that lead to the most recent reward, and avoid taking actions that lead to low reward."*) and the generated rules focused more on the first actions the agent should take, which may not always generalize.

Setting	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$
Re-Initialization	1. Be specific with location details: Agent 1 should provide detailed location information for each element in the image, such as x and y coordinates. 2. Use descriptive language for elements, such as "red dot" or "green triangle".	1. Use specific coordinates when instructing Agent 2 to draw shapes. 2. Use descriptive language to specify the color and shape of each element. For example, "a green triangle" instead of "a green thing".	1. Be specific with location coordinates: provide specific coordinates for the location of each shape, using the format x=0.XX, y=0.YY. 2. Use descriptive shape names: Instead of using generic terms like "dot" or "square," use more descriptive names that indicate the shape's color and size, such as "green triangle" or "red square."
Continual Training	1. Be specific and detailed in your instructions. High reward examples have specific coordinates and shapes, while low reward examples have more general descriptions. 2. Use a consistent format for your instructions. High reward examples have a consistent format for listing coordinates and shapes, while low reward examples have a more free-form format.	1. Provide explicit coordinates for each element in the image, using the format (x, y). 2. Use specific colors when referring to elements in the image, such as "red", "green", or "blue". Avoid using vague terms like "colored" or "shaded".	1. Use a consistent format for describing shapes, such as always listing the x-coordinate first, followed by the y-coordinate. For example, instead of "one green square at the point x=0.53, y=0.24", use "one green square at (0.53, 0.24)". 2. Avoid using vague terms like "various shades of green". Instead, use specific colors, such as "green" or "blue". Additionally, use specific shapes, such as "square" or "triangle", rather than vague terms like "rectangle".

Table 3: Example $\mathcal{L}$ rules generated for the BUILDER environment.

Reward	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$
color	Describe the bird's color, species, and any distinctive markings or patterns.	Describe the bird's coloration accurately.	Describe the bird's coloration accurately.
background	Include details about the bird's surroundings, such as the type of branch or post it is on, and any additional elements in the background.	Include the bird's action (perched, flying, standing) and its location (on a branch, railing, pole, etc.)	Describe the bird's action (flying, perching, standing) and the environment it is in (sky, tree, water).
species	Describe the bird's color, markings, and any distinctive features.	Describe the subject's unique features, such as coloration, beak shape, or other distinguishing characteristics.	Include specific details about the bird's appearance, such as the color of its feathers, beak, or eyes, and any distinctive markings or patterns.

Table 4: Example $\mathcal{L}$ rules generated for the BIRDS environment demonstrate reward-specificity over time.

3. BUILDER: While we do not observe any model-specific features, there do exist variation across samples in the type of formatting and syntax generated rules encourage (e.g. provide coordinates *" using the format (x, y)"* vs. *" using the format "x=0.XX, y=0.YY""*),

leading to agents converging to different descriptions. Furthermore, some rules encourage list formats in image descriptions (e.g. *1.Draw a green dot at (0.72, 0.21). 2. Draw a green dot at (0.73, 0.72).*) while other rules encouraged clustering of identical shapes (e.g. *Draw two green dots at (0.72, 0.21) and (0.73, 0.72).*)

4. BIRDS: We observed that rules demonstrated more reward-specificity (i.e. specific to background, color, or species rewards) when generated with larger VLMs (e.g. `llava-13b`) than smaller models (e.g. `llava-v1.6-vicuna-7b`), with the latter primarily proposing rules that encouraged more details descriptions (e.g. *"Avoid using vague or general terms"*).