HAL open science

Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, Pierre-Yves Oudeyer

To cite this version:

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, et al.. Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning. International Conference on Machine Learning 2023, Jul 2023, Honololu, Hawaii, United States. hal-03970122v3

HAL Id: hal-03970122

https://hal.science/hal-03970122v3

Submitted on 6 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning

Thomas Carta*
Inria (Flowers)
University of Bordeaux, France
thomas.carta@inria.fr

Clément Romac*
Inria (Flowers)
University of Bordeaux, France
Hugging Face
clement.romac@inria.fr

Thomas Wolf
Hugging Face

Sylvain Lamprier
Univ Angers, LERIA,
SFR MATHSTIC, F-49000 Angers, France

Olivier Sigaud
Sorbonne Université, ISIR, Paris, France

Pierre-Yves Oudeyer
Inria (Flowers)
University of Bordeaux, France

Abstract

Recent works successfully leveraged Large Language Models' (LLM) abilities to capture abstract knowledge about world's physics to solve decision-making problems. Yet, the alignment between LLMs' knowledge and the environment can be wrong and limit functional competence due to lack of grounding. In this paper, we study an approach (named GLAM) to achieve this alignment through functional grounding: we consider an agent using an LLM as a policy that is progressively updated as the agent interacts with the environment, leveraging online Reinforcement Learning to improve its performance to solve goals. Using an interactive textual environment designed to study higher-level forms of functional grounding, and a set of spatial and navigation tasks, we study several scientific questions: 1) Can LLMs boost sample efficiency for online learning of various RL tasks? 2) How can it boost different forms of generalization? 3) What is the impact of online learning? We study these questions by functionally grounding several variants (size, architecture) of FLAN-T5.

1 Introduction

The recent rise of Transformer-based Large Language Models (LLMs) trained on massive text datasets in Natural Language Processing has led to models exhibiting impressive capabilities (e.g. natural language generation, question answering, reasoning, translation...) [Devlin et al., 2019, Brown et al., 2020, Rae et al., 2021, Chowdhery et al., 2022, Scao et al., 2022]. Recently, LLMs were shown to capture aspects of the physical rules in our world, e.g. about space Patel and Pavlick [2022], colors Abdou et al. [2021] or even affordances between bodies and objects Ahn et al. [2022]. This form of prior knowledge was exploited to suggest plans of action to solve goals in robotics Huang et al. [2022b], Ahn et al. [2022], Liang et al. [2022]. However, LLMs are known to suffer from a lack of grounding which prevents them from properly dealing with the meaning of inter-related concepts

*These authors contributed equally to this work

This is an extended version of the paper published in *Proceedings of the 40th International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023.

and their use for functional competence in interactive environments Mahowald et al. [2023]. Indeed, alignment between statistical structures in such LLMs and environments can be very limited, or even sometimes entirely wrong. This is partly due to 1) a training process (predicting next words) that is not directly incentivized to solve problems in an environment, 2) lack of abilities to intervene in the environment to identify causal structures; 3) lack in abilities to learn based on data collected as a result of interacting with the environment [Bender and Koller, 2020, Bisk et al., 2020].

In the literature, language grounding has referred to various related objectives Thill et al. [2014]. First, symbol grounding can be formulated as the general problem of connecting a symbol system [Harnad, 1990], internal to an agent, to the environment, in such a way that internal processing of these symbols can be used to to act appropriately in this environment. One dimension of this problem is associating "elementary" symbols, such as the names of objects, with invariant structures in high-dimensional perceptual modalities such as vision Cangelosi et al. [2010], Wiriyathammabhum et al. [2016]. Such a grounding, called "direct grounding", has been extensively studied in the past leading to various efficient methods [Alayrac et al., 2022, Radford et al., 2021, Lu et al., 2023], included in the context of robotic bodies Cangelosi and Stramandinoli [2018]. Another dimension is how to ground higher-order symbolic tokens, or abstract concepts, into elementary symbols, often through approaches such as distributional semantics Harris [1981], Boleda [2019]. This has been called "grounding transfer" Cangelosi and Stramandinoli [2018]. Beyond such mere associations, a key question about grounding is how internal processes that manipulate symbols can model, predict and control external physical and social processes: they need to be aligned on and constrained by these external dynamics and relational structures (at various levels of abstraction). This last notion of grounding, which we refer here as "functional grounding", is relative to a particular environment which may be the human physical environment but also more abstract interactive environments simulated in computers (where abstract physics can differ from human environments).

In this paper, we consider interactive textual worlds Côté et al. [2018], Jansen [2021], which are precisely designed to focus on these higher-level forms of functional grounding. In textual worlds, environments can encode rich forms of physical structures inspired by the ones in the human world, e.g. Wang et al. [2022], yet agents act and perceive in these environments only through the textual modality. In this context, this paper aims to make progress towards the following largely open question: how could LLMs be used as agent policies producing actions towards goals in interactive environments, perceiving the outcome of these actions, and incrementally grounding and updating their knowledge with the new observations they collect?

Building on recent works successfully using Reinforcement Learning (RL) to finetune LLMs for natural language generation tasks [Stiennon et al., 2020, Ouyang et al., 2022, Ramamurthy et al., 2022], we propose the first study about functional grounding of LLMs through incremental online RL. In particular, we aim at empirically answering the following open scientific questions:

• **Q1. Sample efficiency** How fast can an LLM adapt and learn to solve various spatial and navigation problems specified in natural language? How does the use of pre-trained knowledge from LLM boosts sample efficiency?

• **Q2. Generalization to new objects**: Once functionally grounded, how can an LLM generalize to various kinds of changes about objects, yet staying in trained tasks?

• **Q3. Generalization to new tasks**: How can such an interactively trained LLM perform zero-shot generalization to new tasks? How does generalization depend on the kind of new tasks?

• **Q4. Impact of online interventions**: What is the empirical impact of grounding using online RL with incremental interactions in comparison with offline Behavioral Cloning from a dataset of expert trajectories?

To answer these scientific questions in Section 4, we present a functional grounding method for LLMs (see Figure 1 and Section 3), and transpose the BabyAI environment [Chevalier-Boisvert et al., 2019] into a textual version. Additionally, we aim to help the RL community further develop grounding techniques for LLMs in interactive environments by releasing, in addition of the code of this paper[1], a Python library named *Lamorel*[2] facilitating the use of LLMs at scale for RL practitioners. While many

[1] `https://github.com/flowersteam/Grounding_LLMs_with_online_RL`

[2] `https://github.com/flowersteam/lamorel`

tools already exist for LLMs and NLP tasks, moving to an RL setting with interactive environments requires adaptations (e.g. very frequent need of fast inference to compute action probabilities) making previous tools not well suited for RL practitioners (see Section 3.4).

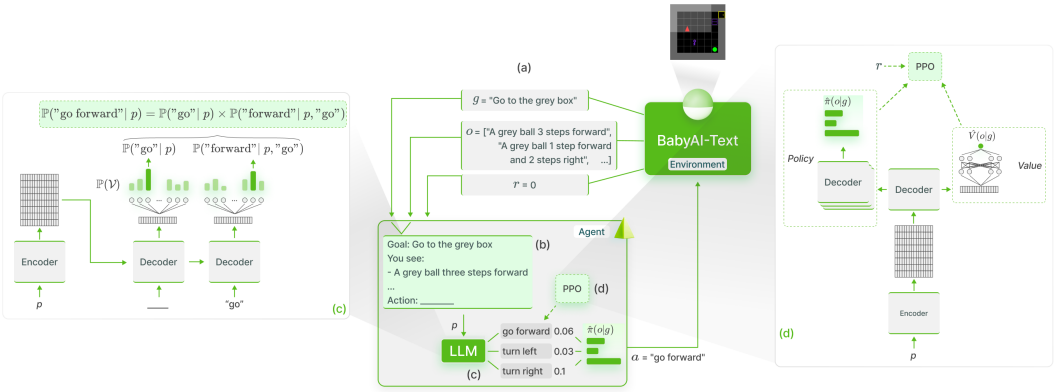


Figure 1: **The GLAM method: we use an LLM as agent policy in an interactive textual RL environment (BabyAI-Text) where the LLM is trained to achieve language goals using online RL (PPO), enabling functional grounding**. (a) BabyAI-Text provides a goal description for the current episode as well as a description of the agent observation and a scalar reward for the current step. (b) At each step, we gather the goal description and the observation in a prompt sent to our LLM. (c) For each possible action, we use the encoder to generate a representation of the prompt and compute the conditional probability of tokens composing the action given the prompt. Once the probability of each action is estimated, we compute a softmax function over these probabilities and sample an action according to this distribution. That is, the LLM is our agent policy. (d) We use the reward returned by the environment to finetune the LLM using PPO. For this, we estimate the value of the current observation by adding a value head on top of our LLM. Finally, we backpropagate the gradient through the LLM (and its value head).

2 Related work

Language-conditioned RL We position our work in the Language-conditioned RL setting, where an *instruction-following* agent learns a policy that executes actions in an interactive environment in order to fulfill a language instruction [Luketina et al., 2019]. While several works studied this setting for various tasks in 2D or 3D environments [Hermann et al., 2017, Misra et al., 2017, Bahdanau et al., 2018, Colas et al., 2020, Chevalier-Boisvert et al., 2019], we here focus on text-only interactions (i.e. performing textual commands given textual observations) as in Shridhar et al. [2020]. However, our work studies how LLMs can not only encode this instruction [Hill et al., 2020] but also be directly used as agent policies choosing actions given the observation.

Textual environments for RL Many text-only environments have been used and developed [Jansen, 2021, Wang et al., 2022]. They usually implement high-level text commands along with very large action spaces and complex dynamics between entities, often aiming to study functional grounding of abstract policies. While these environments offer interesting properties, we had to introduce a new one given the purpose and constraints of our study. Dealing here with computationally expensive LLMs, we chose to trade complex action spaces for systematic experiments studying the questions of the introduction. Second, to perform an in-depth analysis of our functional grounding method, we focused on lower-level navigation skills in spatial environments (which lacks in most textual environments as the agent can usually just change room and has direct access to objects in a room). Moreover, several ablation studies shown in Appendix B.5 required precise control over the procedural generation (usually not offered by textual environments). For these reasons, we adapted the BabyAI platform [Chevalier-Boisvert et al., 2019] into a procedural text-only version that enables decoupling exploration challenges from perception challenges. Additionally, we are still able to use BabyAI's visualization tools to analyze trajectories (see Figure 1).

Foundation Models for decision making Foundation models trained on massive datasets were shown to exhibit impressive abilities along with fast adaptation to a wide range of downstream tasks in vision [Yuan et al., 2021], language [Devlin et al., 2019, Brown et al., 2020] and cross-modalities [Ramesh et al., 2021, Jiang et al., 2022, Alayrac et al., 2022]. While such abilities have been leveraged to provide reward to RL agents Gupta et al. [2022], Fan et al. [2022], a recent line of work started focusing on using Foundation Models (and in particular LLMs) to guide agents policy.

First, SayCan [Ahn et al., 2022], Code as Policies [Liang et al., 2022] and Inner Monologue [Huang et al., 2022b] used LLMs as high-level planners in robotics setups. Because their LLM is not directly used as agent policy for low-level actions and is not grounded using its interactions with the environment, Ahn et al. [2022] had to use an external affordance function to re-rank the actions proposed by the LLM. Similarly, Yao et al. [2022] also featured a closed-loop feedback between an LLM that is the planner and an agent that is the actor but this time in a textual environment. Expanding on this, Dasgupta et al. [2022] added a reporter observing the environment and reporting useful information to the planner. While hinting at the usefulness of prior knowledge contained in LLMs for embodied tasks, these works are limited by the absence of grounding.

Second, several works proposed to first finetune LLMs on expert trajectories before using them in the environment. Using their ScienceWorld benchmark, Wang et al. [2022] showed that LLMs finetuned using Behavioral Cloning performed worse than a much smaller and randomly initialized Deep Q-Network trained using RL supporting the hypothesis that grounding in the environment through direct interactions is crucial. Finally, Reid et al. [2022] reused LLMs to perform offline RL in non-linguistic environments leveraging the internal structures learned by LLMs but no longer using words or symbols they were trained to manipulate (Takagi [2022] investigated how these internal structures can be relevant for unrelated tasks).

Finally, one may also pretrain a policy using Behavioral Cloning or offline RL from expert trajectories before finetuning it with interactions with an environment. Related to our work, the Online Decision Transformer [Zheng et al., 2022] first uses offline RL to pretrain a transformer model and eventually finetunes it with online RL. But compared to our study, it did not use a general Language Modeling pretraining objective and therefore did not study functional grounding of language symbols.

