Improving Intrinsic Exploration with Language Abstractions

Jesse Mu[1*] **Victor Zhong**[2] **Roberta Raileanu**[3] **Minqi Jiang**[3,4]
Noah Goodman[1] **Tim Rocktäschel**[3,4] **Edward Grefenstette**[3,4]
[1]Stanford University [2]University of Washington [3]Meta AI [4]University College London

Abstract

Reinforcement learning (RL) agents are particularly hard to train when rewards are sparse. One common solution is to use *intrinsic* rewards to encourage agents to explore their environment. However, recent intrinsic exploration methods often use state-based novelty measures which reward low-level exploration and may not scale to domains requiring more abstract skills. Instead, we explore *natural language* as a general medium for highlighting relevant abstractions in an environment. Unlike previous work, we evaluate whether language can improve over existing exploration methods by directly extending (and comparing to) competitive intrinsic exploration baselines: AMIGo (Campero et al., 2021) and NovelD (Zhang et al., 2021). These language-based variants outperform their non-linguistic forms by 45–85% across 13 challenging tasks from the MiniGrid and MiniHack environment suites.

1 Introduction

A central challenge in reinforcement learning (RL) is designing agents which can solve complex, long-horizon tasks with sparse rewards. In the absence of extrinsic rewards, one popular solution is to provide an agent with *intrinsic* rewards for exploration and motivation (Schmidhuber, 1991, 2010; Oudeyer et al., 2007; Oudeyer and Kaplan, 2009).

Intrinsic exploration methods invariably face the challenging design choice of how to measure exploration. One common answer is that an agent should be rewarded for attaining "novel" states in the environment, but naïve measures of novelty have limitations. For example, consider an agent in the kitchen of a large house that must make an omelet. Simple novelty-based exploration will reward an agent for visiting every room in the house, but a more effective strategy would be to stay put and use the stove. Moreover, like kitchens with different-colored appliances, states can look cosmetically different but have the same underlying semantics, and thus are not truly novel. Together, these constitute two fundamental challenges for intrinsic exploration, especially in real-world or procedurally-generated

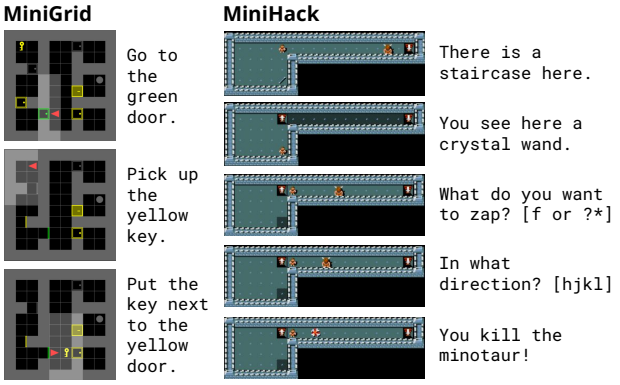


Figure 1: **Natural language conveys meaningful environment abstractions.** Language state annotations in the MiniGrid KeyCorridorS4R3 (Chevalier-Boisvert et al., 2018) and MiniHack Wand of Death (Hard) (Samvelyan et al., 2021) tasks. Language is a natural way to specify the relevant abstractions in an environment, which we use to improve intrinsic exploration methods.

tasks: first, how can we reward true progress in the environment over meaningless exploration? Second, how can we tell when a state is not just superficially, but *semantically* novel?

Fortunately, humans are equipped with a powerful tool for solving both problems: natural language. As a cornerstone of human learning and teaching, language has strong priors over the features and behaviors needed for meaningful interaction and skill acquisition. It also describes a rich and compositional set of concepts as simple as directions (e.g. *move left*) and as abstract as conjunctions of high level tasks (e.g. *acquire the amulet and defeat the wizard*) that can categorize and unify many possible states in the world. Our aim is to see whether language abstractions can improve upon existing state-based methods for exploration in RL. Language-guided exploration methods have been proposed in the past (Goyal et al., 2019; Colas et al., 2020b; Mirchandani et al., 2021), though they are often evaluated in isolated environments, without comparison to other exploration baselines. Instead, we build our investigation directly off of two recent, state-of-the-art intrinsic exploration methods: AMIGo (Campero et al., 2021), where a teacher proposes intermediate location-based goals for a student, and NovelD (Zhang et al., 2021), which rewards an agent for visiting novel regions of the state space. Upon these methods, we propose **L-AMIGo**, where the teacher proposes goals expressed via natural language instead of coordinates, and **L-NovelD**, a variant of NovelD with an additional exploration bonus for visiting linguistically-novel states. This allows us to conduct a con-

*Correspondence to muj@stanford.edu. Work done during an internship at Meta AI.

trolled evaluation of the effect of language on competitive approaches to exploration.

Across 13 challenging, procedurally-generated, sparse-reward tasks in the MiniGrid (Chevalier-Boisvert et al., 2018) and MiniHack (Samvelyan et al., 2021) environment suites, we show that language-parameterized exploration methods outperform their non-linguistic counterparts by 45–85%, especially in more abstract tasks with larger state and action spaces. We also show that language improves the interpretability of the training process, either by developing a natural curriculum of semantic goals (in L-AMIGo) or by allowing us to visualize the most novel states during training (in L-NovelD). Finally, we study when and where agents are able to use the fine-grained compositional semantics of language to learn and explore more efficiently.

2 Related Work

Exploration in RL. Exploration has a long history in RL, from ϵ-greedy (Sutton and Barto, 1998) or count-based exploration (Strehl and Littman, 2008; Bellemare et al., 2016; Ostrovski et al., 2017; Martin et al., 2017; Tang et al., 2017; Machado et al., 2020) to intrinsic motivation (Oudeyer et al., 2007; Oudeyer and Kaplan, 2008, 2009) and curiosity-based learning (Schmidhuber, 1991). More recently, deep neural networks have been used to measure novelty with changes in state representations (Burda et al., 2018; Raileanu and Rocktäschel, 2020; Zhang et al., 2021) or prediction errors in world models (Stadie et al., 2015; Achiam and Sastry, 2017; Pathak et al., 2017). Another long tradition generates curricula of intrinsic goals to encourage learning (Forestier et al., 2017; Florensa et al., 2017; Fang et al., 2019; Campero et al., 2021; Racaniere et al., 2020; Portelas et al., 2020a,b; Colas et al., 2020b,c). In this paper, we explore the potential benefit of language on these approaches to exploration.

Language for Exploration. Our observation that language can improve exploration is not new: language has been used to shape policies (Harrison et al., 2018) and rewards (Bahdanau et al., 2018; Goyal et al., 2019, 2020; Mirchandani et al., 2021), set intrinsic goals (Colas et al., 2020a,b), and even represent states (Schwartz et al., 2019). Accordingly, the two methods we propose here share some similarities with this work, but are also notably different.

L-AMIGo, our variant of AMIGo with language goals, is similar to the IMAGINE agent of Colas et al. (2020b) which sets and completes language goals. However, IMAGINE requires a perfectly compositional space of language goals, which the agent tries to explore so that it can complete novel goals at test time. Instead, we make no assumptions on the language and explore to complete some auxiliary task, using an alternative *goal difficulty* metric to measure progress.

Meanwhile, L-NovelD is inspired by language-guided reward shaping methods (Bahdanau et al., 2018; Harrison et al., 2018; Goyal et al., 2019; Mirchandani et al., 2021). Much of this work, however, requires access to language task descriptions (Bahdanau et al., 2018; Mirchandani et al., 2021). LEARN (Goyal et al., 2019) and Harrison et al. (2018) learn models from offline state-language annotations to reward language-related actions, but in our setting, we receive language annotations *online*, in tasks with noisier language where message-based rewards alone underperform (as shown by our L-NovelD ablations in Appendix D.2).

More generally, our philosophy is not to claim superiority over other language-guided exploration methods; rather, our distinguishing contribution is a direct comparison between state-of-the-art exploration methods and their linguistic variants. The work above has shown that language can improve exploration, but mostly compared to extrinsic rewards alone (Goyal et al., 2019; Colas et al., 2020b) or other language baselines (Mirchandani et al., 2021). Instead, we explore *when linguistic rewards can improve upon alternative intrinsic rewards*, using the same exploration methods and challenging benchmarks typical of recent RL research.

Other uses of language in RL. The intersection of language and RL has seen significant interest in recent years (Luketina et al., 2019), with models that execute language instructions (Branavan et al., 2009; Misra et al., 2017; Anderson et al., 2018; Fried et al., 2018; Chen et al., 2019; Shridhar et al., 2020), play text-based games (Côté et al., 2018; Urbanek et al., 2019; Ammanabrolu and Riedl, 2019), read external knowledge sources (Branavan et al., 2012; Zhong et al., 2020; Hanjie et al., 2021), or use language for hierarchical planning (Andreas et al., 2018; Jiang et al., 2019; Hu et al., 2019; Chen et al., 2021; Prakash et al., 2021). We emphasize that we do not study (hierarchical) instruction following, but instead examine how language can improve exploration in more general non-linguistic settings.

3 Problem Statement

We explore RL within the setting of an augmented Markov Decision Process (MDP) defined as the tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma, L)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the environment transition dynamics, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the *extrinsic* reward function, where $r_t = R(s_t, a_t)$ is the reward obtained at time t by taking action a_t in state s_t, and γ is the discount factor. To add language, we assume access to an *annotator* L that produces **natural language descriptions** for states: $\ell_t = L(s_t)$, such as those in Figure 1. Note that not every state needs a description (which we model with a null description $\emptyset$) and the set of descriptions need not be known ahead of time.[1]

We ultimately seek a policy that maximizes the expected discounted (extrinsic) reward $R_t = \mathbb{E}\left[\sum_{k=0}^{H} \gamma^k r_{t+k}\right]$, where H is the finite time horizon. During training, however, we maximize an *augmented* reward $r_t^+ = r_t + \lambda r_t^i$, where r_t^i is an *intrinsic* reward and λ is a scaling hyperparameter.

[1]For presentational simplicity, the annotator here outputs a single description per state, but in practice, we allow an annotator to produce *multiple* descriptions: e.g. in MiniGrid, *open the door* and *open the red door* describe the same state. This requires two minor changes in the equations, described in Footnotes 2 and 3.

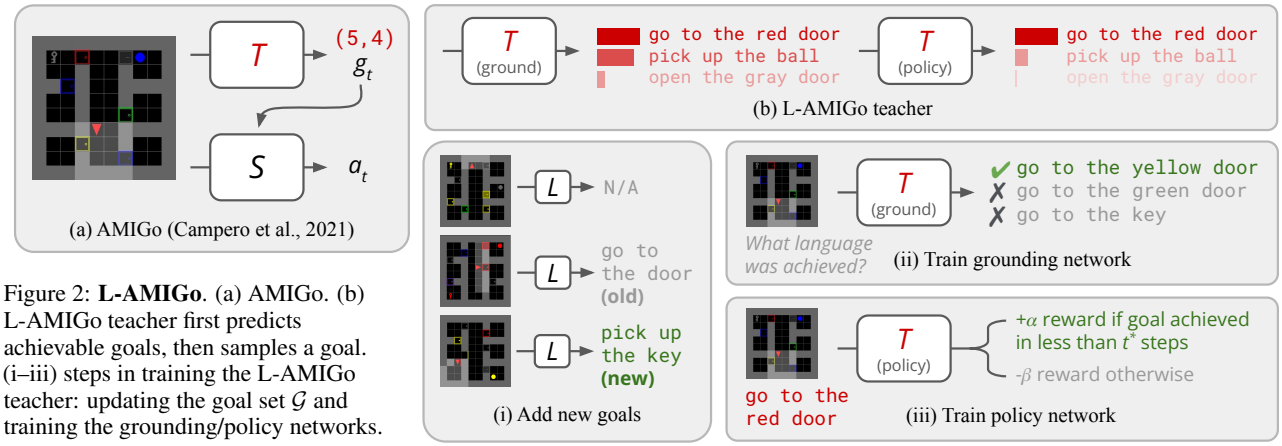


Figure 2: **L-AMIGo**. (a) AMIGo. (b) L-AMIGo teacher first predicts achievable goals, then samples a goal. (i–iii) steps in training the L-AMIGo teacher: updating the goal set $\mathcal{G}$ and training the grounding/policy networks.