Finetuning LLMs with RL Recent works successfully leveraged RL to finetune LLMs. RL was used in particular to improve alignment between generated text and human preferences [Stiennon et al., 2020, Ouyang et al., 2022, Ramamurthy et al., 2022]. In this Reinforcement Learning from Human Feedback (RLHF) framework, text generation is viewed as a sequential decision-making problem where each "action" of the LLM is a new token and the "state" corresponds to the prompt. Most of these methods used PPO [Schulman et al., 2017] to finetune their LLMs using a reward function learned on a dataset of collected human interactions. With this technique, Ouyang et al. [2022] managed to generate more human-aligned outputs despite having a model (InstructGPT) with 100 times fewer parameters than GPT-3 [Brown et al., 2020]. While our work shares the PPO-based finetuning with RLHF, our setup diverges from it in multiple aspects. First, our LLM is functionally grounded using an external task-conditioned reward from the environment (which happens to be sparse in our BabyAI-Text environment) and not a learned reward model. Second, the RLHF setup has no external environment dynamics controlling the next state given a previous state and an action (the next state in RLHF is just the previous state with the last generated token appended). In comparison, our work exposes an outer loop controlled by the environment whose dynamics, providing the next state and reward, are unknown to the LLM (in comparison to RLHF where the RL loop is an inner loop in the token generation process). We believe using RL to finetune LLMs can be taken from a broader perspective in which both our framework and RLHF are particular applications.

3 GLAM: Grounding LLMs with online RL

We introduce the GLAM method (for Grounded LAnguage Models) where an LLM is used as agent policy and is functionally grounded in an interactive environment using online RL, leveraging collected observations and rewards to improve itself towards achieving goals formulated in language. We detail this method in the following paragraphs and redirect the reader to Figure 1 for a schematic view. We first formalize the textual RL problem we tackle (a). Then, we detail how we use an LLM as agent policy to interact with BabyAI-Text (b, c). Finally, we explain how online RL finetuning is used to ground the LLM in BabyAI-Text (d).

3.1 Problem statement

We assume a textual RL setting where, given a language vocabulary $\mathcal{V}$, our environment returns an observation $o \in \mathcal{V}^N$ and a reward $r \in \mathbb{R}$ following an action $a \in \mathcal{A} \subset \mathcal{V}^N$ (i.e. actions are sequences of tokens). We also assume a task or goal description $g \in \mathcal{G} \subset \mathcal{V}^N$ which conditions the reward. Such an environment can be framed as a goal-augmented Partially Observable Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{G}, \mathcal{O}, \gamma)$ with $\mathcal{S}$ the state space, $\mathcal{A} \subset \mathcal{V}^N$ the action space, $\mathcal{G} \subset \mathcal{V}^N$ the goal space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \mapsto \mathbb{R}$ the goal-conditioned reward function, $\mathcal{O} : \mathcal{S} \mapsto \mathcal{V}^N$ the observation function mapping a state to a textual description and finally γ the discount factor.

In this work, we extend the BabyAI platform [Chevalier-Boisvert et al., 2019] initially designed for grounded language learning and propose a text-only extension named BabyAI-Text. We leverage BabyAI's inner procedurally generated minigrid environment where an agent navigates and interacts with objects through 6 text commands: *turn left, turn right, go forward, pick up, drop and toggle*. We also reuse the set of tasks introduced in BabyAI as well as their associated description along with the sparse scalar reward. Our key difference is the textual description $o \in \mathcal{V}^N$ of the agent's partial observation returned by BabyAI-Text instead of the symbolic representation initially returned by BabyAI (see Appendix A.2). We leverage BabyAI-Text in Section 4 to assess our grounding method.

3.2 LLMs as policies in interactive environments

In order to use the LLM as the policy in such a textual interactive environment, we gather the task description, the textual description of the current observation and the set of possible actions in a prompt used to feed the LLM. We chose a single arbitrary and simple prompt template (see Appendix C for examples) and did not perform any intensive prompt engineering. Indeed, as we finetune the LLM, we expect it to adapt to the chosen prompt template. Nonetheless, a more careful design of prompts could improve the results shown in Section 4.

Given this prompt, we now need the LLM to output a probability distribution over the possible actions $\mathbb{P}(\mathcal{A})$. For this, Huang et al. [2022a], Li et al. [2022], Wang et al. [2022] used the LLM to generate text. If the generated sequence of characters corresponds to one of the possible actions (i.e. $s \in \mathcal{A}$), this action is chosen by the agent. Otherwise, an ad-hoc mapping must be performed to select an action $a_i \in \mathcal{A}$ given s. As an alternative method, one could also use more standard RL practices by adding action heads - a Multi-Layer Perceptron (MLP) with $|\mathcal{A}|$ outputs - on top of the LLM. Finally, Ahn et al. [2022] proposed to directly use the LLM to compute the (log) probability of each action $a_i \in \mathcal{A}$ by computing the conditional probability of each token in action $a_i = \{w_0, ..., w_{|a_i|}\}$ given the prompt p:

$$\mathbb{P}_{LLM}(a_i|p) = \prod_{j=0}^{|a_i|} \mathbb{P}_{LLM}(w_j|p, w_{<j}) \tag{1}$$

with $\mathbb{P}_{LLM}(w_j|p, w_{<j})$ the probability computed by the LLM of token w_j given prompt p and previous tokens $w_{<j}$ (see (c) from Figure 1). This method suffers from requiring a forward pass on the LLM for each action to compute the probability of its sequence of tokens (especially in comparison to new action heads that require a single forward pass on the prompt to compute all actions' probability). However, it also has several advantages, in particular, 1) there is no need of potential ad-hoc mapping as when text is generated, 2) we use only pretrained operations from the LLM and leverage language modeling heads' prior and 3) this method is robust to any action space and can thus be used on any textual environment with no change.

For these reasons, we chose the latter method. We first use log probabilities instead of normalized probabilities using $\mathbb{LP}_{LLM}(a_i|p) = \sum_{j=0}^{|a_i|} log\, \mathbb{P}_{LLM}(w_j|p, w_{<j})$ in replacement of $\mathbb{P}_{LLM}(a_i|p)$ to avoid multiple normalization operations. We eventually normalize all the log probabilities to obtain a distribution over $\mathcal{A}$ using a softmax function:

$$\mathbb{P}(a_i|p) = \frac{e^{\mathbb{LP}_{LLM}(a_i|p)}}{\sum_{a_j \in \mathcal{A}} e^{\mathbb{P}_{LLM}(a_j|p)}}. \tag{2}$$

3.3 PPO finetuning

We now propose to leverage experiences gathered by the LLM to perform functional grounding. More formally, we aim to learn a policy $\pi : O \times \mathcal{G} \mapsto \mathbb{P}(\mathcal{A})$ that maximizes the expected discounted sum of rewards for any given goal $g \in \mathcal{G}$. We use for this the PPO algorithm [Schulman et al., 2017] that both learns a policy $\hat{\pi} : O \times G \mapsto \mathbb{P}(A)$ and a value function $\hat{V} : \mathcal{O} \times \mathcal{G} \mapsto \mathbb{R}$ approximating the true value $V(s, g) = \mathbb{E}_{a \sim \hat{\pi}(\mathcal{O}(s), g)} \big[R(s, g, a) + \gamma V(\mathcal{T}(s, a), g) \big]$.

As mentioned in Section 3.2, we compute the probability of each action $a_i \in \mathcal{A}$ using the likelihood computed by the LLM as $\hat{\pi}(a_i|o, g) = \mathbb{P}(a_i|p)$.

For value approximation, we we add an MLP with a single output for the value on top of the last layer of the first Decoder block (i.e. in place of the language modeling heads) in order to compute $\hat{V}(o|g) = \hat{V}(p)$ (see (d) from Figure 1). This position is explained by the fact that we use Encoder-Decoder LLMs in our experiments but our method could easily be used with Decoder-only models by attaching the value head to the Decoder block encoding the last token of the prompt.

3.4 Distributed LLM policies using *Lamorel*

Using LLMs to compute probabilities over action space is computationally expensive as it requires computing $\prod_{j=0}^{|a_i|} \mathbb{P}_{LLM}(w_j|p, w_{<j})$ for each action $a_i = \{w_0, ..., w_{|a_i|}\}$. When one uses very large LLMs (i.e. more than hundreds of million parameters), computing the probability of a single action already means performing a long forward pass over the whole network. As a result, computing the probability of each possible action at every step becomes very slow. Considering the number of interactions usually required to solve tasks in BabyAI (and by extension BabyAI-Text), performing online RL finetuning of LLMs easily became intractable with a single LLM distributed over multiple GPUs. To overcome this, we deployed N LLM workers each handling a subset of actions to score in parallel (allowing a quasi-linear time decrease with N). We add to this distributed inference the possibility to also perform distributed training (i.e. compute the gradient of minibatches in parallel and gather gradients before updating models). We wrap all this in a Python library named *Lamorel* designed for RL practitioners eager to use LLMs. It allows one to use LLMs as black-box but also to perform more advanced methods such as adding new heads on top of them. See Appendix E for more details.

4 Experiments

We design a set of experiments in BabyAI-Text aiming to provide answers for the scientific questions introduced in Section 1. In these experiments, we use Flan-T5 780M [Rae et al., 2021] for 1) the close link between its training corpus (containing instruction-following documents) and our language-conditioned interactive environment, and 2) its simple open-source access through the Hugging Face tools[3]. We apply our GLAM method to Flan-T5 (which we name GFlan-T5 in experiments below for Grounded Flan-T5) and compare it with three baselines. First, we also train a non-pretrained Flan-T5 where we only reuse the pretrained embedding layer and add action heads on top of it (see Figure 10 in appendices). As for GFlan-T5, we propagate the gradient through the entire graph (included the action heads here). We call this baseline NPAE-Flan-T5 (Non-Pretrained with Action heads and Embedding Flan-T5). We show in Appendix B.3 that using a non-pretrained Flan-T5 while keeping the scoring method fails. We also provide as a more classic RL baseline a DRRN [He et al., 2016] agent of approximately 1M parameters which is often used for TextWorlds. We especially reuse the implementation from Wang et al. [2022] which gave SOTA results and outperformed LLMs. At each step, we feed our 3 agents above with the following prompt template filled using the information returned by BabyAI-Text (see Appendix C for examples):

• A header listing what actions are accessible (but not necessarily useful) in the environment in the form of:
Possible action of the agent: <list of actions>

• The goal of the agent: *Goal of the agent: <goal>*

[3] `https://huggingface.co/docs/transformers/model_doc/flan-t5`

• The 3 previous observations and last 2 actions, used as a short-term memory required to complete BabyAI-Text tasks (in comparison, the DRRN uses recurrent layers to deal with short-term memory requirements):
Obs. 0: <description from BabyAI-Text at step $t-2$ >
Action 0:<action chosen by the agent at step $t-2$ >
Obs. 1: <description from BabyAI-Text at step $t-1$ >
Action 1: <action chosen by the agent at step $t-1$ >
Obs. 2: <description from BabyAI-Text at step t >
Action 2: <the next action to be chosen by the agent>

Finally, as BabyAI-Text simply provides an alternative mapping of observations, we add as an indication the performance of the PPO agent used in [Chevalier-Boisvert et al., 2019] that runs on BabyAI rather than BabyAI-Text (i.e. using symbolic observations instead of textual descriptions) and name this agent Symbolic-PPO in results below. In Appendix B.2, we show that symbolic observations provided by BabyAI encode biases that ease learning compared to text descriptions. However, even with this advantage, GFlan-T5 outperforms Symbolic-PPO in all our setups.

We first study Q1 by training the different agents in a multi-task setting assessing their efficiency at learning the different tasks. We then address questions Q2, Q3 and Q4 using a set of generalization experiments (Figure 4) on the zero-shot abilities of the resulted trained agents mostly inspired from [Colas et al., 2020] and [Valmeekam et al., 2022]. We report their average success rate as well as standard deviation. We compare the results of GFlan-T5, DRRN as well as Flan-T5 (i.e. the LLM used in GFlan-T5 but before our finetuning) to show how our grounding method impacted it. All results below are given with their 99% confidence interval (mathematical details are given in Appendix G).

4.1 How fast can an LLM adapt and learn to solve tasks? (Q1)

In order to study question Q1, we train our agents for 1.5 million steps in BabyAI-Text where each episode is a task randomly sampled from the following:

- **Go to <object>**, a simple navigation task that requires reasoning abilities to choose the right plan given objects' position;
- **Pick up <object>**, a reasoning task that combines navigation tasks;
- **Put <object A> next to <object B>**, which requires first reaching <object A>, picking it up, reaching <object B> and finally dropping <object A> next to <object B>;
- **Pick up <object A> then go to <object B>** and **Go to <object B> after pick up <object A>**, both serving to test reasoning abilities on temporal sequences;
- **Unlock <door>**, a task that includes inferring that a key is needed to unlock the door, finding the right key (i.e. the one colored as the door) and eventually using the toggle action with the key on the door.