Like past work (Jiang et al., 2019; Waytowich et al., 2019; Mirchandani et al., 2021) we make the simplifying assumption of access to oracle language state annotations. Many modern RL environments and games come pre-equipped with language commentary, including NetHack/MiniHack (Küttler et al., 2020; Samvelyan et al., 2021) and text-based games (Côté et al., 2018; Urbanek et al., 2019; Shridhar et al., 2021), though our method should scale to annotator models learned offline (Bahdanau et al., 2018; Goyal et al., 2019; Mirchandani et al., 2021), given a sufficiently large dataset of states and language annotations. Since this idea has been well-proven, we assume oracle access to L, though a similar approach should work in our setting.

In the following sections, we describe our models. We stress that our aim is *not* to exhaustively evaluate every method in the literature to claim that ours are "best". Instead, we ask if language can improve the exploration process by extending two methods representative of state-of-the-art approaches to exploration (as of writing): AMIGo and NovelD.

4 L-AMIGo

We now describe our approach to jointly training a student and a goal-proposing teacher, extending AMIGo (Campero et al., 2021) to arbitrary natural language goals.

4.1 Adversarially Motivated Intrinsic Goals (AMIGo)

AMIGo (Campero et al., 2021) augments an RL student policy with goals generated by a teacher, which provide intrinsic reward when completed (Figure 2a). The teacher should propose intermediate goals that start simple, but grow harder to encourage an agent to explore its environment.

Student. Formally, the student S is a *goal-conditioned* policy parameterized as $\pi_S(a_t \mid s_t, g_t; \theta_S)$, where g_t is the goal provided by the teacher, and the student receives an intrinsic reward r_t^i reward of 1 if the teacher's goal at that timestep is completed. The student receives a goal from the teacher either at the beginning of an episode, or mid-episode, if the existing intrinsic goal has been completed.

Teacher. Separately, AMIGo trains a *teacher* policy $\pi_T(g_t \mid s_t; \theta_T)$ to propose goals to the student given its initial state. The teacher is trained with a reward r_t^T that depends on a *difficulty threshold* t^*: the teacher is given a positive reward of $+\alpha$ for proposing goals that take the student more than t^* timesteps to complete, and $-\beta$ for goals that are completed sooner, or never completed within the finite time horizon. To encourage proposing harder and harder goals that promote exploration, t^* is increased linearly throughout training: whenever the student completes 10 goals in a row under the current difficulty threshold, it is increased by 1, up to some tunable maximum difficulty.

This teacher is updated separately from the student at different time intervals. Formally, its training data is batches of (s_0, g_t, r_t^T) tuples collected from student trajectories for nonzero r_t^T, where s_0 is the initial state of the student's trajectory and g_t is the goal that led to reward r_t^T.

The original paper (Campero et al., 2021) implements AMIGo for MiniGrid only, where the goals g_t are (x, y) coordinates that must be reached by the student. The student gets the teacher's goal embedded directly as a feature of the $M \times N$ environment, and the teacher is a dimensionality-preserving convolutional neural network which encodes the student's $M \times N$ environment into an $M \times N$ distribution over coordinates, from which a single goal is selected.

4.2 Extension to L-AMIGo

Student. The L-AMIGo student is a policy conditioned not on (x, y) goals, but on *natural language goals* ℓ_t: $\pi_S(a_t \mid s_t, \ell_t; \theta_S)$. Given the "goal" ℓ_t, the student is now rewarded if it reaches a state with the natural language description ℓ_t, i.e. if $\ell_t = L(s_t)$. Typically this student will encode the goal with a learned language model and concatenate the goal representation with its state representation.

Teacher. Now the L-AMIGo teacher selects goals from the set of possible language descriptions in the environment. Because the possible goals are initially unknown, the teacher maintains a running set of goals $\mathcal{G}$ that is updated as the student encounters new state descriptions (Figure 2i).

This move to language creates a challenge: not only must a teacher choose which goal to give to the student, but it must also determine which goals are achievable in the first place. For example, the goal *go to the red door*

only makes sense in environments that contain red doors. In L-AMIGo, these tasks are factorized into a **policy network**, which produces the distribution over goals given a student's state, and a **grounding network**, which predicts the probability that a goal is likely to be achieved in the first place (Figure 2b):

$$\pi_T(\ell_t \mid s_t; \theta_T) \propto p_{\text{ground}}(\ell_t \mid s_t; \theta_T) \cdot p_{\text{policy}}(\ell_t \mid s_t; \theta_T) \tag{1}$$

$$p_{\text{ground}}(\ell_t \mid s_t; \theta_T) = \sigma\left(f(\ell; \theta_T) \cdot h_{\text{ground}}(s; \theta_T)\right) \tag{2}$$

$$p_{\text{policy}}(\ell_t \mid s_t; \theta_T) \propto f(\ell_t; \theta_T) \cdot h_{\text{policy}}(s_t; \theta_T) \tag{3}$$

Equation 3 describes the policy network as producing a probability for a goal by computing the dot product between goal and state representations $f(\ell_t; \theta_T)$ and $h_{\text{policy}}(s_t; \theta_T)$, normalizing over possible goals; this policy is learned identically to the standard AMIGo teacher (Figure 2iii). Equation 2 specifies the grounding network as predicting whether a goal is *achievable* in an environment, by applying the sigmoid function to the dot product between the goal representation $f(\ell_t; \theta_T)$ and a (possibly separate) state representation $h_{\text{ground}}(s_t; \theta_T)$. Given an oracle grounding classifier, which outputs only 0 or 1, this is equivalent to restricting the teacher to proposing only goals that are achievable in a given environment. In practice, however, we learn the classifier online (Figure 2ii). Given the initial state s_0 of an episode, we ask the grounding network to predict the first language description encountered along this trajectory: $\ell_{\text{first}} = L(s_{t'})$, where t' is the minimum t where $L(s_t) \neq \emptyset$. This is formalized as a multilabel binary cross entropy loss,

$$\mathcal{L}_{\text{ground}}(s_0, \ell_{\text{first}}) = -\big(\log(p_{\text{ground}}(\ell_{\text{first}} \mid s_0; \theta_T)) + \frac{1}{|\mathcal{G}|-1} \sum_{\ell' \in \mathcal{G} \setminus \{\ell_{\text{first}}\}} \log(1 - p_{\text{ground}}(\ell' \mid s_0; \theta_T))\big), \tag{4}$$

where the second term can be seen as noisily generating negative samples of (start state, unachieved description) pairs based on the set of descriptions $\mathcal{G}$ known to the teacher at the time.[2] Note that since $\mathcal{G}$ is updated during training, Equation 4 grows to include more terms over time.

To summarize, training the teacher involves three steps: (1) updating the running set of descriptions seen in the environment, (2) learning the policy network based on whether the student achieved goals proposed by the teacher, and (3) learning the grounding network by predicting descriptions encountered from initial states. Algorithm S1 in Appendix A describes how L-AMIGo trains in an asynchronous actor-critic framework, where the student and teacher are jointly trained from batches of experience collected from separate actor threads, as used in our experiments (see Section 6).

5 L-NovelD

Next, we describe NovelD (Zhang et al., 2021), which extends simpler tabular- (Strehl and Littman, 2008) or pseudo- (Bellemare et al., 2016; Burda et al., 2018) count-based intrinsic exploration methods, and our language variant, L-NovelD. Instead of simply rewarding an agent for rare states, NovelD rewards agents for *transitioning* from states with low novelty to states with higher novelty. Zhang et al. (2021) show that NovelD surpasses Random Network Distillation (Burda et al., 2018), another popular exploration method, on a variety of tasks including MiniGrid and Atari.

5.1 NovelD

NovelD defines the intrinsic reward r_t^i to be the difference in novelty between state s_t and the previous state s_{t-1}:

$$r_t^i = \text{NovelD}_s(s_t, s_{t-1}) \triangleq \underbrace{\max(N(s_{t-1}) - \alpha N(s_t), 0)}_{\text{Term 1 (NovelD)}} \cdot \underbrace{\mathbb{1}(N_e(s_t) = 1)}_{\text{Term 2 (ERIR)}}. \tag{5}$$

In the first NovelD term, $N(s_t)$ is the novelty of state s_t; this quantity describes the difference in novelty between successive states, which is clipped > 0 so the agent is not penalized from moving back to less novel states. α is a hyperparameter that scales the average magnitude of the reward. The second term is the **E**pisodic **R**eduction on **I**ntrinsic **R**eward (ERIR): a constraint that the agent only receives reward when encountering a state *for the first time in an episode*. $N_e(s_t)$ is an episodic state counter that tracks state visitation counts, as defined by (x, y) coordinates.

Measuring novelty with RND. In MDPs with smaller state spaces, it is sufficient to track exact state visitation counts, in which case the novelty is typically the inverse square root of visitation counts (Strehl and Littman, 2008). However, in richer environments like ours where states are rarely ever revisited, NovelD proposes to use the popular Random Network Distillation (RND) (Burda et al., 2018) technique as an approximate novelty measure. Specifically, the novelty of a state is measured by the prediction error of a state embedding network that is trained jointly with the agent to match the output of a fixed, random target network. The intuition is that states which the RND network has been trained on will have lower prediction error than novel states.

5.2 Extension to L-NovelD

Our incorporation of language is simple: we add an additional exploration bonus based on novelty defined according to the natural language descriptions of states:

$$\text{NovelD}_\ell(\ell_t, \ell_{t-1}) \triangleq \max(N(\ell_{t-1}) - \alpha N(\ell_t), 0) \cdot \mathbb{1}(N_e(\ell_t) = 1). \tag{6}$$

This bonus is identical to standard NovelD: $N(\ell)$ is the novelty of the description ℓ as measured by a *separately parameterized* RND network encoding the description (producing a zero embedding if there is no description),[3] and $N_e(\ell_t)$ is an episodic counter that is 1 when the language

[2] If the student encounters multiple "first" descriptions in a state, the teacher predicts 1 for *each* description, and 0 for all others.

[3] For multiple messages, we average NovelD of each message.

description has been encountered for the first time this episode.

We keep the original NovelD exploration bonus, as language rewards may be sparse and a basic navigation bonus can encourage the agent to reach language-annotated states. Therefore, the intrinsic reward for L-NovelD is

$$r_t^i = \text{NovelD}_s(s_t, s_{t-1}) + \lambda_\ell \text{NovelD}_\ell(\ell_t, \ell_{t-1}) \quad (7)$$

where λ_ℓ controls the trade-off between Equations 5 and 6.