In each task, the agent must navigate in one procedurally generated room with 8 distractors (i.e. useless objects for the completion of the task).

We plot the mean and standard deviation of the success rate (i.e. 1 if the goal has been reached, 0 otherwise) over 4 seeds of GFlan-T5, NPAE-Flan-T5, DRRN and Symbolic-PPO in Figure 2. In addition, we also monitor the evolution of probability of each possible action on a set of 11 evaluation prompts to assess agents' abilities to solve each task in Appendix C. By plotting the evolution of the distribution over possible actions in Figure 18, we better grasp how and when the agents learn skills (e.g. navigation skills).

Looking at the evolution of the average success rate, GFlan-T5 quickly reaches 0.8 after only 250.000 steps (and 0.9 after approximately 600.000 steps). In comparison, both DRRN and NPAE-Flan-T5 are still under 0.2 after 1.5 million steps. Even when compared to Symbolic-PPO, which uses symbolic observations (easier to process than language as shown in Appendix 7), GFlan-T5 exhibits a drastically better sample efficiency with Symbolic-PPO almost reaching 0.4 after 1.5 million steps. Figure 18 and Table 2 highlight how GFlan-T5 leverages its knowledge about the relationships between entities to learn navigation tasks in less than a hundred updates.

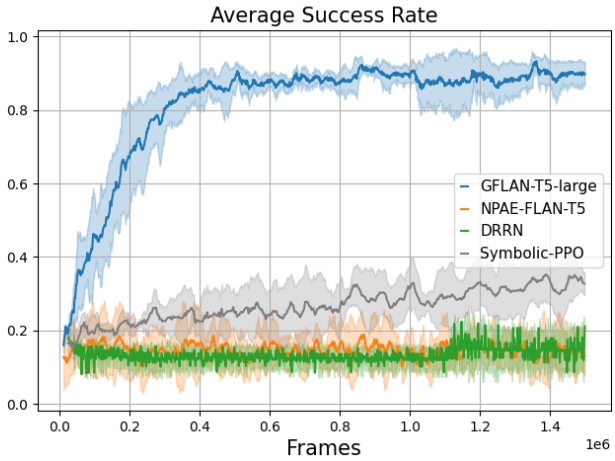
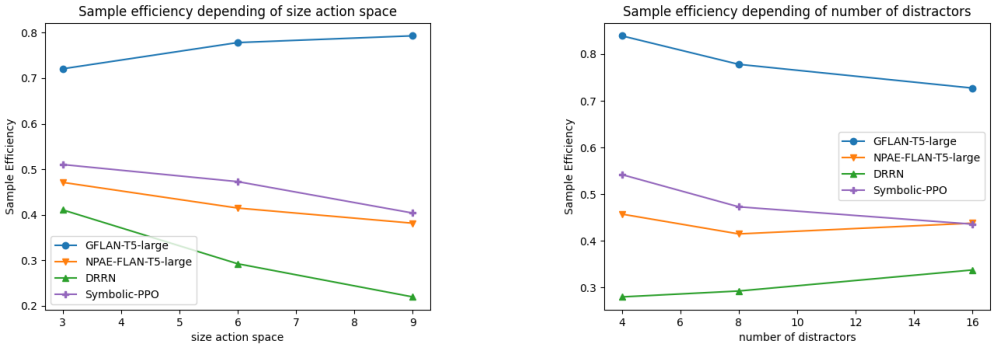


Figure 2: **Q1. Sample efficiency**: Evolution over 4 seeds of the average success rate and standard deviation on all Q1 tasks.



(a) Impact of the action space size (3, 6 or 9 actions with always only 3 useful actions).

(b) Impact of the number of distractors.

Figure 3: Impact of the aciton space size and number of distractors on the sample efficiency measure (Equation (3)). We report results averaged over 2 seeds for training on the *Go To* task.

The failure of NPAE-Flan-T5 both highlights how GFlan-T5 leverages the LLM's pretrained knowledge to deal with the proposed tasks and how the finetuning method helps achieve the grounding objective. Furthermore, the fact that GFlan-T5 strongly outperforms Symbolic-PPO and the latter is better than NPAE demonstrates how language can be used as a tool to scaffold learning if already acquired. It also exaplins how counterproductive it can be if one asks an agent to both learn a task and language at the same time (see Appendix B.2 for further results).

We now perform a deeper analysis of this sample efficiency by studying the impact of varying the action space and the number of distractors. We provide both the evolution of success rate and a sample efficiency measure SE:

$$SE = \frac{1}{T} \sum_{t=0}^{T} SR_t \tag{3}$$

where T is the number of steps or frames seen and SR the success rate at frame t.

4.1.1 Impact of the dimension of the action space

In this experiment, we test the sensitivity of LLMs to the size of the action space by using 3 different action spaces when trained on the *Go to <object>* task:

- The **restricted** action space composed of the only 3 useful actions: `turn left`, `turn right`, `go forward`.
- The **canonical** action space composed of the 6 actions that can be performed in the environment with 3 useful and 3 useless actions that are `pick up`, `drop` and, `toggle` (they are useless here as the agent is only navigating).
- The **augmented** action space composed of 9 actions (3 useful and 6 useless with `pick up`, `drop`, `toggle`, `sleep`, `do nothing` and `think`). The last three actions have been chosen such that they clearly have no use for the *Go To <object>* task and consequently should not impact an agent that has knowledge about the world.

We conduct our tests in an environment with 1 room, 8 distractors and report results in Figure 3a. Results show no impact on GFlan-T5, while the performance of other agents decreases with larger action spaces. We hypothesize that this is due to the LLM's ability to discard useless actions quickly at the beginning of finetuning.

4.1.2 Impact of the number of distractors

Similarly, we expect LLMs to be less sensitive to variations in task complexity. We assess this by plotting the evolution of sample efficiency (Equation (3)) for 4, 8 and 16 distractors. We conduct these tests in an environment with 1 room and observe a slight performance loss from GFlan-T5 when the number of distractors increases (Figure 3b). In comparison, Symbolic-PPO degrades as the number of distractors increases with a success rate decreasing by 38% from 4 to 16 distractors whereas the GFlan-T5 success rate only decreases by 14%. We hypothesize that the LLM manages to focus on the relevant aspect of the environment quickly.

Thus, GFlan-T5 seems robust with similar learning curves when one increases the action space size (from 3 to 9 actions with only 3 useful ones) or the number of distractors (from 4 to 16) We also provide in Appendix B an additional ablation analyzing the impact of the LLM's size B.4. Results highlight that the number of parameters has a high impact on the learning process. Indeed, we observe a strong difference on sample efficiency and asymptotic performance between a small LLM (80 million parameters) and the 780 million parameters we used here. We also plot the full learning curves for the ablation on the action space size and the number of distractors in appendices B.5.1 and B.5.2 respectively.

4.2 Q2. Generalization to new objects

In this section, we analyze how a functionally grounded agent can generalize its skills to new objects. Indeed, we expect our agents to focus on the the geometry of the environment (how objects are positioned and how their positioning is described), but not on the identity of the objects themselves (e.g. *Go to <object>* should be achieved even if the object has not been seen during training). We test if this property is present in our trained agents by measuring their zero-shot performance in two environments. First, an environment with nouns not in the training vocabulary (e.g. "tree")[4] and second, an environment with invented objects (made of an invented adjectives and an invented nouns such as *faze dax*).[5] We use the environment the agent has been finetuned on (i.e. without any word substitutions) as a control environment. Results in Figure 4 (Q2 part) indicate that GFlan-T5 is not affected when tasks contain out-of-vocabulary nouns. Moreover, even if the GFlan-T5's success rate decreases by 13% when it is in an environment with invented objects, it still retains strong performances compared to baselines. These results support the hypothesis that GFlan-T5 has functionally grounded the symbols that describe the geometry of the environment and the instructions (e.g. words such as "in front", or the meaning of "steps" as a distance measure)[6].

[4]The out-of-vocabulary nouns are given in Appendix H.1.

[5]The invented objects are given in Appendix H.2.

[6]See Section A.2 for more details on the geometry.

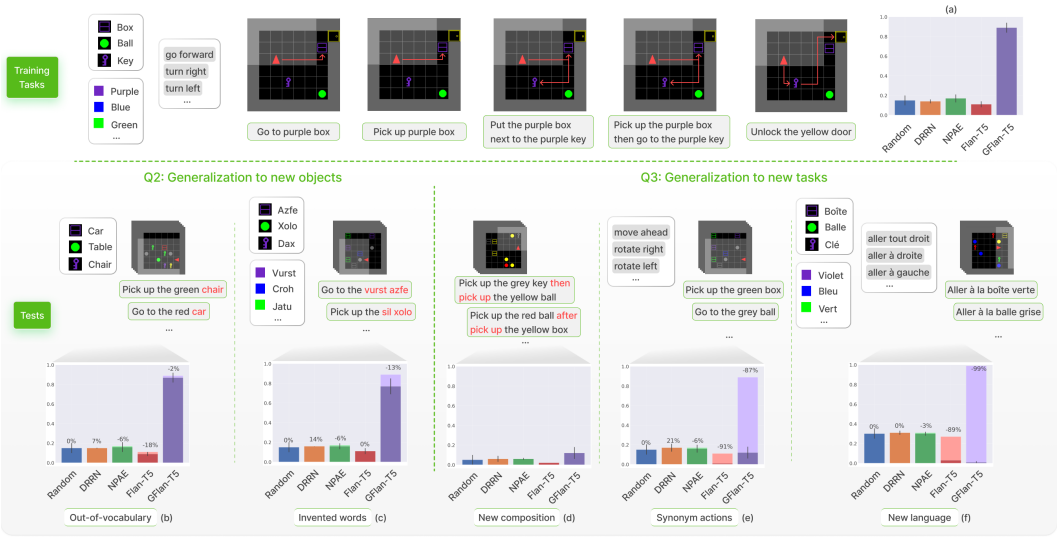


Figure 4: **Generalization tests**: We train all agents on a mix of 5 different tasks and evaluate their generalization abilities on 1000 test episodes (also containing a mix of these 5 tasks) (a). We compare them to two baselines: an agent choosing actions randomly (Random) and the zero-shot Flan-T5 (without any finetuning). We then perform several generalization studies to answer Q2 and Q3 by (b) substituting object names out-of-vocabulary names, (c) substituting objects and colors by invented words, (d) testing a new composition of tasks, (e) substituting actions by synonyms and (f) translating the whole environment to French for the *Go To* task. For each agent, we plot its mean success rate over 2 seeds along with the confidence interval and the delta with performance on the same task without any change (except for (d), on which no baseline result can be provided as this task is completely new).

4.3 Q3. Generalization to new tasks

In this Section, we perform generalization tests as in Section 4.2, but with new unseen tasks. Using these, we verify to what extent an agent is able to compose and generalize over the symbols it has grounded during finetuning.

Table 1: Generalization tests for Behavioral Cloning

	Environments	GFlan-T5	BC-GFlan-T5	BC-Bot	Random
Q4	Go To task no change	0.82 ± 0.02	0.69 ± 0.08	0.73 ± 0.07	0.30 ± 0.05
	Go To task with invented words	0.74 ± 0.004	0.7 ± 0.07	0.63 ± 0.08	

New composition of learned tasks: Pick up <object A> then/after pick up <object B> During finetuning, agents learn to do both 1) *Pick up <object A>* and 2) *Pick up <object A> then go to <object B>* or *Go to <object B> after pick up <object A>* tasks. We test in this experiment if an agent can compose grounded symbols to solve the new tasks *Pick up <object A> <then/after> pick up <object B>*. Results in Figure 4 (Q3 part) hint that, while all agents fail to solve these new tasks, GFlan-T5 outperforms other baselines by reaching an 0.12 success rate compared to Flan-T5 (0.07) or Random (0.05). These low results can be explained by the fact that none of the agents managed to master the *Pick up <object A> then go to <object B>* or *Go to <object B> after pick up <object A>* tasks during training (see Appendix C). More details about the grounding of "then" and "after" are given in the Appendix D.4.

Seen tasks with synonym actions In this task, we test the robustness of our agents to actions by replacing the actions used during training by synonyms. For instance, "go forward" is replaced with

"move ahead"[7]. We expect LLMs, which already learned to map words to an embedding space, to also ground synonyms as they ground words of the environment. In this environment (see Figure 4 Q3 part), the success rate of GFlan-T5 is 0.12 vs 0.01 for Flan-T5. Thus the grounding of some words (here the actions) also improves the grounding of their synonyms. However, we observe an 87% drop in performance compared to the original settings, which we assume is due to an over-fitting of the actions' vocabulary.