One might ask why we do not simply include the language description ℓ as input into the RND network, along with the state. While this can work in some cases, decoupling the state and language novelties allow us to precisely control the trade-off between the two, with a hyperparameter that can be tuned to different tasks. In contrast, a combined input obfuscates the relative contributions of state and language to the overall novelty. Appendix D.2 shows ablation experiments where we show that (1) combining the state and language inputs or (2) using the language novelty term alone leads to less consistent performance across tasks.

6 Experiments

To facilitate apples-to-apples comparison between linguistic and non-linguistic exploration, we evaluate L-AMIGo against AMIGo, and L-NovelD against NovelD, as written in the TorchBeast (Küttler et al., 2019) implementation of IMPALA (Espeholt et al., 2018), a common asynchronous actor-critic framework. We run each model 5 times across 13 tasks within two challenging procedurally-generated RL environments, MiniGrid (Chevalier-Boisvert et al., 2018) and MiniHack (Samvelyan et al., 2021). We adapt baseline models provided for both environments (Campero et al., 2021; Samvelyan et al., 2021); for full architectural, training, and hyperparameter details, see Appendix B.

6.1 Environments

MiniGrid. Following Campero et al. (2021), we evaluate on the most challenging tasks in MiniGrid (Chevalier-Boisvert et al., 2018), which involve navigation and manipulation tasks in symbolic gridworlds: **KeyCorridorS{3,4,5}R3** (Figure 1) and **ObstructedMaze_{1Dl,2Dlhb,1Q}**. These tasks involve picking up a ball in a locked room, with the key to the door hidden in boxes or other rooms and the door possibly obstructed by objects. The suffix indicates the size of the environment, in increasing order. See Appendix E for more details.

To add language, we use the language annotations from the BabyAI platform (Chevalier-Boisvert et al., 2019). The set of possible descriptions includes 651 instructions in the BabyAI language, which involve *goto*, *open*, *pickup*, and *putnext* commands which can be applied to a variety of objects qualified by type (e.g. *box*, *door*) and optionally color (e.g. *red*, *blue*). Appendix E contains the full list of messages. The oracle language annotator emits a message when the corresponding action is completed.

MiniHack. MiniHack (Samvelyan et al., 2021) is a suite of procedurally-generated tasks of varying difficulty set in the roguelike game NetHack (Küttler et al., 2020). MiniHack contains a diverse action space beyond simple MiniGrid-esque navigation, including planning, inventory management, tool use, and combat. These actions cannot be expressed by (x, y) positions, but instead are captured by in-game messages (Figure 1). We evaluate our methods on a representative suite of tasks of varying difficulty: **River**, **Wand of Death (WoD)-{Medium,Hard}**, **Quest-{Easy,Medium}**, and **MultiRoom-{N2,N4}-Extreme**.

For space reasons, we describe the WoD-Hard environment here, but defer full descriptions of tasks (and messages) to Appendix E. In WoD-Hard, depicted in Figure 1, the agent must learn to use a *Wand of Death*, which can zap and kill enemies. This involves a complex sequence of actions: the agent must find the wand, pick it up, choose to *zap* an item, select the wand in the inventory, and finally choose the direction to zap (towards the minotaur which is pursuing the player). It must then proceed past the minotaur to the goal to receive reward. Taking these actions out of order (e.g. trying to *zap* something with nothing in the inventory, or selecting something other than the wand) has no effect.

It is difficult to enumerate all MiniHack messages, as they are hidden in low-level game code and there are many rare edge cases. To gain a rough sense of the messages in each task, we can look at expert policies. Agents which have solved Wand of Death tasks encounter approximately 60 messages. Only 5–10 of these messages must be encountered to solve the task, including inventory actions (*f - a metal wand [acquired]*, *what do you want to zap?*, *in what direction?* and combat messages (*You kill the minotaur!*, *Welcome to level 2.*). Most messages are irrelevant to the task (e.g. picking up and throwing stones) or nonsensical (*There is nothing to pick up*, *That is a silly thing to zap*).

7 Results

Figure 3 shows training curves with AMIGo, NovelD, language variants, and the extrinsic reward-only IMPALA baseline. Following Agarwal et al. (2021), we summarize these results with the interquartile mean (IQM) of all methods in Figure 4, with bootstrapped 95% confidence intervals constructed from 5k samples per model/env combination.[4][5] We come to the following conclusions:

Linguistic exploration outperforms non-linguistic exploration. Both algorithms, L-AMIGo and L-NovelD, outperform their nonlinguistic counterparts. Despite variance in runs and across environments, averaged across all environments (**Overall**) we see a statistically significant improvement of L-AMIGo over AMIGo (.26 absolute, 45% relative improvement) and of L-NovelD over NovelD (.35 absolute, 85% relative improvement). In some cases, the training curves in Figure 3 show that L-AMIGo and L-NovelD reach the same asymptotic performance as their non-linguistic versions, but with fewer samples or more stability (e.g. KeyCorridorS3R3 L-AMIGo, Quest-Easy

[4] Appendix C also plots probabilities of improvement and area under the curve (AUC) for each model, with similar conclusions.

[5] See Appendix D for ablation studies of L-AMIGo's grounding network and the components of L-NovelD.

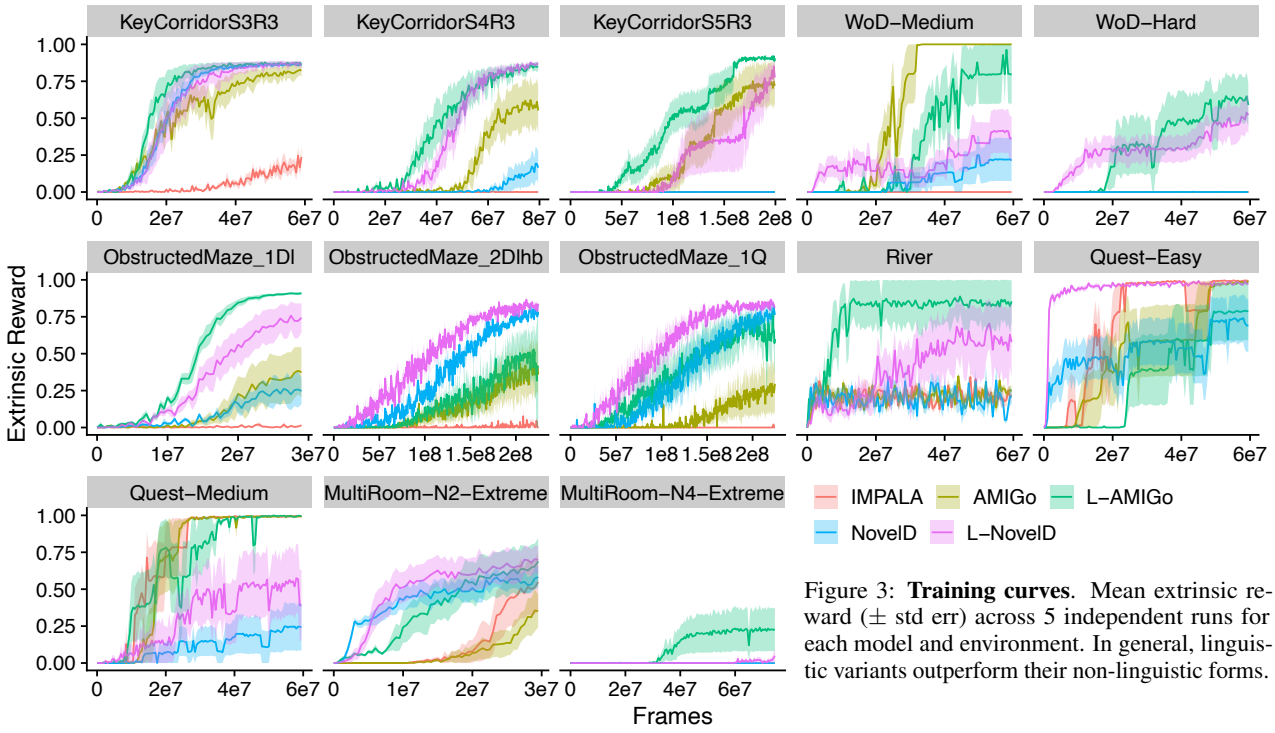


Figure 3: **Training curves**. Mean extrinsic reward ($\pm$ std err) across 5 independent runs for each model and environment. In general, linguistic variants outperform their non-linguistic forms.

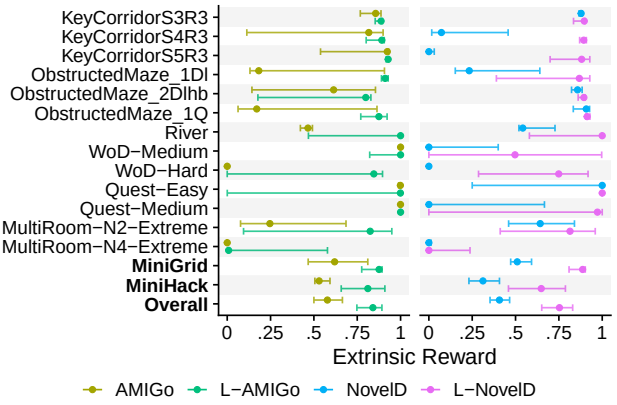


Figure 4: **Aggregate performance**. Interquartile mean (IQM) of models across tasks. **Overall** plots the average IQM across all tasks. Dot is median; error bars are 95% bootstrapped CIs.

L-NovelD). The AUC plots in Appendix C.2 show these differences more clearly.

Linguistic exploration excels in larger environments. Our tasks include sequences of environments with the same underlying dynamics, but larger state spaces and thus more challenging exploration problems. In general, larger environments result in bigger improvements of linguistic exploration over non-linguistic exploration. For example, there is no difference in asymptotic performance for language/non-language variants on KeyCorridorS3R3, yet the gaps grow as the environment size grows to KeyCorridorS5R3, especially in L-NovelD. A similar trend can be seen in the WoD tasks, where AMIGo actually outperforms L-AMIGo in WoD-Medium, but in WoD-Hard is unable to learn at all.