New language In order to understand how far agents can generalize, we test them with a language not seen during training (French). Knowing that Flan-T5 has been pretrained with a multilingual corpus and is able to translate simple sentences, we test whether grounding in GFlan-T5 has also impacted its manipulation of other languages. However, we observe that even only for a simple navigation task (i.e. *Go To*), the model fails to generalize to a new language with a success rate (0.02) worse than random (0.30). We hypothesize that when too many grounded symbols are modified at once, functional grounding fails to be transferred to this new subsystem of symbols. Complementary experiments that confirm and reinforce this result are presented in appendices D.2 and D.3.

4.4 What is the impact of using RL vs Behavioral Cloning for grounding? (Q4)

Finally, we study how online interactions with an environment, enabling learning through interventions and trial-and-error, improves grounding in comparison to pure Behavioral Cloning (BC). We compare a GFlan-T5 trained on the **Go To** task over 400000 steps with two baselines trained with Behavioral Cloning using 400000 transitions (see Appendix F.2). For the baseline called BC-GFlan-T5, transitions are collected from GFlan-T5 finetuned on the **Go To** task. For BC-Bot, transitions are collected using the BabyAI procedural bot achieving a success rate of 1.

In Table 1, we measure the success rate of GFlan-T5 and the baselines on two tasks: *Go To* and *Go To* with invented nouns and adjectives. First, once can see that GFlan-T5 outperforms all baselines in both tasks. Second, as GFlan-T5 does not achieve a success rate of 1 on the **Go To** task, its collected trajectories for BC can contain deceptive transitions in comparison to the ones collected by the bot. Hence, we obtain the expected result that BC-Bot outperforms BC-GFlan-T5. Finally, we expect our agents not to be affected by an environment where nouns and adjectives are replaced by invented ones in such navigation tasks. Experiments show that GFlan-T5 is less affected $(0.82 \rightarrow 0.74)$ than the BC-Bot $(0.73 \rightarrow 0.63)$. GFlan-T5 also performs better in the *invented words* task than the BC-GFlan-T5 (success rate of 0.7).

5 Conclusion

In this paper, we proposed the GLAM method for functional grounding (i.e. aligning internal symbols to external dynamics so that the agent can use them to solve tasks in the environment) of LLMs in interactive textual environments based on online RL. Using our new BabyAI-Text environment, we performed several experiments studying 4 scientific questions. We showed how GLAM, which requires almost no environment-specific modifications on the LLM, enables to drastically improve performances to solve RL tasks in this environment as compared to zero-shot use the LLM, to supervised finetuning and to RL finetuning of non-pretrained LLMs. We showed how it boosts both sample efficiency and generalization abilities in zero-shot tests (both to new objects and several new tasks). In addition to these key results, we provided in-depth ablations showing the effect of several parameters (e.g. size) on grounding. We believe this method can act as a milestone towards grounding and using LLMs in interaction with our world. However, this study still suffers several limitations, in particular the fact that current experiments are limited to a textual environment, and the computational inefficiency when scaling up the action space and the size of the LLM. This computational inefficiency constrained this paper to using a single environment and rather small LLMs. Yet, improving computational efficiency (or access to more computational resources) could enable to leverage recent multi-modal Foundation models [Alayrac et al., 2022]) for grounding LLMs in broader environments (e.g. to robotics setups [Lu et al., 2021, Ahn et al., 2022]). Parallel to this, a future direction would be to study how functionally grounding an LLM on a specific environment affects its zero-shot abilities but also its plasticity and ability to acquire new skills in

[7]A table giving all the used synonyms is given in Appendix H.3

other environments. Moreover, these results hint that using LLMs as agent policies opens an avenue for escaping the *Tabula-Rasa* RL setting and creating much more sample efficient RL agents.

Finally, the recent rise of real-world deployed applications using LLMs highlighted the various societal and ethical challenges of using such models in real-world scenarios. Similar to RLHF, our work studies how to better align LLMs (but this time to environments in which tasks must be solved). While our approach stands as a first important building block for future works making LLMs more in line with their environment, it is not designed to be ready for real-world deployment and thus we do not recommend to use it in such an applicative context.

Acknowledgments and Disclosure of Funding

Experiments presented in this paper were carried out using the HPC resources of IDRIS under the allocation 2022-[A0131011996] made by GENCI. This work also has received funding from the European Commission's Horizon Europe Framework Programme under grant agreement No 101070381 (PILLAR-robots project). We would also like to thank Victor Gondat for his kind help on schemas.

References

Mostafa Abdou, Artur Kulmizev, Daniel Hershcovich, Stella Frank, Ellie Pavlick, and Anders Søgaard. Can language models encode perceptual structure without grounding? a case study in color. In *Proceedings of the 25th Conference on Computational Natural Language Learning (CoNLL),*, 2021.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Jayant Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego M Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. Do as i can, not as i say: Grounding language in robotic affordances. *ArXiv*, abs/2204.01691, 2022.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. *ArXiv*, abs/2204.14198, 2022.

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

Emily M. Bender and Alexander Koller. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.463.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. Experience Grounds Language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8718–8735, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.703.

Gemma Boleda. Distributional semantics and linguistic theory. *ArXiv*, abs/1905.01896, 2019.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

Angelo Cangelosi and Francesca Stramandinoli. A review of abstract concept learning in embodied agents and robots. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 373 (1752):20170131, June 2018. doi: 10.1098/rstb.2017.0131. Publisher: Royal Society.

Angelo Cangelosi, Giorgio Metta, Gerhard Sagerer, Stefano Nolfi, Chrystopher Nehaniv, Kerstin Fischer, Jun Tani, Tony Belpaeme, Giulio Sandini, Francesco Nori, et al. Integration of action and language knowledge: A roadmap for developmental robotics. *IEEE Transactions on Autonomous Mental Development*, 2(3):167–195, 2010.

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. In *International Conference on Learning Representations (ICLR)*, 2019.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311, 2022.

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

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben A. Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. In *CGW@IJCAI*, 2018.

Ishita Dasgupta, Christine Kaeser Chen, Kenneth Marino, Arun Ahuja, Sheila Babayan, Felix Hill, and Rob Fergus. Collaborating with language models for embodied reasoning. In *Advances in Neural Information Processing Systems (NeurIPS) LaReL workshop*, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

Linxi (Jim) Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *ArXiv*, abs/2206.08853, 2022.

Tarun Gupta, Peter Karkus, Tong Che, Danfei Xu, and Marco Pavone. Foundation models for semantic novelty in reinforcement learning. *ArXiv*, abs/2211.04878, 2022.

Steve Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

Zellig S. Harris. Distributional Structure. In Zellig S. Harris and Henry Hiż, editors, *Papers on Syntax*, Synthese Language Library, pages 3–22. Springer Netherlands, Dordrecht, 1981. ISBN 978-94-009-8467-7. doi: 10.1007/978-94-009-8467-7_1.

Ji He, Mari Ostendorf, Xiaodong He, Jianshu Chen, Jianfeng Gao, Lihong Li, and Li Deng. Deep Reinforcement Learning with a Combinatorial Action Space for Predicting Popular Reddit Threads. *arXiv:1606.03667 [cs]*, September 2016. URL `http://arxiv.org/abs/1606.03667`. arXiv: 1606.03667.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech M. Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. Grounded language learning in a simulated 3d world. *ArXiv*, abs/1706.06551, 2017.

Felix Hill, Sona Mokra, Nathaniel Wong, and Tim Harley. Human Instruction-Following with Deep Reinforcement Learning via Transfer-Learning from Text. *arXiv:2005.09382 [cs]*, May 2020. arXiv: 2005.09382.

Wenlong Huang, P. Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *ArXiv*, abs/2201.07207, 2022a.

Wenlong Huang, F. Xia, Ted Xiao, Harris Chan, Jacky Liang, Peter R. Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. *ArXiv*, abs/2207.05608, 2022b.

Peter A. Jansen. A Systematic Survey of Text Worlds as Embodied Natural Language Environments. *arXiv:2107.04132 [cs]*, July 2021. arXiv: 2107.04132.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi (Jim) Fan. Vima: General robot manipulation with multimodal prompts. *ArXiv*, abs/2210.03094, 2022.

Jared Kaplan, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling laws for neural language models. *ArXiv*, abs/2001.08361, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Shuang Li, Xavier Puig, Yilun Du, Clinton Jia Wang, Ekin Akyürek, Antonio Torralba, Jacob Andreas, and Igor Mordatch. Pre-trained language models for interactive decision-making. *ArXiv*, abs/2202.01771, 2022.

J. Liang, Wenlong Huang, F. Xia, Peng Xu, Karol Hausman, Brian Ichter, Peter R. Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *ArXiv*, abs/2209.07753, 2022.

Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. UNIFIED-IO: A Unified Model for Vision, Language, and Multi-modal Tasks. February 2023. URL `https://openreview.net/forum?id=E01k9048soZ`.

Yao Lu, Karol Hausman, Yevgen Chebotar, Mengyuan Yan, Eric Jang, Alexander Herzog, Ted Xiao, Alex Irpan, Mohi Khansari, Dmitry Kalashnikov, and Sergey Levine. Aw-opt: Learning robotic skills with imitation and reinforcement at scale. In *Conference on Robot Learning*, 2021.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob N. Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. In *International Joint Conference on Artificial Intelligence*, 2019.

Kyle Mahowald, Anna A. Ivanova, Idan Asher Blank, Nancy G. Kanwisher, Joshua B. Tenenbaum, and Evelina Fedorenko. Dissociating language and thought in large language models: a cognitive perspective. *ArXiv*, abs/2301.06627, 2023.

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

Dipendra Kumar Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *Conference on Empirical Methods in Natural Language Processing*, 2017.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022.

Roma Patel and Elizabeth-Jane Pavlick. Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*, 2022.

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. In *arXiv preprint arXiv:2103.00020v1*, 2021.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John F. J. Mellor, Irina Higgins, Antonia Creswell, Nathan McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, L. Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, N. K. Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Tobias Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew G. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem W. Ayoub, Jeff Stanway, L. L. Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv*, abs/2112.11446, 2021.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. *ArXiv*, abs/2210.01241, 2022.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *ArXiv*, abs/2102.12092, 2021.

Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *ArXiv*, abs/2201.12122, 2022.

Teven Le Scao, Angela Fan, Christopher Akiki, Elizabeth-Jane Pavlick, Suzana Ili'c, Daniel Hesslow, Roman Castagn'e, Alexandra Sasha Luccioni, Franccois Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Rose Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurenccon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa Etxabe, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris C. Emezue, Christopher Klamm, Colin Leong, Daniel Alexander van Strien, David Ifeoluwa Adelani, Dragomir R. Radev, Eduardo G. Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady ElSahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jorg Frohberg, Josephine L. Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro von Werra, Leon Weber, Long Phan, Loubna Ben Allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, Mar'ia Grandury, Mario vSavsko, Max Huang, Maximin Coavoux,

Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad Ali Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto L'opez, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, S. Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal V. Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Févry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiang Tang, Zheng Xin Yong, Zhiqing Sun, Shaked Brody, Y Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre Franccois Lavall'ee, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aur'elie N'ev'eol, Charles Lovering, Daniel H Garrette, Deepak R. Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, S. Osher Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdenvek Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ananda Santa Rosa Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Olusola Ajibade, Bharat Kumar Saxena, Carlos Muñoz Ferrandis, Danish Contractor, David M. Lansky, Davis David, Douwe Kiela, Duong Anh Nguyen, Edward Tan, Emily Baylor, Ezinwanne Ozoani, Fatim T Mirza, Frankline Ononiwu, Habib Rezanejad, H.A. Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jan Passmore, Joshua Seltzer, Julio Bonis Sanz, Karen Fort, Lívia Macedo Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, M. K. K. Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nourhan Fahmy, Olanrewaju Modupe Samuel, Ran An, R. P. Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas L. Wang, Sourav Roy, Sylvain Viguier, Thanh-Cong Le, Tobi Oyebade, Trieu Nguyen Hai Le, Yoyo Yang, Zachary Kyle Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Kumar Singh, Benjamin Beilharz, Bo Wang, Caio Matheus Fonseca de Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel Le'on Perin'an, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully A. Burns, Helena U. Vrabec, Iman I.B. Bello, Isha Dash, Ji Soo Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthi Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, María Andrea Castillo, Marianna Nezhurina, Mario Sanger, Matthias Samwald, Michael Cullan, Michael Weinberg, M Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patricia Haller, R. Chandrasekhar, R. Eisenberg, Robert Martin, Rodrigo L. Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Pratap Bharati, T. A. Laud, Th'eo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yashasvi Bajaj, Y. Venkatraman, Yifan Xu, Ying Xu, Yun chao Xu, Zhee Xao Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv*, abs/2211.05100, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *ArXiv*, abs/2010.03768, 2020.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan J. Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. *ArXiv*, abs/2009.01325, 2020.