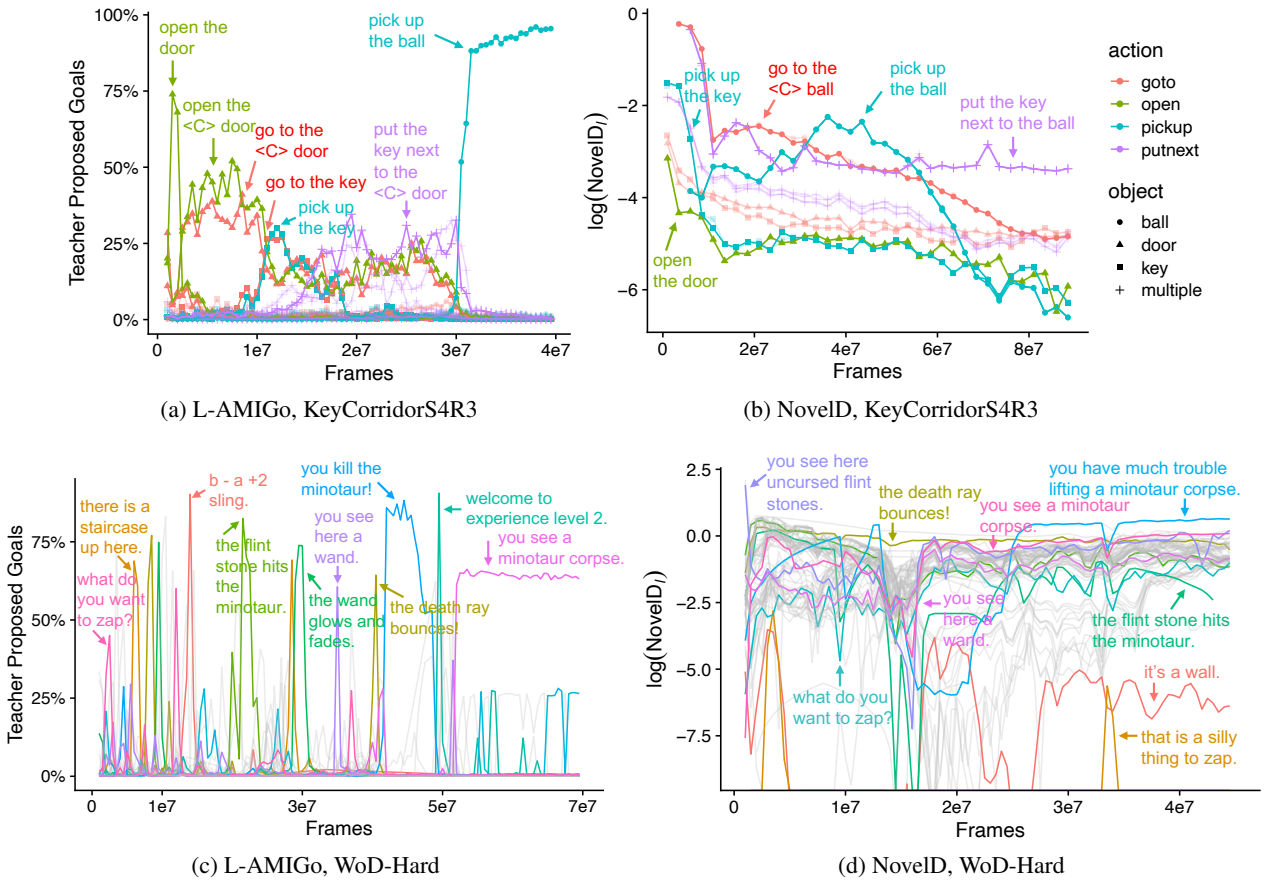
7.1 Interpretability

One auxiliary benefit of our language-based methods is that the language states and goals can provide insight into an agents' training and exploration process, as in Figure 5.

Emergent L-AMIGo Curricula. Campero et al. (2021) showed that AMIGo teachers produce an interpretable curriculum, with initially easy (x, y) goals located next to the student's start location, and later harder goals referencing distant locations behind doors. In L-AMIGo, we can see a similar curriculum emerge through the proportion of *language goals* proposed by the teacher throughout training. In the KeyCorridorS4R3 environment (Figure 5a), the teacher first proposes the generic goal *open (any) door* before then proposing goals involving specific colored doors (where <C> is a color). Next, the agent discovers keys, and the teacher proposes *pick[ing] up the key* and putting it in certain locations. Finally, the teacher and student converge on the extrinsic goal *pick up the ball*.

Due to the complexity of the WoD-Hard environment, the curriculum for the teacher is more exploratory (Figure 5c). The teacher proposes unrelated goals at first, such as finding staircases and slings on the ground. At one point, the teacher proposes throwing stones at the minotaur (an ineffective strategy) before devoting more time towards wand actions (*you see here a wand*, *the wand glows and fades*). Eventually, as the student grows more competent, the teacher begins proposing goals that involve directly killing the minotaur (*you kill the minotaur*, *welcome to experience level 2*) before converging on the message *you see a minotaur corpse*—the final message needed to complete the episode.

L-NovelD Message Novelty. Similarly, L-NovelD allows for interpretation by examining the messages with highest intrinsic reward as training progresses. In KeyCorridorS4R3 (Figure 5b), overall novelty decreases over time, but easy goals such as *open the door* decrease fastest, while the true extrinsic goal (*pick up the ball*) and even rarer actions (*put the key next to the ball*) remain high throughout training. In WoD-Hard (Figure 5d), messages vary widely in novelty: simple and nonsensical messages like *that is a*



(a) L-AMIGo, KeyCorridorS4R3

(b) NovelD, KeyCorridorS4R3

(c) L-AMIGo, WoD-Hard

(d) NovelD, WoD-Hard

Figure 5: **Interpretation of language-guided exploration.** For the KeyCorridorS4R3 and WoD-Hard environments, shown are curricula of goals proposed by the L-AMIGo teacher (a,c) and the intrinsic reward of messages (some examples labeled) for L-NovelD (b,d).

silly thing to zap and *it's a wall* quickly plummet, while more novel messages are rarer states that require killing the minotaur (*you see/you have trouble lifting a minotaur corpse*).

7.2 Does semantics matter?

Language not only denotes meaningful features in the world; its lexical and compositional semantics also explains how actions and states relate to each other, which could be used by our methods. For example, in L-AMIGo, an agent might more easily *go to the red door* if it already knows how to *go to the yellow door*. Similarly, in L-NovelD, training the RND network on the message *go to the yellow door* could also lower novelty of similar messages like *go to the red door*, which might encourage an agent to cover a semantically broader language space. To check this hypothesis, we run "one-hot" variants of L-AMIGo and L-NovelD where the semantics of the language goals/states are hidden to the models: each unique goal is replaced with a one-hot identifier (e.g. *go to the red door* $\rightarrow$ 1, *go to the blue door* $\rightarrow$ 2, etc) but otherwise functions identically to the original goal.

Figure 6 shows results comparing L-AMIGo and L-NovelD to their one-hot variants. We make two observations. **(1)** One-hot goals actually perform quite competitively, demonstrating that the primary benefit of language in these tasks is to coarsen the state space, rather than provide fine-grained semantic similarity relations between states. **(2)** Nevertheless, L-AMIGo is better able to exploit semantics, with a more significant improvement in aggregate performance over one-hot goals, in contrast to L-NovelD, which shows less of a difference. We leave for future work a more in-depth exploration of what kinds of environments might stand to gain more from semantics in language-guided exploration.

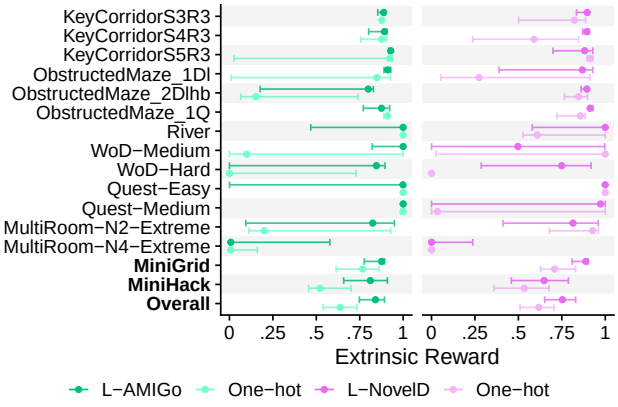


Figure 6: **IQM performance of L-AMIGo and L-NovelD with one-hot goals.** Plot elements same as Figure 4.

8 Limitations and Discussion

We have presented variants of two popular, state-of-the-art intrinsic exploration methods, L-AMIGo and L-NovelD, which parameterize novelty through language abstractions of states. These variants outperform their non-linguistic

counterparts on 13 tasks spread across two challenging procedurally-generated RL domains, MiniGrid and MiniHack, with 45–85% relative improvement over non-linguistic baselines across environments. They also allow for interpretation of the training process by visualizing the language goals and states encountered during training.

Despite their success as presented here, our models have some limitations. First, as is common in work like ours, it will be important to alleviate the restriction on oracle language annotations, perhaps by using learned state description models. Furthermore, L-AMIGo specifically cannot handle tasks such as the full NetHack game which have unbounded language spaces and many redundant goals (e.g. *go to/approach/arrive at the door*), since it selects a single goal which must be achieved verbatim. An exciting extension to L-AMIGo would propose abstract goals (e.g. *kill [any] monster* or *find a new item*), possibly in a continuous semantic space, that can be satisfied by multiple messages.

There are also more general avenues of work: for example, in better understanding when and why language semantics benefits exploration (Section 7.2), and using pretrained models to imbue semantics into the models beforehand. Additionally, one benefit of language is that it is biased towards relevant environment features, but as shown in Section 6.1, there are many superfluous messages in MiniGrid and MiniHack. Our models are encouragingly still able to use the language to explore, but one could imagine degenerate settings where the language is completely unrelated to the extrinsic task and therefore useless. Follow-up work should measure how robust these methods are to the "cleanliness" of the linguistic signal. Nevertheless, we believe the success of L-AMIGo and L-NovelD demonstrates the power of even relatively noisy language in these domains, and underscores the importance of abstract and semantically-meaningful measures of novelty for intrinsic exploration.

Acknowledgements

We thank Heinrich Küttler, Mikael Henaff, Andy Shih, and Alex Tamkin for helpful comments and feedback, and Mikayel Samvelyan for help with MiniHack. JM is supported by an Open Philanthropy AI Fellowship.

References

Joshua Achiam and Shankar Sastry. 2017. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. 2021. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*.

Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3557–3565.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Jacob Andreas, Dan Klein, and Sergey Levine. 2018. Learning with latent language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2166–2179.

Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Arian Hosseini, Pushmeet Kohli, and Edward Grefenstette. 2018. Learning to understand goal specifications by modelling reward. In *International Conference on Learning Representations*.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*.

SRK Branavan, Harr Chen, Luke S Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90.

SRK Branavan, David Silver, and Regina Barzilay. 2012. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2018. Exploration by random network distillation. In *International Conference on Learning Representations*.

Andres Campero, Roberta Raileanu, Heinrich Kuttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. 2021. Learning with AMIGo: Adversarially motivated intrinsic goals. In *International Conference on Learning Representations*.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12538–12547.

Valerie Chen, Abhinav Gupta, and Kenneth Marino. 2021. Ask your humans: Using human instructions to improve generalization in reinforcement learning. In *International Conference on Learning Representations*.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. BabyAI: A platform to study the sample efficiency of grounded language learning. In *International Conference on Learning Representations*.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. 2018. Minimalistic gridworld environment for OpenAI gym. https://github.com/maximecb/gym-minigrid.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *International Conference on Learning Representations*.

Cédric Colas, Ahmed Akakzia, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. 2020a. Language-conditioned goal generation: a new approach to language grounding for RL. *arXiv preprint arXiv:2006.07043*.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Dominey, and Pierre-Yves Oudeyer. 2020b. Language as a cognitive tool to imagine goals in curiosity driven exploration. In *Advances in Neural Information Processing Systems*.

Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. 2020c. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: A short survey. *arXiv preprint arXiv:2012.09830*.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. TextWorld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. 2018. IMPALA: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1407–1416.

Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. 2019. Curriculum-guided hindsight experience replay. In *Advances in Neural Information Processing Systems*.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. 2017. Reverse curriculum generation for reinforcement learning. In *Proceedings of the 1st Conference on Robot Learning*, pages 482–495.

Sébastien Forestier, Rémy Portelas, Yoan Mollard, and Pierre-Yves Oudeyer. 2017. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pages 3318–3329.

Prasoon Goyal, Scott Niekum, and Raymond J Mooney. 2019. Using natural language for reward shaping in reinforcement learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19*.

Prasoon Goyal, Scott Niekum, and Raymond J Mooney. 2020. Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards. In *Proceedings of the 4th Conference on Robot Learning*, pages 485–497.

Austin W Hanjie, Victor Zhong, and Karthik Narasimhan. 2021. Grounding language to entities and dynamics for generalization in reinforcement learning. In *Proceedings of the Thirty-Eighth International Conference on Machine Learning*.

Brent Harrison, Upol Ehsan, and Mark O Riedl. 2018. Guiding reinforcement learning exploration using natural language. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1956–1958.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hengyuan Hu, Denis Yarats, Qucheng Gong, Yuandong Tian, and Mike Lewis. 2019. Hierarchical decision making by generating and following natural language instructions. In *Advances in Neural Information Processing Systems*.

Yiding Jiang, Shixiang Gu, Kevin Murphy, and Chelsea Finn. 2019. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Heinrich Küttler, Nantas Nardelli, Thibaut Lavril, Marco Selvatici, Viswanath Sivakumar, Tim Rocktäschel, and Edward Grefenstette. 2019. TorchBeast: A PyTorch platform for distributed RL. *arXiv preprint arXiv:1910.03552*.

Heinrich Küttler, Nantas Nardelli, Alexander H Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. 2020. The NetHack learning environment. In *Advances in Neural Information Processing Systems*.

J Luketina, N Nardelli, G Farquhar, J Foerster, J Andreas, E Grefenstette, S Whiteson, and T Rocktäschel. 2019. A survey of reinforcement learning informed by natural language. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 6309–6317.

Marlos C Machado, Marc G Bellemare, and Michael Bowling. 2020. Count-based exploration with the successor representation. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 5125–5133.

Jarryd Martin, Suraj Narayanan Sasikumar, Tom Everitt, and Marcus Hutter. 2017. Count-based exploration in feature space for reinforcement learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 2471–2478.

Suvir Mirchandani, Siddharth Karamcheti, and Dorsa Sadigh. 2021. ELLA: Exploration through learned language abstraction. In *Advances in Neural Information Processing Systems*.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015.

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. 2017. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2721–2730.

Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. 2007. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286.

Pierre-Yves Oudeyer and Frederic Kaplan. 2008. How can we define intrinsic motivation? In *Proceedings of the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*.

Pierre-Yves Oudeyer and Frederic Kaplan. 2009. What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 1:6.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2778–2787.

Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. 2020a. Teacher algorithms for curriculum learning of deep RL in continuously parameterized environments. In *Proceedings of the 4th Conference on Robot Learning*, pages 835–853.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. 2020b. Automatic curriculum learning for deep RL: A short survey. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20) Survey Track*, pages 4819–4825.

Bharat Prakash, Nicholas Waytowich, Tim Oates, and Tinoosh Mohsenin. 2021. Interactive hierarchical guidance using language. *arXiv preprint arXiv:2110.04649*.

Sebastien Racaniere, Andrew K Lampinen, Adam Santoro, David P Reichert, Vlad Firoiu, and Timothy P Lillicrap. 2020. Automated curricula through setter-solver interactions. In *International Conference on Learning Representations*.

Roberta Raileanu and Tim Rocktäschel. 2020. RIDE: Rewarding impact-driven exploration for procedurally-generated environments. In *International Conference on Learning Representations*.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Kuttler, Edward Grefenstette, and Tim Rocktäschel. 2021. MiniHack the planet: A sandbox for open-ended reinforcement learning research. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*.

Jürgen Schmidhuber. 1991. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 222–227.

Jürgen Schmidhuber. 2010. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.

Erez Schwartz, Guy Tennenholtz, Chen Tessler, and Shie Mannor. 2019. Language is power: Representing states using natural language in reinforcement learning. *arXiv preprint arXiv:1910.02789*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10740–10749.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. 2015. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

Alexander L Strehl and Michael L Littman. 2008. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT Press.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2017. #exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 673–683.

Nicholas Waytowich, Sean L Barton, Vernon Lawhern, and Garrett Warnell. 2019. A narration-based reward shaping approach using grounded natural language commands. *arXiv preprint arXiv:1911.00497*.

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. 2021. NovelD: A simple yet effective exploration criterion. In *Advances in Neural Information Processing Systems*.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. 2020. RTFM: Generalising to new environment dynamics via reading. In *International Conference on Learning Representations*.

A L-AMIGo algorithm

Algorithm S1 Asynchronous learning step for L-AMIGo

Input: student batch size B_S, teacher policy batch size B_{policy}, grounding network batch size B_{ground}
$\mathcal{B}_{\text{policy}} \leftarrow \emptyset, \mathcal{B}_{\text{ground}} \leftarrow \emptyset$ ▷ Init teacher batches
$\mathcal{G} \leftarrow \emptyset$ ▷ Track descriptions seen thus far
while not converged **do**
  Sample batch $\mathcal{B}$ of size B_S from actors
  Train student on $\mathcal{B}$
  Add new descriptions ℓ_t in the batch to $\mathcal{G}$
  Update $\mathcal{B}_{\text{policy}}$ with (s_0, ℓ_t, r_t^T) tuples where goal ℓ_t completed ($r_t^T = +\alpha$) or episode ended ($r_t^T = -\beta$)
  if $|\mathcal{B}_{\text{policy}}| > B_{\text{policy}}$ **then**
    Train teacher policy on $\mathcal{B}_{\text{policy}}$; $\mathcal{B}_{\text{policy}} \leftarrow \emptyset$
  end if
  Update $\mathcal{B}_{\text{ground}}$ with $(s_0, \ell_{\text{first}})$ tuples
  if $|\mathcal{B}_{\text{ground}}| > B_{\text{ground}}$ **then**
    Train grounding net on $\mathcal{B}_{\text{ground}}$; $\mathcal{B}_{\text{ground}} \leftarrow \emptyset$
  end if
end while

Algorithm S1 describes the joint student/teacher learning step of L-AMIGo. Batches of experience $\mathcal{B}$ are generated from actors, which are used to update the student policy, and the teacher policy at different intervals defined by batch sizes B_{policy} and B_{ground}. The set of goals $\mathcal{G}$ known to the teacher is also progressively updated.

To reiterate the teacher training process: the teacher policy network is trained on tuples of student initial state s_0, proposed goal ℓ_t, and teacher rewards r_t^T (which is $+\alpha$ if the goal was completed by the student in $\geq t^*$ steps, or $-\beta$ if the goal was completed in $< t^*$ steps or never completed before episode termination). The teacher grounding network is trained on tuples of initial state s_0 and the first language description encountered along that trajectory ℓ_{first}. Given s_0, the grounding network is asked to predict 1 for ℓ_{first} and 0 for all other goals known to the teacher at the time.

B Architecture and training details

Here, we describe the architecture, training details, and hyperparameters for the MiniGrid and MiniHack tasks.

B.1 MiniGrid

All models are adapted from Campero et al. (2021). Figure S1 from Campero et al. (2021) details the architecture of the standard AMIGo student and teacher.

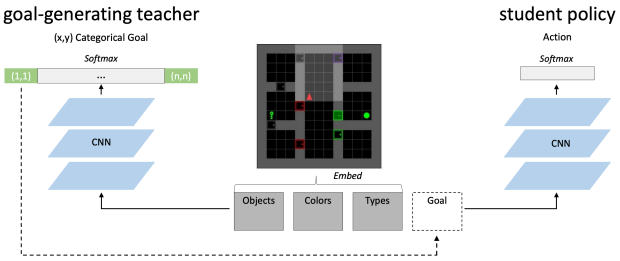


Figure S1: **Original AMIGo model overview.** Original figure from Campero et al. (2021), reproduced with permission.

AMIGo student. The student gets an $7 \times 7 \times 3$ partial and egocentric grid representation, where each cell in the 7×7 grid is represented by 3 features indicating the type of the object, color of object, and object state (e.g. for doors, open/closed). The student first embeds the features into type/color/state embeddings of size 5, 3, and 2 respectively, as well as the (x, y) goal as a singleton feature, to get a $7 \times 7 \times 11$ grid which is then fed through a 5-layer convolutional neural network interleaved with Exponential Linear Units (ELUs; Clevert et al., 2016). Each layer has 32 output channels, filter size of $(3, 3)$, 2 stride, and 1 padding. The output of the ConvNet is then flattened and fed through a fully-connected layer to produce a 256-d state embedding. The policy and value functions are linear layers on top of this state embedding. The policy in particular produces a distribution over 7 possible actions (forward, pick up, put down, open, left, right, and a "done" action).

AMIGo teacher. The teacher gets an $N \times N \times 3$ fully observed grid encoding of the environment, where N varies according to the environment size. Like the student, the teacher embeds the grid representation into embeddings which are then fed through a 4-layer dimensionality-preserving neural network again interwoven with ELUs. Each convolutional layer has 16 output channels, filter size of $(3, 3)$, 1 stride, and 1 padding, except the last layer which has only 1 output channel. The ConvNet processes the input embeddings into a spatial action distribution over grid locations, from which an goal is sampled. The value head takes as input the penultimate layer from the ConvNet (i.e. a grid of $N \times N \times 16$) to produce the value estimate.

L-AMIGo Student. The L-AMIGo student has the same ConvNet base as the standard AMIGo student, but without an (x, y) goal channel feature (so the input to the ConvNet is $7 \times 7 \times 10$ instead of 11). To encode the goal, the student uses a one-layer Gated Recurrent Unit (GRU; Cho et al. (2014)) recurrent neural network (RNN) with embedding size 64 and embedding size 256. The last hidden state of the GRU is taken as the goal representation, which is then concatenated with the state embedding to be fed into the policy and value functions, respectively.

L-AMIGo Teacher. The L-AMIGo *policy network* has the same ConvNet as the standard AMIGo teacher, but without the last layer (so the output is a $N \times N \times 16$ grid). The last output of the ConvNet is averaged across the spatial map to form a final 16-dimensional state embedding. The L-AMIGo teacher also has a GRU of same dimensionality as the L-AMIGo student. To propose a goal, the teacher embeds each known goal in the vocabulary with the GRU to produce 256-dimensional goal embeddings which are then projected via a linear layer into the 16-dimensional goal embedding space. The dot product of the goal embeddings and the state form the logits of the goal distribution. The value head takes as input the $N \times N \times 16$ unaveraged state representation concatenated with the logits of the distribution of language goals.

The *grounding network* also uses the 16-dimensional state embedding and the same GRU as the L-AMIGo policy network, but the 256-d goal embeddings are projected via a

separate linear layer to a separate set of 16-dimensional embeddings. The dot product between these grounder-specific goal embeddings and the state embedding represent the log probabilities of the goal being achievable in an environment.

NovelD and L-NovelD. For NovelD experiments we use the student policy of L-AMIGo but with the goal embedding always set to the 0 vector.

The NovelD RND network uses the same convolutional network as the AMIGo/L-AMIGo students, taking in an egocentric agent view. The L-NovelD message embedding network uses a GRU parameterized identically to those of the L-AMIGo student and teacher. For NovelD experiments we set $\alpha = 1$, scale the RND loss by 0.1, and use the same learning rate as the main experiments. Using a grid search, we found optimal scaling factors of the standard NovelD reward to be 0.5 and the L-NovelD reward (i.e. λ_ℓ) also to be 0.5.