Shiro Takagi. On the effect of pre-training for transformer in different modality on offline reinforcement learning. *ArXiv*, abs/2211.09817, 2022.

Serge Thill, Sebastia Padó n, and Tom Ziemke. On the importance of a rich embodiment in the grounding of concepts: Perspectives from embodied cognitive science and computational linguistics. *Topics in Cognitive Science*, 6(3):545–558, 2014. doi: https://doi.org/10.1111/tops.12093.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). *ArXiv*, abs/2206.10498, 2022.

Ruoyao Wang, Peter Alexander Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *ArXiv*, abs/2203.07540, 2022.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed Huai hsin Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *ArXiv*, abs/2206.07682, 2022.

Peratham Wiriyathammabhum, Douglas Summers-Stay, Cornelia Fermüller, and Yiannis Aloimonos. Computer vision and natural language processing: Recent approaches in multimedia and robotics. *ACM Comput. Surv.*, 49(4), dec 2016. ISSN 0360-0300. doi: 10.1145/3009906.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629, 2022.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel C. F. Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou, and Pengchuan Zhang. Florence: A new foundation model for computer vision. *ArXiv*, abs/2111.11432, 2021.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online Decision Transformer. In *Proceedings of the 39th International Conference on Machine Learning*, pages 27042–27059. PMLR, June 2022. URL `https://proceedings.mlr.press/v162/zheng22c.html`. ISSN: 2640-3498.

Appendices

This supplementary material provides additional results and discussion, as well as implementation details.

- Section A presents the BabyAI and BabyAI-Text environments.
- Section B, contains several additional results. We report the per-task success rate at the end of the training (B.1). We also analyze the influence of the observation's structure (i.e. either a symbolic image for the Symbolic-PPO agent or text for LLM based agents) in B.2. We then study the influence of pretraining in B.3 and conduct several ablation tests to understand the influence of the size of the LLM (B.4), the impact of the size of the action space (B.5.1), and the effect of the number of distractors (B.5.2). Eventually, we verify in B.6 the robustness of our method to domain-specific vocabulary.
- Section C is a qualitative analysis of GFlan-T5 during its training on the environment with a mix of tasks. We plot the evolution of the distribution of actions during training for 11 prompts.
- In Section D, we detail complementary tests for questions Q2 (D.2) and Q3 (D.3). We also analyze the functional grounding of temporal symbols "then" and "after" (D.4).
- Section E gives details related to the distributed experimental setup.
- Section F reports hyperparameters and implementation details used to finetune the models using PPO or Behavioral Cloning.
- In Section G, we detail how the confidence intervals given in Figure 4 and Appendix D are obtained.
- In Section H, we give the word substitutions used in our generalization experiments from sections 4.2 and 4.3.

A Environments

We extend the BabyAI platform [Chevalier-Boisvert et al., 2019] and create a text-only version named BabyAI-Text that encapsulates BabyAI and returns linguistic observations. Figure 5 explains our environment.

A.1 BabyAI

BabyAI Chevalier-Boisvert et al. [2019] is a language-conditioned environment where the agent has a limited number of steps to complete a language goal. This platform relies on a gridworld environment (MiniGrid) to generate a set of complex instructions-following environments. It has been specifically designed for research on grounded language learning and related sample efficiency problems. The gridworld environment is populated with the agent and objects (of 6 possible colors): boxes, balls, doors, and keys. These entities are placed in rooms of 8×8 tiles that are connected by doors that can be locked or closed. The grid is procedurally generated (i.e. objects populating an episode are randomly chosen and their position, as well as the agent's position, are also random). Some of the objects are useful for the task to achieve, while others are considered as distractors (objects can't be crossed, the agent has to either bypass them or move them). The agent can do 6 primitive actions: `turn left`, `turn right`, `go forward`, `toggle`, `pick up` to solve the language instruction (for instance `Pick up the red box`). To observe its environment, the agent has access to a partial view (i.e. it only sees the objects that belong to the 6×6 grid in front of it). BabyAI proposed to access this partial view through a symbolic mapping that returns 3 matrices of size 6×6. The first matrix contains which object is in the observed cells, the second gives the color of these objects, and the last one their state (e.g. locked, open). When the agent completes the task after N steps, it receives the reward $r_N = 1 - 0.9\frac{N}{H}$, where H is the maximum number of steps. During training, we multiply all rewards by 20 to ensure a good propagation of the rewards as per [Mirchandani et al., 2021]. If the agent has not completed the task in the current step, the reward is 0. Additionally, BabyAI also provides visualization tools for experimenters to observe the grid and better grasp agents' behaviors.

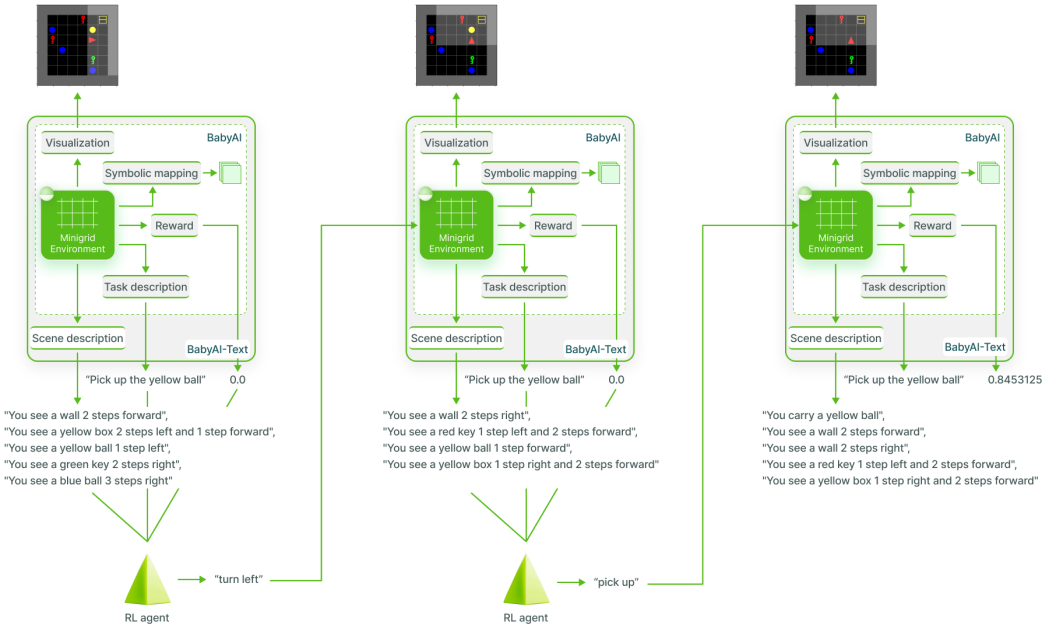


Figure 5: An illustration of how our BabyAI-Text environment encapsulates BabyAI. We keep the inner minigrid environment as well as task descriptions and reward but map the partial view of the agent to a text description.

A.2 BabyAI-Text

BabyAI-Text is a textual environment that encapsulates BabyAI and provides a description of each observation instead of a symbolic representation. A textual description consists of a list of template descriptions with the following structure:

- "You see a <*object*> <*location*>" if the object is a key, a ball, a box or a wall.
- "You see a(n) *open/closed* door <*location*>" , if the agent sees a door.
- "You carry a <*object*>", if the agent carries an object.

The <*object*>, is composed of an *adjective* (among 6 possible colours: red, blue, green, yellow, grey, purple) and a noun (among 4 possible: key, door, box, ball). The <*location*> is given as the number of steps *right*, *left*, and or *forward* from the agent to the object. We illustrate this in the leftmost observation of Figure 5 where the "yellow box" is "2 steps left and 1 step forward" from the agent (the red triangle). Thus an object described as "1 step forward" is right in front of the agent that does not need to *go forward* if it wants to pick that object. Walls of the room are the only spatially extended objects in BabyAI-Text. We give their location at the closest distance to the agent. See the leftmost image of Figure 5 for an example where the agent sees a wall "2 steps forward" and another wall "2 steps left". All of the choices for describing the environment constitute what we call the geometry of the environment, that the agent has to ground in order to succeed in the task. The presence of a fine grained geometry (with distances in steps to the different object in the room) is one of the main differences from other textual games such as TextWorld or ScienceWorld where all objects in a room are not spatially described.

Thanks to this extension, BabyAI-Text resembles a TextWorld (i.e. provides text descriptions of the observation and executes text commands) while keeping the inner minigrid environment along with BabyAI's tasks and visualization tools. Moreover, as our extension simply provides an alternative mapping of observations, one can both use and compare agents that either expect text-only observations (with BabyAI-Text) or symbolic observations (with BabyAI).

B Additional results

B.1 Per-task success rate

In order to get a better understanding of our agent's capabilities, we report in Figure 6 the success rate on each task from our "no-change" evaluation in Figure 4 (assessing the post-training performance of agents on 1000 test episodes of our mixed setup) of Flan-T5 and GFlan-T5. These results (with 4 seeds after 1.5 million training steps) show that our functional grounding leads GFlan-T5 to master the *GoTo* and *PickUp* tasks while improving results on *PutNextTo* and *PickupThen/AfterPickup*. However, GFlan-T5 has not found yet any robust strategy for the *OpenDoor* task (being the hardest as the agent must find the right key and discover that the action "toggle" opens the door) in the relatively short allocated time.

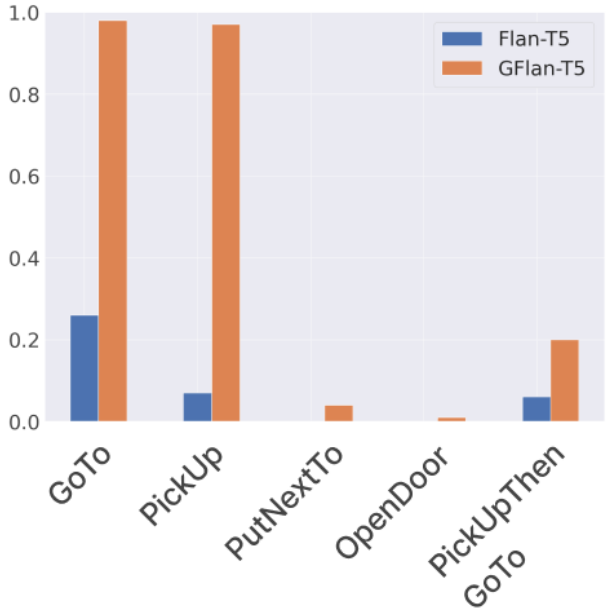


Figure 6: Per-task success rate for the 1000 evaluation trajectories performed in Figure 4.

B.2 Textual vs symbolic representation

In order to understand how the structure of the observation (i.e. either symbolic image using 3 matrices containing integers defining respectively the object seen, its color and property if any or text) influences the success rate of an RL agent, we compare the DRRN and Symbolic-PPO respectively trained on BabyAI-Text and BabyAI on the `Go To Red Ball` task. In this task, the agent has to go in front of a red ball in 1 room without any distractor (i.e. the task never changes, only the position of agent and red ball do). The task has been voluntarily chosen as trivial so that the main difference only comes from the way the information is given to the agent. Both the DRRN and Symbolic-PPO agents have a similar number of parameters (1M), they both use recurrent layers to deal with partial observability and use the canonical action space.

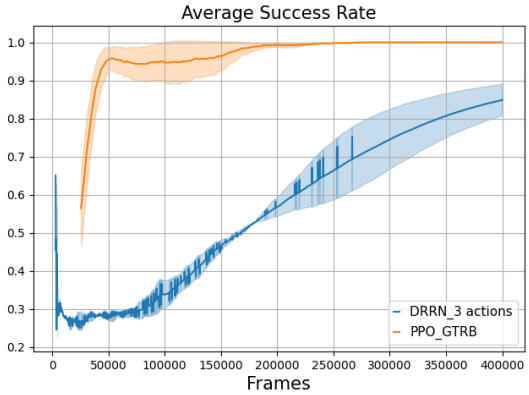


Figure 7: Average success rate for DRRN and Symbolic-PPO on the *Go To Red Ball* task with standard deviation over two random seeds. The PPO receives symbolic information and the DRRN gets textual observations.

Contrary to what one might assume in Figure 7 the PPO agent converges faster than the DRRN agent on this trivial task. Thus, symbolic observations make the learning easier for the agent. We conclude that even if language contains high-level information, understanding the link between spatial information and language is far more difficult than using symbolic information given in a matrix. Indeed, the matrix already contains a geometric bias favorable to the agent. We also want to point out that the DRRN is an off-policy RL method compared to PPO (which is on-policy) and that consequently, the DRRN was expected to be, by-design, more sample efficient.

B.3 Impact of pretraining

We test how pretraining structured our LLM allowing for efficient finetuning. We vary which weights of Flan-T5 are kept pretrained as well as how we compute actions' probability (i.e. either using our method reusing language modeling heads or using new action heads with an MLP). We evaluate the performance of 5 models:

• The full LLM is pretrained and language modeling heads are used for actions probability: GFlan-T5 (Figure 8)

• The full LLM is pretrained and new action heads are used: AFlan-T5 (Figure 9)

• Only the embedding layer's weights are kept pretrained (the rest of the LLM is randomly initialized) and new action heads are used: NPAE-Flan-T5 (Figure 10)

• Only the embedding layer's weights are kept pretrained (the rest of the LLM is randomly initialized) and the (randomly initialized) language modeling heads are used for actions' probability: **NPE-FlanT5** (Figure 11)

• All LLM's weights are randomly initialized and action heads are used: **NPA-Flan-T5** (Figure 12)

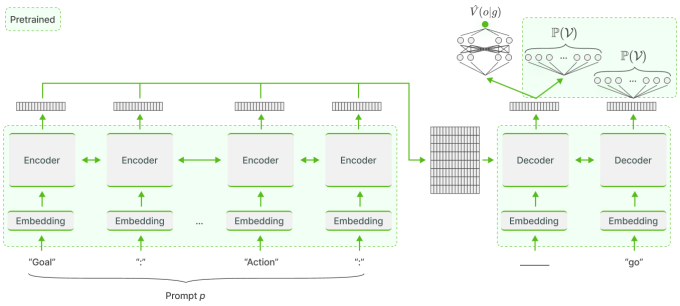


Figure 8: GFlan-T5: We use the Flan-T5 architecture and add a value head. We initialize the agent with the pretrained weights (framed in green in the diagram) including its language modeling heads to compute action probabilities. The weights of the value head are initialized randomly. GFlan-T5 stands for: grounded Flan-T5.

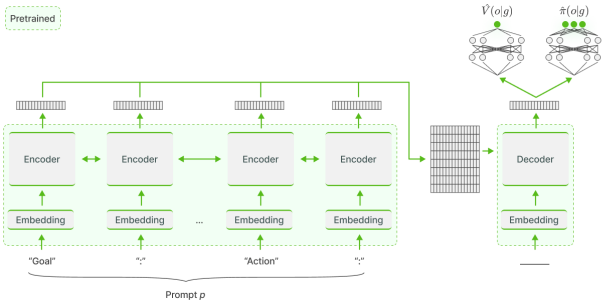


Figure 9: AFlan-T5: We use the Flan-T5 architecture but replace the language modeling heads with action heads (that return the probability for each action) and add a value head. We initialize the embedding, the encoder and decoder parts of the agent with the pretrained weights (framed in green in the diagram) and the other weights randomly. AFlan-T5 stands for action heads Flan-T5.

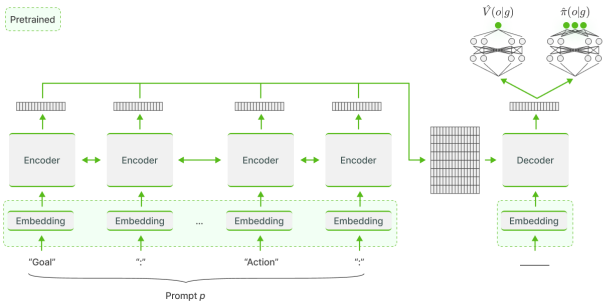


Figure 10: NPAE-Flan-T5: We use the Flan-T5 architecture but replace the language modeling heads by action heads and add a value head. We initialize the embedding with the pretrained weights (framed in green in the diagram) and the other weights randomly. NPAE-Flan-T5 stands for: non-pretrained with action heads and pretrained embedding Flan-T5.

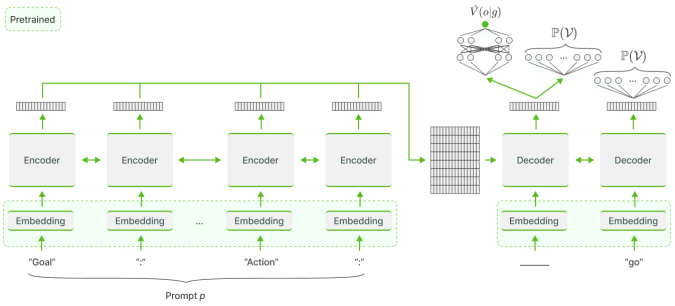


Figure 11: NPE-Flan-T5: We use the Flan-T5 architecture and add a value head. We initialize the embedding with the pretrained weights (framed in green in the diagram) and the other weights randomly. NPE-Flan-T5 stands for: non-pretrained with pretrained embedding Flan-T5.

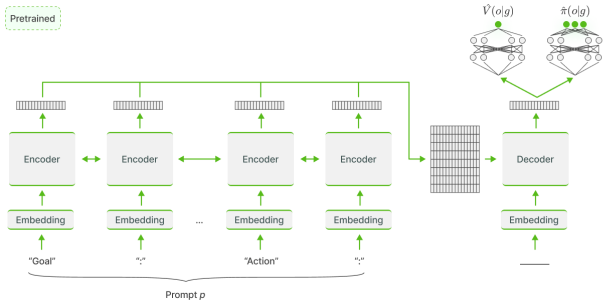


Figure 12: NPA-Flan-T5: We use the Flan-T5 architecture but replace the language modeling heads with action heads and add a value head. We initialize all the weights randomly. NPA-Flan-T5 stands for: non-pretrained with action heads Flan-T5.

Figure 13 compares the training curves of the agents above on the task *Go To <object>*. GFlan-T5 has unsurprisingly the best results as it is fully pretrained. More surprisingly, AFlan-T5 takes more steps than expected to perform better than the non-pretrained networks (250000 frames). We hypothesize that during the pertaining, the last transformer layer encodes information in a space designed for language modeling heads ($\approx$ 32000 heads) which is not convenient for the non-pretrained 6 actions heads. Indeed, AFlan-T5 has to make sense of this space before getting the benefits of having the rest of the network trained. This could explain why it suddenly performs better after 250000 steps. Comparing NPAE, NPA and NPE Flan-T5, we see that the presence of an action head is crucial for non-pretrained networks. Indeed, the NPE fails to learn in the given number of steps compared to NPAE and NPA that have similar learning curves. A possible explanation is that for NPE, the information flow that is backpropagated through the gradient is really small due to the huge number of language modeling heads and the few number of tokens updated (< 100). On the opposite, GFlan-T5, that also uses language modeling heads but is fully pretrained, only needs a light finetuning for the necessary tokens explaining its high success rate and sample efficiency.

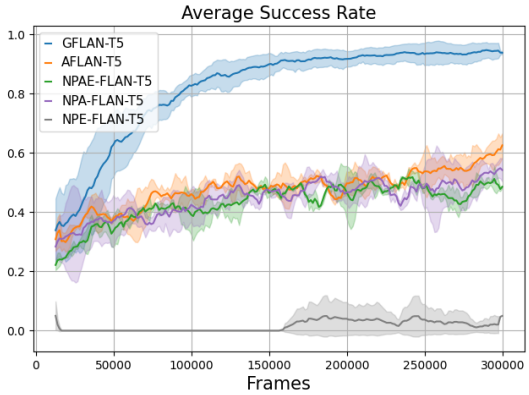


Figure 13: Average success rate of varying pretrained weights and scoring method with standard deviation over two random seeds. We train all LLMs on the *Go to <object>* task in 1 room, with 8 distractors, the 6 canonical actions and using Flan-T5 large (780 million parameters) as architecture.

B.4 Impact of the size of the LLM

The capacities of LLMs depend strongly on their size [Kaplan et al., 2020] and many properties of these networks only appear when they are large enough [Wei et al., 2022]. We consequently test the influence of the size of the LLM on our results by training 3 different GFlan-T5 (as well as the DRRN and Symbolic-PPO baselines) on the *Go to <object>* task for 400.000 steps: GFlan-T5 **small** (80 million parameters), GFlan-T5 **large** (780 million parameters) and GFlan-T5 **XL** (3 billion parameters).

We show the evolution of average success rate over 2 seeds in Figure 14 highlighting that pretraining prior knowledge only looks impactful when the network is large enough. The difference between the learning properties of small and large models relates to the definition of an emergent behavior given by Wei et al. [2022]: *"an ability is emergent if it is not present in smaller models but is present in larger models"*. Beyond the data on which a model has been trained, the size of this model seems crucial for the acquisition of new knowledge about relations between entities during the finetuning phase.

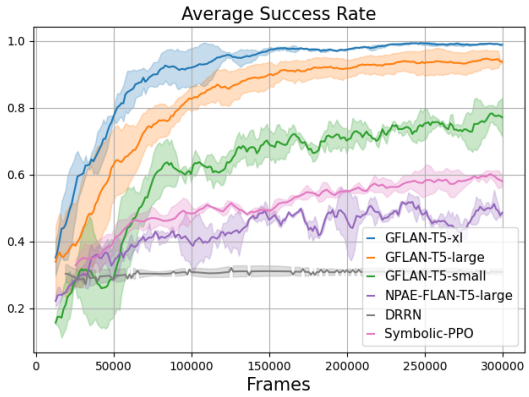


Figure 14: Impact of the size of the LLM on online RL finetuning. We conduct the tests with the *Go to <object>* task in 1 room, with 8 distractors. We measure the evolution of average success rate over 2 seeds with standard deviation for GFlan-T5 **small** (80 million parameters), GFlan-T5 **large** (780 million parameters) and GFlan-T5 **XL** (3 billion parameters). DRRN, NPAE-Flan-T5-large and Symbolic-PPO are given as baselines.

B.5 Impact of varying action space and distractors

In this section, we detail the studies about the impact of varying the action space and the number of distractors.

B.5.1 Impact of the dimension of the action space

One of the expected advantages of pretrained LLMs in RL is that they avoid the *Tabula-Rasa* paradigm and already have useful biases. In these experiments, we test the sensibility of LLMs to the size of the action space by using 3 different action spaces (**restricted**, **canonical**, **augmented**) when trained on the *Go to <object>* task.

We conduct the tests in an environment with 1 room, 8 distractors and in Figure 15 report full learning curves used to draw Figure 3a. We show that GFlan-T5 efficiently handles the different action spaces compared to the other agents. Its initial biases are particularly helpful when the action space is large. Indeed, when we look at the difference of sucess rate between GFlan-T5 and the second best-performing agent after the 50000 first steps, there is almost no difference in the restricted settings and 0.35 in the augmented settings. That supports the hypothesis that the results are due to LLMs' ability to discard useless action quickly at the beginning of finetuning.

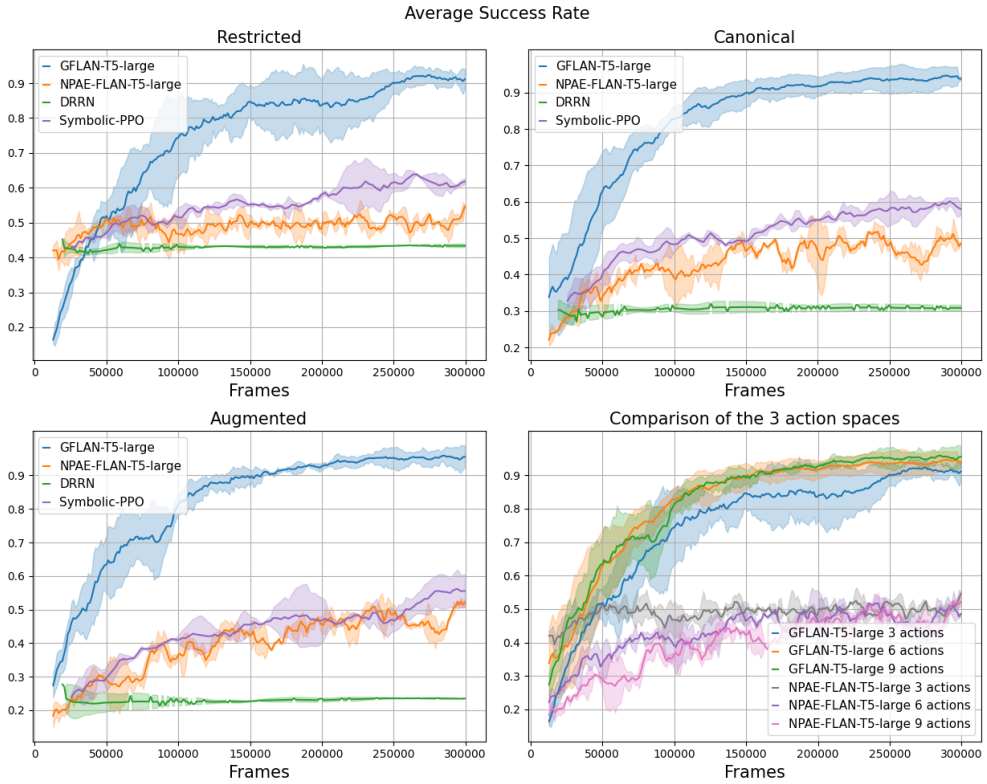


Figure 15: Learning curves for the agents on the *Go To* task for different sizes of action space (Restricted: 3 actions, Canonical: 6, Augmented: 9, with only the 3 actions that are useful). The success rate is given over 2 seeds along with standard deviation.