Hyperparameters. We use the same hyperparameters as in the original AMIGo paper: a starting difficulty threshold t^* of 7, a maximum difficulty t^* of 100, positive reward for the teacher $+\alpha = 0.7$, negative reward $-\beta = -0.3$, learning rate 10^{-4} which is linearly annealed to 0 throughout training, batch size 32, teacher policy batch size 32, teacher grounder batch size 100, unroll length 100, RMSprop optimizer with $\varepsilon = 0.01$ and momentum 0, entropy cost 0.0005, generator entropy cost 0.05, value loss cost 0.5, intrinsic reward coefficient $\lambda = 1$.

B.2 MiniHack

Models are adapted from baselines established for the NetHack (Küttler et al., 2020) and MiniHack (Samvelyan et al., 2021) Learning Environments.

AMIGo student. The student is a recurrent LSTM-based (Hochreiter and Schmidhuber, 1997) policy. The NetHack observation contains a 21×79 matrix of glyph identifiers, a 21-dimensional feature vector of agent stats (e.g. position, health, etc), and a 256-character (optional) message. The student produces 4 dense representations from this observation which are then concatenated to form the state representation.

First, an embedding of the entire game area is created. Each glyph is converted into a 64-dimensional vector embedding, i.e. the input is now a $21 \times 79 \times 64$ grid. This entire grid is fed through a 5-layer ConvNet interleaved with ELUs. Each conv layer has a filter size of $(3, 3)$, stride 1, padding 1, and 16 output channels, except for the last which has 8 output channels. **Second**, an embedding of a 9×9 egocentric crop of the grid around the agent is created. This is created by feeding the crop through another separately-parameterized ConvNet of the same architecture. **Third**, the 21-d feature vector of agent stats is fed into a 2-layer MLP (2 layers with 64-d outputs with ReLUs in between) to produce a 64-d representation of the agent stats. **Fourth**, the message is parsed with the BERT (Devlin et al., 2019) WordPiece tokenizer and fed into a GRU with embedding size 64 and hidden size 256. These four embeddings are then concatenated and form the state representation, which is then fed into the LSTM which contextualizes the current representation. This final representation is fed into linear policy and value heads to produce action distribution and value estimates.

AMIGo teacher. The teacher first embeds the agent stats using the same MLP that the student uses. It has the same 5-layer ConvNet used by the student to embed the full game area, except the input channels are modified to accept the 64-d feature vector concatenated to every cell in the 21×79 grid, so the input to the ConvNet is $21 \times 79 \times 128$. Additionally, the output layer has only 1 output channel. This produces a spatial action distribution over the 21×79 grid from which a "goal" as (x, y) coordinate is sampled.

L-AMIGo student. The student works identically to the standard AMIGo student, except without the teacher goal channel concatenated into the grid input. Also, in addition to the 4 representation constructed from the observation, the student gets 2 more representations constructed from the teacher's language goal. **First**, the student uses the same GRU used to process the game message to encode the teacher's language goal to create a 256-d representation. **Second**, the difference between the observed language embedding and the teacher language embedding is used as an additional feature (when this is the 0 vector then the student has reached the goal). All together, this forms 6 representations that together constitute the state representation.

L-AMIGo teacher. The teacher constructs the same state representation as the standard AMIGo student (*not* the AMIGo teacher). The teacher then embeds all known goals with the a word-level GRU with the same architecture as the message network used by the student. These 256-d message embeddings are then projected to the state representation hidden size, and the dot product between message embeddings and state representations forms the distribution over goals. Similarly, for the grounding network, the 256-d message embeddings are projected via a separate linear layer to produce probabilities of achievability. Like in MiniGrid, the state representation and the goal logits are used as input for the value head.

NovelD and L-NovelD. As in MiniGrid experiments, for NovelD experiments we use the L-AMIGo student policy where the goal embedding is always 0.

The NovelD RND network uses the cropped ConvNet representation of the student, fed through a linear layer to produce a 256-d state representation. Egocentric crops change more over time and are thus a more reliable signal of "novelty" than the full grid representation (verified with experiments). Additionally, as done in Burda et al. (2018), two more additional layers are added to the final MLP of the predictor network only (not the random target network). The L-NovelD message RND network uses the same word-level GRU architecture of the student. We set $\alpha = 0.5$, scale the RND loss by 0.1, and use the same learning rate as the main student policy. The standard NovelD reward is left alone, and the L-NovelD reward hyperparameter (λ_ℓ) is grid-searched and set to 50 for all MiniHack experiments except Quest-{Easy,Medium}, where it is 30.

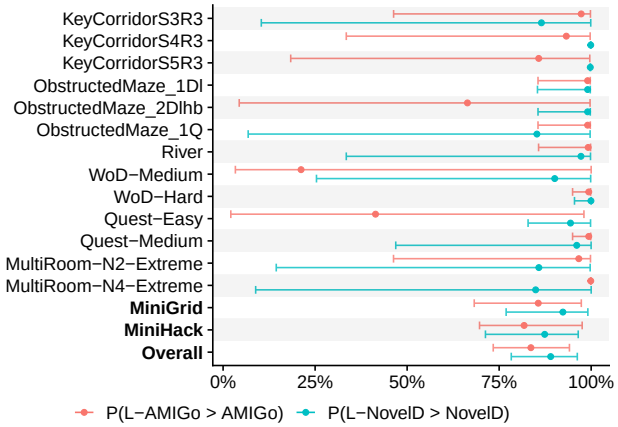


Figure S2: Probability of improvement of L-AMIGo over AMIGo, and L-NovelD over NovelD, as measured by Mann-Whitney U tests between their final performances. Plot elements same as Figure 4.

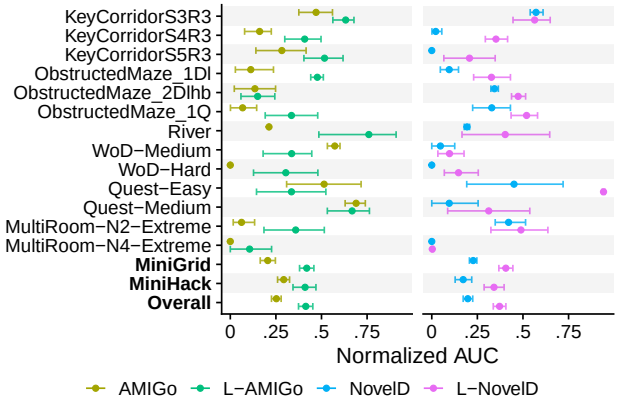


Figure S3: Normalized area under the curve (AUC) for AMIGo, L-AMIGo, NovelD, and L-NovelD. Plot elements same as Figure 4.

Hyperparameters. We generally stick to the same hyperparameters of Samvelyan et al. (2021). We use a starting difficulty threshold t^* of 1, a maximum difficulty t^* of 2 (a high goal difficulty is not as important for MiniHack), positive reward for the teacher $+\alpha = 0.7$, negative reward $-\beta = -0.3$, linearly-annealed learning rate 10^{-4}, batch size 32, teacher policy batch size 32, teacher grounder batch size 500, unroll length 100, RMSprop optimizer with $\varepsilon = 0.01$ and momentum 0, entropy cost 0.0005, generator entropy cost 0.05, value loss cost 0.5, intrinsic reward coefficient $\lambda = 0.4$. ε-greedy exploration was used for the teacher policy with $\varepsilon = 0.05$, which we found helped learning.

B.3 Compute Details

Each model was run for 5 independent seeds on a machine with 40 CPUs, 1 Tesla V100 GPU, and 64GB RAM. Runs take between 4 hours (for ObstructedMaze_1Dl) to 20 hours (for the longest MultiRoom-N4-Extreme and KeyCorridorS5R3 tasks).

C Additional visualizations of main results

C.1 Probability of improvement.

In addition to the IQM, another alternative interpretation of results as advocated in Agarwal et al. (2021) is to use the *probability of improvement* of algorithm A over algorithm B, as measured by the nonparametric Mann-Whitney U test between independent runs from both algorithms. Figure S2 shows results from such an evaluation, again evaluated over bootstrapped confidence intervals constructed from 5000 samples per model/env combination. Qualitative results are the same as in Figure 4: overall, across environments, L-AMIGo and L-NovelD are both highly likely to outperform AMIGo and NovelD.

C.2 Area Under the Curve (AUC).

The IQM (Figure 4) and probability plots (Figure S2) use point estimates of ultimate performance attained after a fixed compute budget, which does not measure differences in sample efficiency between runs. As a measure which also elucidates differences in sample efficiency, we propose to use the normalized Area Under the Curve (AUC) to compare runs. Results are in Figure S3, and are qualitatively similar to the IQM and probability plots, but here, differences in sample efficiency can be seen in some environments, e.g. in KeyCorridorS3R3 for L-AMIGo, which are absent from the IQM plot.

D Ablations

D.1 L-AMIGo grounding network

Figure S4 shows performance of L-AMIGo without the grounding network, where the policy network directly produces a distribution over goals without first predicting goal achievability, aggregated across domains. Overall, L-AMIGo without the grounding network performs reasonably well, matching full L-AMIGo performance on MiniHack. On MiniGrid, we see a modest difference in aggregate performance, though importantly, we also see greatly increased training stability with the grounding network on individual environments. Specifically, the confidence intervals are much larger for L-AMIGo without the grounding network on **KeyCorridorS{3,4,5}R3** and **ObstructedMaze_{1Dl,1Q}**.

We hypothesize that the difference between MiniGrid and MiniHack tasks is because MiniGrid goals differ more between episodes: for example, since the colors of the doors are randomly shuffled, not all environments have red doors, so it is helpful to explicitly predict such features of the environment with a grounding network. In contrast, MiniHack envs have a more consistent set of goals (e.g. every WoD seed has a wand, a minotaur, etc), and so the grounding network is less necessary in this case.

D.2 L-NovelD components

To examine the relative performance of each component of L-NovelD, we run ablation experiments by (1) using the language-based reward *only*, or (2) combining the language and state embedding into a single representation. Note that using the language-based reward only is quite similar to approaches like LEARN (Goyal et al., 2019) and Harrison et al. (2018) that give language-based intermediate rewards, though those methods do not implement a notion of novelty that reduces the reward as an agent encounters the same

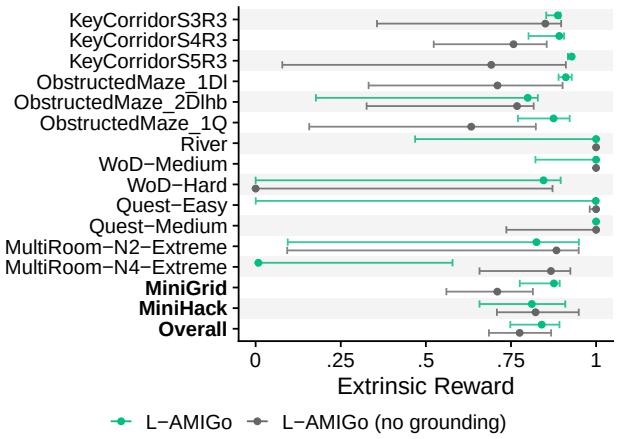


Figure S4: **L-AMIGo grounding network ablation**. IQM of L-AMIGo with and without grounding network across environments. Plot elements same as Figure 4.