B.5.2 Impact of the number of distractors

In Figure 3b we have shown that LLMs are less sensitive to variations on task complexity by plotting the evolution of sample efficiency (Equation (3)) for different numbers of distractors: 4, 8 and 16. In Figure 16 we report the full learning curves. We observe that Symbolic-PPO's performances collapse when we go from 4 to 16 distractors whereas GFlan-T5's performances remain similar.

NPAE-Flan-T5's performances are also non-affected by the change in the number of distractors but in this case we suppose it is because it cannot learn the task in 400000 steps.

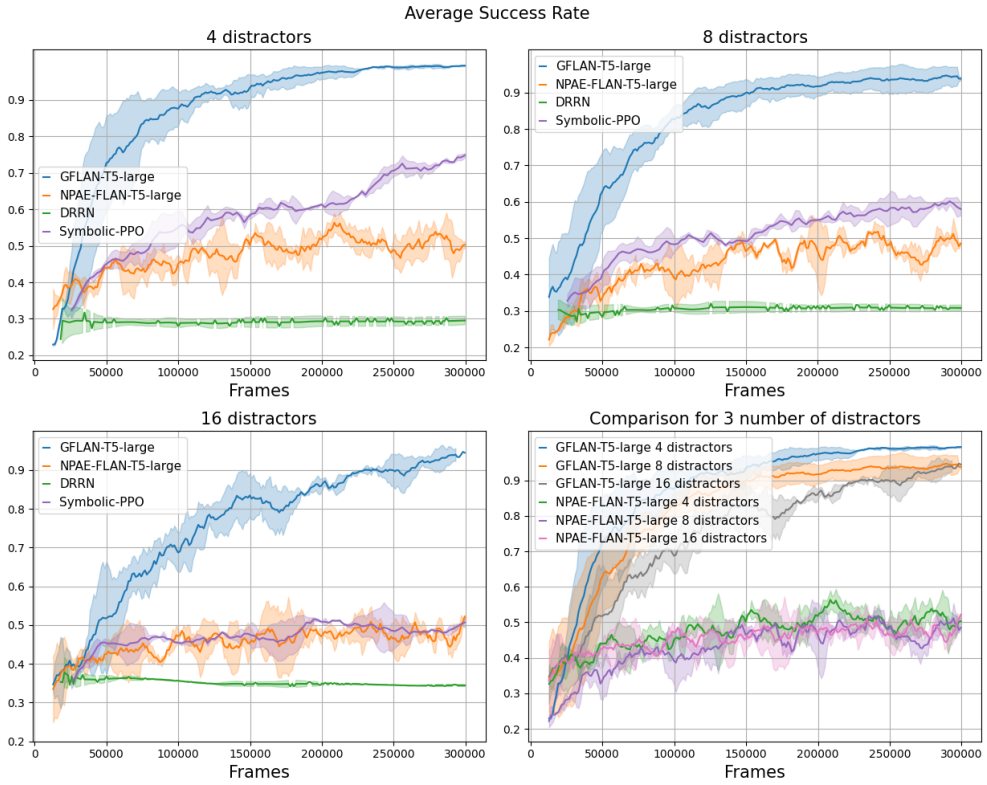


Figure 16: Learning curves for the agents on the *Go To* task for different number of distractors (4, 8, 16). The success rate is given over 2 seeds with standard deviation.

B.6 Robustness to domain-specific vocabulary

In Section 4.1, we have shown the robustness of our method to random vocabulary for words that do not influence the grounding of actions (in our case, the objects and their colors). Nonetheless, one can imagine an environment with a specific vocabulary where common words are used to describe particular technical terms with possibly very different meanings. To verify the impact of such environment on our training process, we trained GFlan-T5 on the *GoTo* task where the actions "turn left" and "turn right" are flipped (i.e. using "turn left" makes the agent rotate to the right, and the opposite for "turn right"). Figure 17 shows that, while the prior knowledge of the LLM leads to poorer performance at the very beginning of training (as the LLM must learn to rotate left and right), GFlan-T5 converges at a similar speed than in the non-flipped environment. This result hints robustness that our grounding method for LLMs can adapt to domain-specific vocabularies.

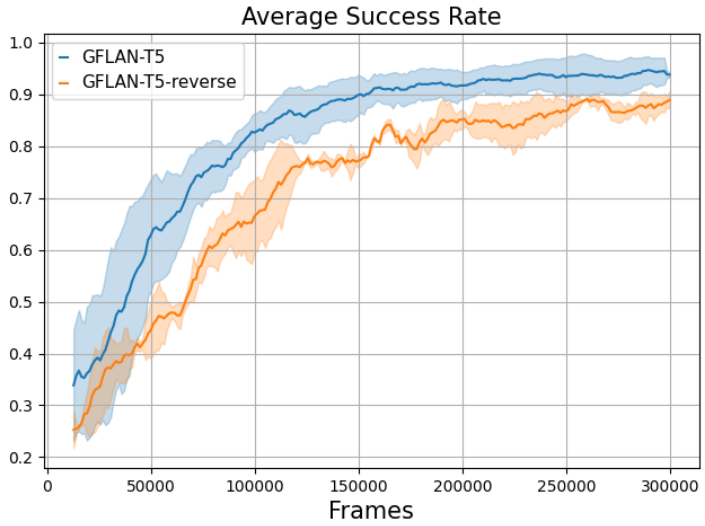


Figure 17: Comparison of the average success rate over training for GFlan-T5 on the *Go To* when the actions “turn left” and “turn right” are flipped ("-reverse"). The success rate is given over 2 seeds with standard deviation.

C Evolution of actions distribution on evaluation prompts

To better grasp the skill acquisition dynamics when performing online RL grounding on GFlan-T5 in the multi-task setting of Section 4.1, we test at each update the LLM on 11 prompts listed in Table 2. We plot in Figure 18 the evolution of action probabilities outputted by our LLM aiming to partially decipher the changes in the LLM and visualize which skill is acquired when.

Prompts 0 and 1 are simple navigation tasks. The agent has to move in the direction given in the prompt. Looking at the corresponding plots we observe two things: first, the optimal behavior is learned in less than a hundred of updates, even for prompt 1 for which the bias at the beginning is both wrong and high. Second, from the beginning, only the navigation actions (`turn left`, `turn right`, `go forward`) relevant for the *Go To <object>* task have a high probability. Therefore, the Flan-T5 780M seems to already have useful biases for navigation and is able to quickly update or correct them through interactions.

We observe similar useful biases with the *Pick Up <object>* task (using prompts 5 and 6). Indeed, at the beginning, both the `pick up` action and navigation actions already have a high probability.

We can see that the agent struggles to ground the geometry of the environment with prompts 6 and 7. Indeed, it has to understand that an object described at "1 step forward" is in front of it such that it can pick it up or drop it directly without moving further. While GFlan-T5 eventually seems to understand it, it still shows some hesitation as proven by the fact that it gives almost the same probability to `go forward` and `pick up` or `drop` for the prompt 7 at the end of training (see Figure 18).

We also verify how GFlan-T5 understands temporal constructions such as doing an action A then an action B (prompt 8) or doing an action A after doing an action B (prompt 9). These two test prompts are exactly the same except for the goal where prompt 8 uses "then" and prompt 9 uses "after" to link the two actions. We observe that when the order of actions in the task specification is the same as the one the agent has to do (i.e. prompt 8), the LLM quickly and learns to choose the right action even if during the learning it loses its ability (with `turn left` and `turn right` that are almost at the same probability. However, when the order of actions mentioned in the goal specification is reversed (prompt 9), the LLM ends up favoring the wrong direction and exhibits much more hesitation from the begining of the training. This qualitative observation concurs with the measure of success rate given in Appendix D.4.

The prompts 2, 3 and 4 show that the agent has difficulties with the task *Open <door>*. This task is fairly complex since the agent has to infer that a key of the same color as the closed door is required to open it. In the given training budget, the agent fails to associate the need of a key with the task.

Finally we test the agent on a task that is not seen during training. It is the generalization task *Pick up <object A> then/after Pick up <object B>* from Q3 Section 4.3, composed from two tasks seen during training *Pick up* and *Pick up then Go To* (prompt 10). The prompt is built such that the agent has accomplished half of the instruction and has to drop the object it carries in order to pick another one. The action `drop` is the optimal one because it is the only one that allows the agent to complete the goal in a minimum number of steps. Between the updates 400 and 600 the agent begins to increase the probability of the `drop` action. This change is correlated to the change of distribution in prompt 7. It can be interpreted as the fact that the action `drop` begins to be grounded after 800 updates.

Table 2: Test prompts. The prompts' header (*Possible action of the agent: turn left, turn right, go forward, pick up, drop, toggle*) is not shown below as it remains the same for all prompts

Ids	Tasks	Prompts	Comments
0	Go To <object>	**Goal of the agent**: go to the green ball **Observation 0**: You see a wall 2 step left, You see a purple key 1 step left and 2 steps forward, You see a yellow key 1 step left and 1 step forward, You see a green ball 3 steps forward, You see a grey ball 1 step right and 5 steps forward, You see a green key 1 step right and 2 steps forward, You see a grey ball 1 step right and 1 step forward, You see a green key 2 steps right and 4 steps forward, You see a red box 2 steps right and 2 steps forward, **Action 0**: **Expected answer: go forward**	Simple navigation task.
1	Go To <object>	**Goal of the agent**: go to the green ball **Observation 0**: You see a wall 2 step left, You see a purple key 1 step left and 2 steps forward, You see a yellow key 1 step left and 1 step forward, You see a green ball 3 steps forward, You see a grey ball 1 step right and 5 steps forward, You see a green key 1 step right and 2 steps forward, You see a grey ball 1 step right and 1 step forward, You see a green key 2 steps right and 4 steps forward, You see a red box 2 steps right and 2 steps forward, **Action 0**: go forward **Observation 1**: You see a purple key 1 step left and 1 step forward, You see a yellow key 1 step left, You see a green ball 2 steps forward, You see a grey ball 1 step right and 4 steps forward, You see a green key 1 step right and 1 step forward, You see a grey ball 1 step right, You see a green key 2 steps right and 3 steps forward, You see a red box 2 steps right and 1 step forward, **Action 1**: turn right **Observation 2**: You see a wall 2 step right, You see a green key 3 steps left and 2 steps forward, You see a green ball 2 steps left, You see a red box 1 step left and 2 steps forward, You see a green key 1 step left and 1 step forward, You see a grey ball 1 step forward, **Action 2**: **Expected answer: turn left**	Simple navigation task.
2	Open <adj> door	**Goal of the agent**: open the purple door **Observation 0**: You see a wall 3 steps forward, You see a wall 3 steps left, You see a yellow key 1 step right and 1 step forward, You see a locked purple door 2 steps right and 3 steps forward, You see a purple ball 3 steps right and 1 step forward, You see a green box 3 steps right, You see a purple key 2 steps left, **Action 0**: **Expected answer: turn left**	Inference task The agent has to infer that a key of the same color is needed and moves toward it.
3	Open <adj> door	**Goal of the agent**: open the purple door **Observation 0**: You see a wall 3 steps forward, You see a wall 3 steps left, You see a yellow key 1 step right and 1 step forward, You see a locked purple door 2 steps right and 3 steps forward, You see a purple ball 3 steps right and 1 step forward, You see a green box 3 steps right, You see a purple key 2 steps left, **Action 0**: turn left **Observation 1**: You see a wall 3 steps forward, You see a wall 3 steps right, You see a purple key 2 steps forward, **Action 1**: go forward **Observation 2**: You see a wall 2 steps forward, You see a wall 3 steps right, You see a purple key 1 step forward, **Action 2**:	Inference task The agent has to infer that a key of the same color is needed and pick it up.