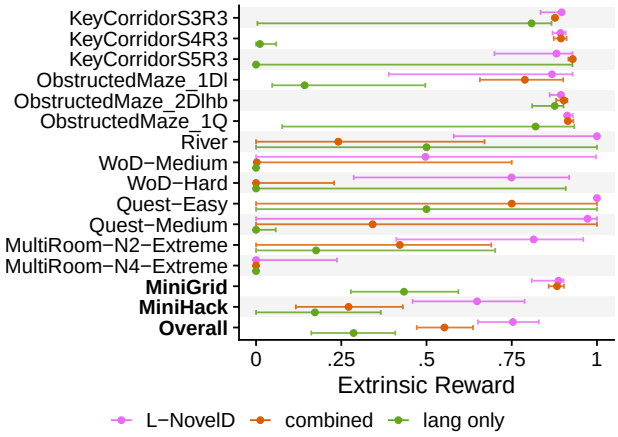


Figure S5: **L-NovelD ablations**. IQMs for Full L-NovelD, NovelD with combined state/language input, and language-only L-NovelD across environments. Plot elements same as Figure 4.

message over and over again (this is important to encourage exploration).

Results are in Figure S5. They show that using the language reward alone results in uniformly worse performance across environments, suggesting that it can still be helpful to provide a simpler navigation-based bonus to encourage exploration. Additionally, while combining the state and language into a single embedding works perfectly well for MiniGrid, it does not work as well for MiniHack tasks, suggesting that the additional flexibility afforded by the separate L-NovelD term can be helpful in certain settings. In principle we believe that it might be possible to tune the state and language embedding sizes of combined NovelD to see comparable performance to L-NovelD, but the point of L-NovelD is to clarify the contributions made by both the language and state and make it easier to trade-off between the two.

E Full description of tasks and language

Here we describe each task in MiniGrid and MiniHack in detail, and enumerate the list of messages available in each task. Examples of all tasks explored in this work are located in Figure S7.

E.1 MiniGrid

Language. As described in Section 6.1, the set of possible descriptions in the BabyAI language 652 involves *goto, open, pickup*, and *putnext* commands which can be applied to boxes, doors, and balls, optionally qualified by color. These messages can be grouped into 66 message "templates" where specific colors are replaced with a placeholder <C>, as shown in Figure S6. Not all messages needed for success on MiniGrid tasks, nor achieved by expert policies during training. We explain which messages are needed and/or encountered for each task below.

KeyCorridorS{3,4,5}R3. In these tasks (Figure S7a–c), the agent is tasked with picking up a ball behind a locked door. To do so, it must first find and pick up key which is hidden in (possibly several) nested rooms, return to the locked door, unlock the door, place the key down, and pick up the ball. The agent start location, object colors, and room/door positions are randomized across seeds.

A typical policy trained on each task will encounter anywhere from 92 (KeyCorridorS3R3) to 141 messages (KeyCorridorS{4,5}R3) throughout training. Regardless of the environment size, a successful trajectory requires encountering anywhere from 8–12 messages: *go to the door* (up to 3x), *open the door* (up to 3x), *go to the key*, *pick up the key*, *go to the* [locked] *door*, *open the* [locked] *door*, *go to the ball*, *pick up the ball*.

ObstructedMaze-{1Dl,2Dlhb,1Q}. In these tasks (Figure S7d–f), the agent is also tasked with picking up a ball behind a locked door. In the easier 1Dl task, the key is in the open; in the harder tasks, the keys are hidden in boxes which must be opened, and the doors are blocked by balls which must be moved to access them.

A typical policy trained on each task will encounter anywhere from 66 messages (ObstructedMaze-1Dl) to 244 messages (ObstructedMaze-1Q) throughout training. Of these, a successful trajectory requires anywhere from 6 messages in ObstructedMaze-1Dl (*go to the key*, *pick up the key*, *go to the door*, *open the door*, *go to the ball*, *pick up the ball*) to 11 messages in ObstructedMaze-1Q (*go to the door*, *open the door*, *go to the box*, *open the box*, *pick up the key*, *go to the ball*, *pick up the ball*, *go to the door*, *open the door*, *go to the ball*, *pick up the ball*).

E.2 MiniHack

Language. As discussed previously, it is difficult to enumerate all messages in MiniHack, though we describe the messages encountered for each environment below, with a raw dump in Appendix F. We perform some preprocessing on the messages to prevent unbounded growth: we replace all numbers (e.g. *3 flint stones*, *2 flint stones*, etc) with a single variable *N*; we fix wands encountered in the environment to be a single Wand of Death (instead of having a wand of over 30 different types); items that can be arbitrarily named with random strings (e.g. *a scroll labeled zelgo mer*; *a dog named Hachi* have their names removed; finally, we cap the number of unique messages for each environment at 100.

```
go to the <C> ball
go to the <C> box
go to the <C> door
go to the <C> key
go to the ball
go to the box
go to the door
go to the key
open the <C> box
open the <C> door
open the box
open the door
pick up the <C> ball
pick up the <C> box
pick up the <C> key
pick up the ball
pick up the box
pick up the key
put the <C> ball next to the <C> ball
put the <C> ball next to the <C> box
put the <C> ball next to the <C> door
put the <C> ball next to the <C> key
put the <C> ball next to the ball
put the <C> ball next to the box
put the <C> ball next to the door
put the <C> ball next to the key
put the <C> box next to the <C> ball
put the <C> box next to the <C> box
put the <C> box next to the <C> door
put the <C> box next to the <C> key
put the <C> box next to the ball
put the <C> box next to the box
put the <C> box next to the door
put the <C> box next to the key
put the <C> key next to the <C> ball
put the <C> key next to the <C> box
put the <C> key next to the <C> door
put the <C> key next to the <C> key
put the <C> key next to the ball
put the <C> key next to the box
put the <C> key next to the door
put the <C> key next to the key
put the ball next to the <C> ball
put the ball next to the <C> box
put the ball next to the <C> door
put the ball next to the <C> key
put the ball next to the ball
put the ball next to the box
put the ball next to the door
put the ball next to the key
put the box next to the <C> ball
put the box next to the <C> box
put the box next to the <C> door
put the box next to the <C> key
put the box next to the ball
put the box next to the box
put the box next to the door
put the box next to the key
put the key next to the <C> ball
put the key next to the <C> box
put the key next to the <C> door
put the key next to the <C> key
put the key next to the ball
put the key next to the box
put the key next to the door
put the key next to the key
```

Figure S6: **Full list of possible MiniGrid messages**. Messages are synthesized with the BabyAI (Chevalier-Boisvert et al., 2019) grammar, then divided into 66 templates. <C> is one of 6 possible colors: *grey*, *green*, *blue*, *red*, *purple*, *yellow*.

River. In this task (Figure S7g), the agent must cross the river located to the right on the environment, which can only be done by planning and pushing at least two boulders into the river in a row (to form a bridge over the river). A typical policy will encounter 14 distinct messages during training (Appendix F.1), of which 7–8 messages (3–4 unique) are seen during training (*with great effort you move the boulder*, *you push the boulder into the water.*, *now you can cross it!*; these messages must be repeated twice).

Wand of Death ({Medium,Hard}). In these tasks (Figure S7i–j; also described in the main text), the agent must learn to use a *wand of death*, which can zap and kill enemies. This involves a complex sequence of actions: the agent must find the wand, pick it up, choose to *zap* an item, select the wand in the inventory, and finally choose the direction to zap (towards the minotaur which is pursuing the player). It must then proceed past the minotaur to the goal to receive reward. Taking these actions out of order (e.g. trying to *zap* something with nothing in the inventory, or selecting something other than the wand) has no effect.

In the Medium environment, the agent is placed in a narrow corridor with the wand somewhere in the corridor, and the minotaur is asleep (which gives the agent more time to explore). In the Hard environment, the agent is placed in a larger room where it must first find the wand, with the added challenge that the minotaur is awake and pursues the player, leading to death if the minotaur ever touches the player.

In both Medium and Hard environments, a policy encounters around 60 messages (Appendix F.2), of which 7 messages (7 unique) are typically necessary to complete the task (*you see here a wand*, *f - a wand*, *what do you want to zap?*, *in what direction? you klil the minotaur!*, *welcome to experience level 2*, *you see here a minotaur corpse*).

Quest ({Easy,Medium}). These tasks (Figure S7j–k) require learning to use a *Wand of Cold* to navigate over a river of lava while simultaneously fighting monsters. The agent spawns with a Wand of Cold in its inventory, and must learn to zap the Wand of Cold at the lava, which freezes it and forms a bridge over the lava.

In the Easy environment, the agent must first cross the lava river, then survive fights with one or two monsters to the staircase at the end of the hall. In the Medium environment, the agent must first fight several monsters *before* crossing the lava river. There is an additional challenge: note the narrow corridor before the main room in the Medium environment. If the agent runs into the room and tries to fight the monsters all at once, it quickly will become overwhelmed and die; to successfully kill all monsters, the agent must learn to use the narrow corridor as a "bottleneck", first baiting then leading the monsters into the corridor, until all are defeated. It can then use the Wand of Cold to cross the lava bridge to the staircase beyond.

In both Easy and Medium environments, a typical policy encounters the maximum of 100 messages (Appendix F.3). Very few messages, besides *what do you want to zap? / in what direction ? / the lava cools and solidifies* are *required* to complete an episode, since there is a large variety of monsters faced and uncertainty in their behaviors, which can trigger additional messages. However, highly efficient expert play typically encounters 8–12 messages (7–10 unique) in the Easy environment and 30–40 messages (8–12 unique) in the Medium environment: besides freezing the lava to cross the environment, the rest of the messages are combat related (e.g. *you kill the enemy!*, *an enemy corpse*, additional zapping commands, etc).

MultiRoom-N{2,4}-Extreme. These asks (Figure S7l–m) are ported from the MiniGrid MultiRoom tasks, but with significant additional challenges. The ultimate task is to reach the last room in a sequence of interconnected rooms, but in the Extreme versions of this task, the walls

are replaced with lava (which result in instant death if touched), there are monsters in each room (which must be fought), and additionally the doors are locked and must be "kicked" open (which requires that select the kick action, then kick in the direction of the door).

In both environments, a typical policy encounters the maximum of 100 messages (Appendix F.4). In a successful trajectory, an agent encounters 20–30 (12–14 unique) messages in MultiRoom-N2 and 60–70 messages (16–18 unique) in MultiRoom-N4. These messages are mostly combat-related (*the enemy hits!*, *you hit the enemy!*, *you kill the enemy*, with repeated messages relating to kicking down doors (*in what direction?*, *as you kick the door, it crashes open*, *WHAAAAM!*, sometimes repeated up to 3 times per door, of which there are 1–3 doors).

F Raw MiniHack Messages

Messages are located on the next page. Each shows the list of messages encountered by a single agent trained on an environment in the corresponding categories. Separate training runs will encounter different messages.

F.1. River

```
the stairs are solidly fixed to the floor.
with great effort you move the boulder.
that was close.
it’s solid stone.
you don’t have anything to zap.
you try to move the border, but in vain.
you try to crawl out of the water.
there is a boulder here, but you cannot lift any more.
there is nothing here to pick up.
never mind.
pheew!
you push the boulder into the water.
now you can cross it!
it sinks without a trace!
```

F.2. WoD-{Medium,Hard}

```
c - a uncursed flint stone ( in quiver pouch ).
c - n uncursed flint stone ( in quiver pouch ).
in what direction?
it's a wall.
you don't have anything to zap.
you see here a wand.
you see here a uncursed flint stone.
you see here n uncursed flint stones.
there is nothing here to pick up.
what a strange direction!
the stairs are solidly fixed to the floor.
there is a staircase up here.
f - a wand.
the death ray bounces!
that is a silly thing to zap.
what do you want to throw?
what do you want to zap?
the wand glows and fades.
the death ray whizzes by you!
nothing happens.
b - a + 2 sling.
the wand turns to dust.
b - a blessed + 2 sling.
the flint stone misses the minotaur.
a wand shatters into a thousand pieces!
c - n uncursed flint stones.
c - a uncursed flint stone.
the flint stone hits another object.
the flint stone hits another object and falls down the stairs.
you see here a minotaur corpse.
welcome to experience level 2.
the flint stone hits the minotaur.
the flint stone hits other objects.
continue? ( q )
from the impact, the other object falls.
movement is very hard.
you stagger under your heavy load.
you kill the minotaur!
the sling misses the minotaur.
you have much trouble lifting a minotaur corpse.
the flint stone falls down the stairs.
the flint stone hits the minotaur!
the death ray misses the minotaur.
you see here a + 2 sling.
from the impact, the other objects fall.
the minotaur butts!
the sling hits the minotaur!
the sling hits the minotaur.
the wand misses the minotaur.
the wand falls down the stairs.
the sling hits another object.
g - a minotaur corpse.
you have extreme difficulty lifting a minotaur corpse.
the wand hits another object.
you see here a blessed + 2 sling.
the minotaur hits!
the flint stone hits other objects and falls down the stairs.
the sling hits another object and falls down the stairs.
the wand hits another object and falls down the stairs.
your movements are now unencumbered.
your movements are slowed slightly because of your load.
you can barely move a handspan with this load!
```

F.3. Quest-{Easy,Medium}

```
a enemy blocks your path.
a crude dagger misses you.
c - a uncursed flint stone ( in quiver pouch ).
f - a horn.
g - a wand.
h - a enemy corpse.
h - a crude dagger.
h - a scroll.
h - n darts.
i - a gem.
the wand glows and fades.
the enemy just misses!
the enemy hits!
the enemy picks up a uncursed flint stone.
the enemy throws a crude dagger!
the enemy touches you!
the enemy bites!
the enemy wields a crude dagger!
the enemy misses!
the enemy thrusts her crude dagger.
the stairs are solidly fixed to the floor.
the bolt of cold bounces!
the flint stone misses the enemy
the lava cools and solidifies.
in what direction?
that is a silly thing to zap.
it's a wall.
you are almost hit by a crude dagger.
you are hit by a crude dagger.
you get zapped!
you stop at the edge of the lava.
you kill the enemy
you miss the enemy
you destroy the enemy
there is nothing here to pick up.
what a strange direction!
what do you want to zap?
your movements are slowed slightly because of your load.
invalid direction for'g'prefix.
h - a wand.
h - n throwing stars.
i - a potion.
the enemy throws a dart!
the enemy escapes the dungeon!
the bolt of cold hits you!
the flint stone hits the enemy
you are hit by a dart.
a lit field surrounds you!
h - a skull cap.
h - a food ration.
h - a towel.
h - a dart.
i - a food ration.
i - a scroll.
i - n darts.
the bolt of cold hits the enemy
the dart misses the enemy
j - a dart.
you are almost hit by a dart.
you cannot escape from the enemy
```

```
a dart misses you.
h - a potion.
the enemy is killed!
c - n uncursed flint stone ( in quiver pouch ).
i - a wand.
the wand turns to dust.
the crude dagger hits the enemy
the crude dagger slips as the enemy throws it!
you don't have enough stamina to move.
you don't have anything to zap.
you rebalance your load.
nothing happens.
movement is difficult.
i - a conical hat.
you pull free from the enemy
your movements are only slowed slightly by your load.
h - a ring.
h - a egg.
i - a crude dagger.
h - a cursed crude dagger.
i - a ring.
the enemy picks up n crude daggers.
the enemy wields n crude daggers!
the enemy thrusts one of her crude daggers.
the bolt of cold whizzes by you!
the crude dagger welds itself to the goblin's hand!
h - a tripe ration.
you hit the enemy
your movements are now unencumbered.
h - a gem.
i - a enemy corpse.
the enemy picks up a gem.
h - a apple.
the enemy picks up a crude dagger.
h - a clear potion.
i - a skull cap.
you have a little trouble lifting h - a enemy corpse.
you have a little trouble lifting i - a food ration.
the crude dagger misses the enemy
```

F.4. MultiRoom-N{2,4}-Extreme

```
a crude dagger misses you.
a dart misses you.
g - a skull cap.
g - a enemy corpse.
g - a food ration.
h - a enemy corpse.
h - a lock pick.
h - a crude dagger.
whammm!!!
the enemy is killed!
the enemy just misses!
the enemy hits!
the enemy throws a crude dagger!
the enemy touches you!
the enemy jumps, nimbly evading your kick.
the enemy bites!
the enemy wields a crude dagger!
the enemy misses!
the enemy thrusts her crude dagger.
the little dog misses the enemy
the crude dagger misses the enemy
the kitten jumps, nimbly evading your kick.
the kitten bites the enemy
the kitten eats a enemy corpse.
the kitten misses the enemy
in what direction?
as you kick the door, it crashes open!
that hurts!
it burns up!
you are hit by a crude dagger.
you get zapped!
you see no door there.
you hit the enemy
you kill the enemy
you miss the enemy
you cannot escape from the enemy
you pull free from the enemy
you kick the enemy
you kick the kitten.
you kick at empty space.
you destroy the enemy
you strain a muscle.
you swap places with your kitten.
this door is locked.
what a strange direction!
do you want your possessions identified? ( n )
your leg feels better.
never mind.
really attack the little dog? ( n )
really attack the little dog? ( n ) n
really attack the kitten? ( n )
dumb move!
ouch!
g - a potion.
h - a potion.
h - a tripe ration.
the enemy escapes the dungeon!
the enemy misses the little dog.
the crude dagger hits the enemy
the dart misses the enemy
you are almost hit by a dart.
you swap places with your little dog.
do you want to see your attributes? ( n )
thump!
g - n darts.
h - n darts.
i - a enemy corpse.
j - a violet gem.
the enemy blocks your kick.
the enemy strikes at your displaced image and misses you!
the enemy throws a dart!
the little dog bites the enemy
the corpse misses the enemy
you are almost hit by a crude dagger.
you are hit by a dart.
you hear the rumble of distant thunder...
g - a crude dagger.
g - a yellowish gem.
g - n fortune cookies.
h - a wand.
h - n arrows.
the enemy picks up a crude dagger.
the enemy misses sirius.
the crude dagger slips as the enemy throws it!
you stop.
you swap places with hachi.
you swap places with sirius.
hachi misses the enemy
sirius bites the enemy
sirius misses the enemy
h - a skull cap.
sirius is in the way!
g - a scroll.
h - a faded pall.
h - a yellowish gem.
j - a enemy corpse.
the saddled pony drops a enemy corpse.
the saddled pony bites the enemy
g - a ring.
the saddled pony picks up a enemy corpse.
```

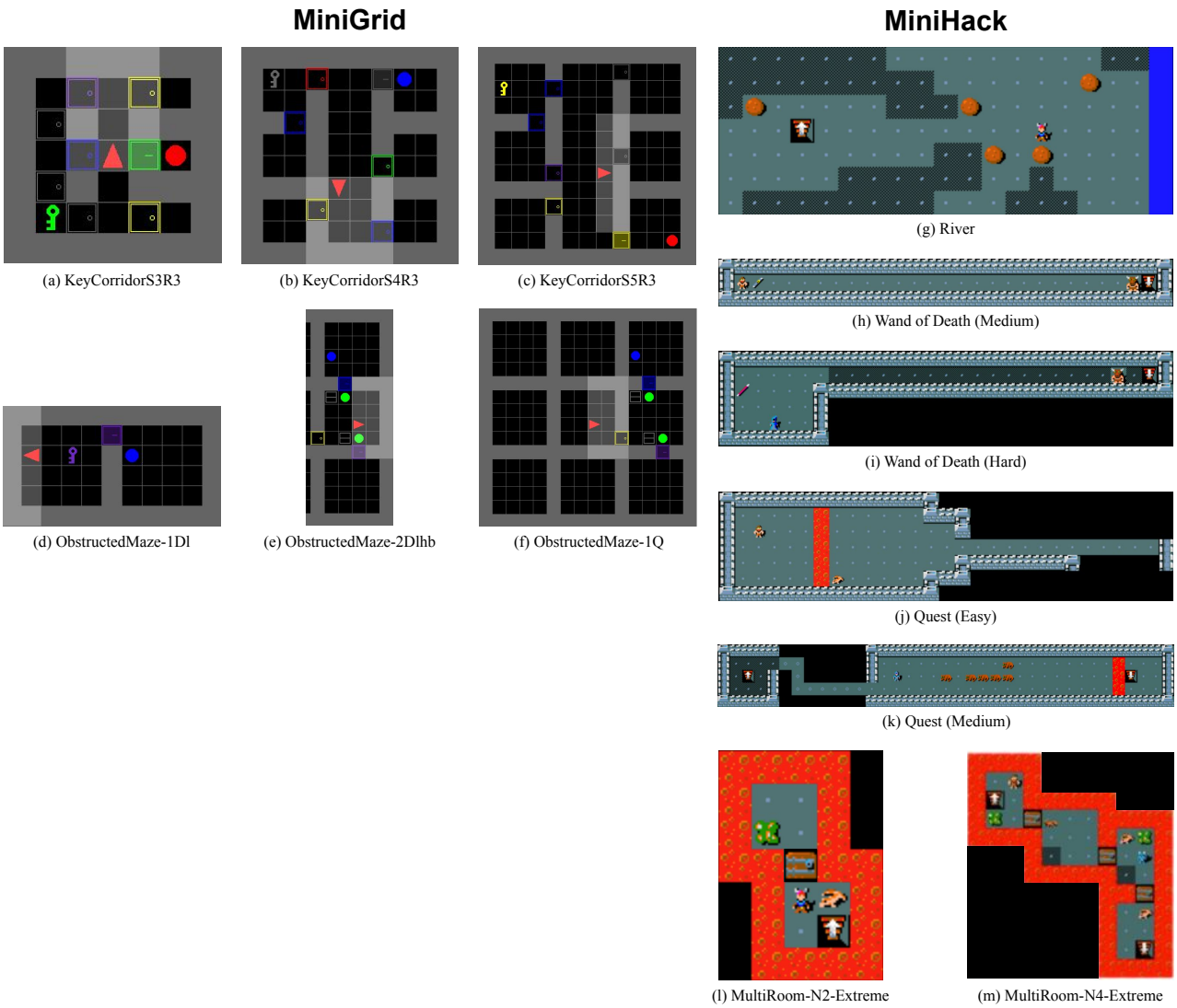


Figure S7: **Examples of all tasks evaluated in this work.** (a–f) MiniGrid tasks; (g–m) MiniHack tasks.