Continued on next page

Table 2 – continued from previous page

Id	Task	Prompt	Comments
		Expected answer: pick up	
4	Open <adj> door	**Goal of the agent**: open the purple door **Observation 0**: You carry a purple key, You see a wall 3 steps forward, You see a wall 5 steps left, You see a yellow key 1 step left and 1 step forward, You see a locked purple door 3 steps forward, You see a purple ball 1 step right and 1 step forward, You see a green box 1 step right, **Action 0**: go forward **Observation 1**: You carry a purple key, You see a wall 2 steps forward, You see a wall 5 steps left, You see a yellow key 1 step left, You see a locked purple door 2 steps forward, You see a purple ball 1 step right, **Action 1**: go forward **Observation 2**: You carry a purple key, You see a wall 1 step forward, You see a wall 5 steps left, You see a locked purple door 1 step forward, **Action 2**: **Expected answer: toggle**	Inference task The agent has to infer that you can open a closed door by toggling it while having a key of the same color.
5	Pick up <object>	**Goal of the agent**: pick up green box **Observation 0**: You see a wall 2 steps forward, You see a wall 2 steps left, You see a yellow ball 1 step left and 1 step forward, You see a green box 2 steps right, **Action 0**: **Expected answer: turn right**	The agent has to reuse knowledge from navigation task.
6	Pick up <object>	**Goal of the agent**: pick up green box **Observation 0**: You see a wall 2 steps forward, You see a wall 2 steps left, You see a yellow ball 1 step left and 1 step forward, You see a green box 2 steps right, **Action 0**: turn right **Observation 1**: You see a wall 2 steps left, You see a blue key 1 step right, You see a red ball 2 steps right and 1 step forward, You see a green box 2 steps forward, **Action 1**: go forward **Observation 2**: You see a wall 2 steps left, You see a red ball 2 steps right, You see a green box 1 step forward, **Action 2**: **Expected answer: pick up**	The agent has to reuse knowledge from navigation task. and understand the geometry of the room.
7	Put <object A> next to <object B>	**Goal of the agent**: put blue ball next to red box **Observation 0**: You carry a blue ball, You see a wall 5 steps forward, You see a wall 2 steps left, You see a grey key 1 step right and 2 steps forward, You see a red box 3 steps forward, **Action 0**: go forward **Observation 1**: You carry a blue ball, You see a wall 4 steps forward, You see a wall 2 steps left, You see a grey key 1 step right and 1 step forward, You see a red box 2 steps forward, **Action 1**: **Expected answer: drop**	The agent has to reuse knowledge from navigation and understand the geometry of the room.
8	Pick up <object A> then go to <object B>	**Goal of the agent**: pick up the blue ball then go to the red box **Observation 0**: You see a wall 3 steps forward, You see a wall 4 steps right, You see a purple key 2 steps forward, You see a red box 2 steps right, You see a blue ball 2 steps left, **Action 0**: **Expected answer: turn left**	Prompt 8 and 9 test the ability of the agent to understand temporal concepts.
9	Go to <object B> after you pick up <object A>	**Goal of the agent**: go to the red box after you pick up the blue ball **Observation 0**: You see a wall 3 steps forward, You see a wall 4 steps right, You see a purple key 2 steps forward, You see a red box 2 steps right, You see a blue ball 2 steps left, **Action 0**:	Same as prompt 8 but sentence action order different from execution order.

Continued on next page

Table 2 – continued from previous page

Id	Task	Prompt	Comments
		Expected answer: turn left	
10	Pick up <object A> then pick up <object B>	**Goal of the agent**: pick up the green key then pick up the red box **Observation 0**: You carry a green key, You see a wall 4 steps forward, You see a wall 4 steps left, You see a red box 1 step left, You see a purple ball 2 steps left and 1 step forward, **Action 0**: **Expected answer: drop**	Task never seen in training to analyze generalization.

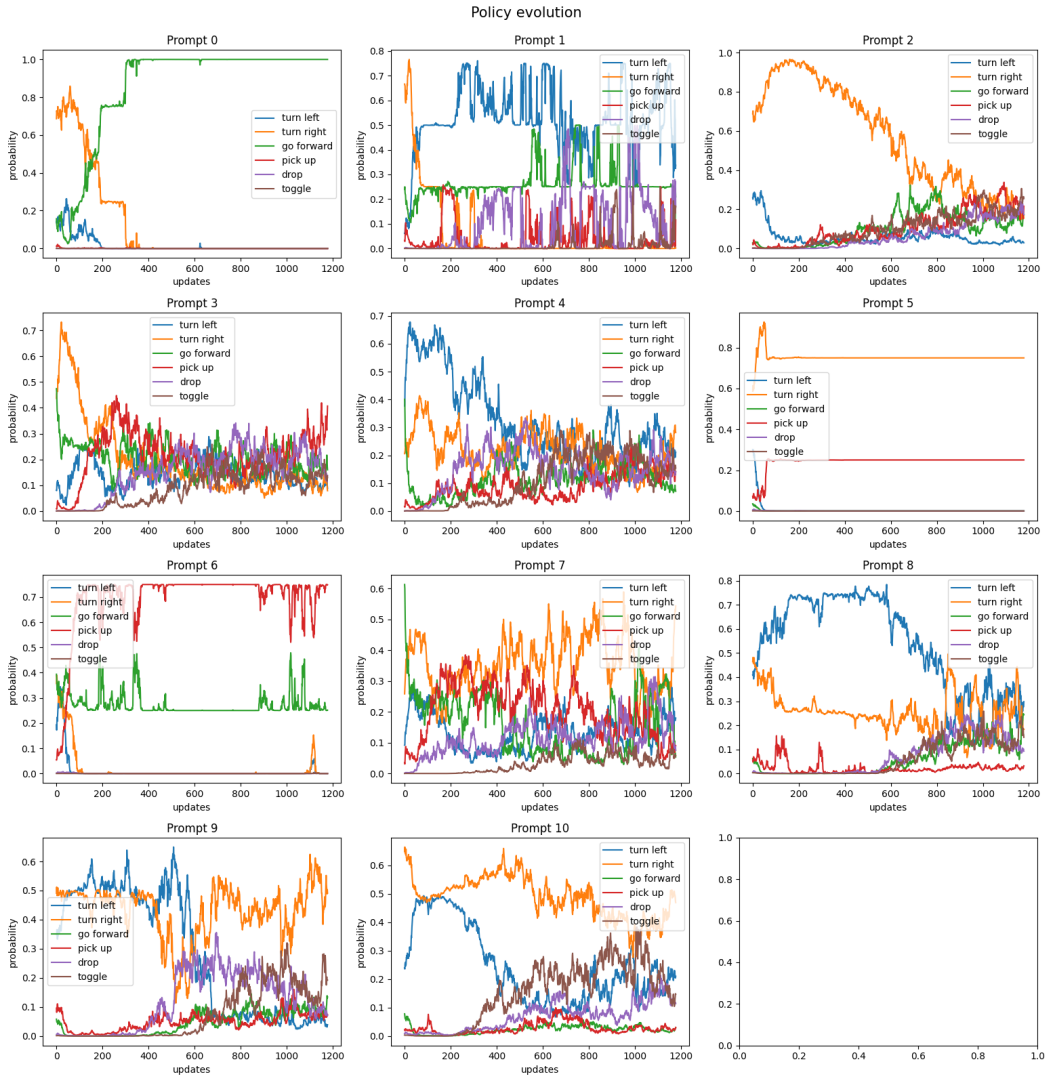


Figure 18: Evolution of actions' probability over training for test prompts listed in Table 2.

D Generalization tests details

D.1 Recapitulating results table

In this section we summarize the numerical results shown in Figure 4 with the confidence intervals calculated as explained in Appendix G.

Table 3: Generalization tests

	Environments	GFlan-T5	Flan-T5	NPAE	DRRN	Random
Q2	Mix - no change	0.89 ± 0.05	0.11 ± 0.03	0.17 ± 0.04	0.14 ± 0.02	0.15 ± 0.05
	Mix - out-of-vocabulary nouns	0.87 ± 0.05	0.09 ± 0.02	0.16 ± 0.05	0.15 ± 0.00	
	Mix - invented nouns and adjectives	0.88 ± 0.06	0.11 ± 0.03	0.16 ± 0.03	0.16 ± 0.00	
Q3	Pick up then/after pick up	0.12 ± 0.06	0.02 ± 0.00	0.06 ± 0.01	0.06 ± 0.03	0.05 ± 0.05
	Mix - synonym actions	0.12 ± 0.12	0.02 ± 0.00	0.16 ± 0.04	0.17 ± 0.04	0.15 ± 0.05
	Go To - English	0.99 ± 0.01	0.27 ± 0.03	0.31 ± 0.04	0.31 ± 0.03	0.30 ± 0.05
	Go To - French	0.02 ± 0.01	0.03 ± 0.00	0.30 ± 0.02	0.31 ± 0.02	

D.2 Complementary tests for Q2

Table 4: Complementary tests for Q2

Environments	GFlan-T5	Flan-T5	NPAE	DRRN	Random
Mix - no change	0.89 ± 0.05	0.11 ± 0.03	0.17 ± 0.02	0.14 ± 0.02	0.15 ± 0.05
Mix - unseen in-vocabulary objects	0.87 ± 0.03	0.12 ± 0.03	0.16 ± 0.08	0.17 ± 0.09	
Mix - out-of-vocabulary adjectives	0.87 ± 0.07	0.16 ± 0.03	0.16 ± 0.02	0.16 ± 0.01	

To further analyze results from in Section 4.2, we conduct more systematic tests on different aspects of the generalization to new words. The results are given in Table 4.

Unseen in-vocabulary objects During training we remove tasks whose goal contain the following objects: yellow box, red key, red door, green ball and, grey door. Nonetheless, the agent can have these objects as distractors and so have seen them during training. We assess how our agents perform on the mix of tasks with goals using only these objects. The success rate of 0.87 points out that GFlab-T5 is unaffected by the use of unseen in-vocabulary objects.

Unseen out-of-vocabulary adjectives We perform the same test as for out-of-vocabulary nouns in Section 4.2 but this time with adjectives that do not belong to the BabyAI-Text vocabulary. We generate the prompt by exchanging the adjectives with predefined synonyms (see Table 9). Similarly to the test with out-of-vocabulary nouns, the test with out-of-vocabulary adjectives reveals that GFlan-T5 is unaffected by this change. Indeed, the success rate is of 0.87 compared to the one of mix of tasks without change at 0.89.

D.3 Complementary tests for Q3

In Section 4.3, we observe that GFlan-T5 fails to generalize to an environment where we change the language. We hypothesize that such a change modifies too many grounded symbols at once. To verify this hypothesis, we test a middle-ground version, where we keep the environment in English

Table 5: Complementary tests for Q3

Environments	GFlan-T5	Flan-T5	NPAE	DRRN	Random
Go To - English	0.99 ± 0.01	0.27 ± 0.03	0.31 ± 0.04	0.31 ± 0.03	0.30 ± 0.05
Go To - French	0.02 ± 0.01	0.03 ± 0.00	0.30 ± 0.02	0.31 ± 0.02	
Go To - English with actions in French	0.15 ± 0.04	0.26 ± 0.02	0.31 ± 0.01	0.33 ± 0.00	

but actions are in French. In this setting, Table 5 shows that the success rate of the agent (0.15) is better than the fully french environment (0.02). This observation supports that finetuned agents tend to generalize to related words in other languages. Nonetheless, this ability seems highly dependent on the number of grounded words we modify.

D.4 LLM grounding of temporal symbols: "then" and "after"

In this experiment we observe the dynamics of functional grounding of instructions containing the temporal symbols "then" and "after" using the tasks: *Pick up <object A> then go to <object B>* and *Go to <object B> after pick up <object A>*. As the order of the action matters to have the task considered completed, a correct grounding of these symbols is crucial. Table 6 shows that GFlan-T5 has a better grounding of these words than the original Flan-T5 agent. Moreover, we observe a slight bias after finetuning: the agent has stronger performances for the tasks with "then" (success rate of 0.22) compared to the tasks with "after" (success rate of 0.17). We hypothesize it is easier to ground the word "then" because the order of the actions the agent must do is the same as the order in which the actions appear in the instructions. A qualitative example of this behavior is given in Appendix C (prompts 8, 9).

Table 6: Test on tasks with temporal components

Environments	GFlan-T5	Flan-T5	NPAE	DRRN	Random
Mix of tasks then/after	0.23 ± 0.06	0.12 ± 0.01	0.09 ± 0.01	0.09 ± 0.02	0.04 ± 0.05
Tasks with then only	0.22 ± 0.11	0.12 ± 0.01	0.10 ± 0.003	0.10 ± 0.02	
Tasks with after only	0.17 ± 0.05	0.13 ± 0.05	0.10 ± 0.03	0.10 ± 0.01	

E Distributed experimental setup

In order to accelerate our online RL finetuning, we first leverage a classic distributed data collection setup where 32 BabyAI-Text environments are running in parallel (all on CPUs). Our environments are run in a synchronous way, meaning that at every step, we get 32 current states and need to send 32 actions back to the environment. In very classic RL setups, policy networks are usually small and we simply batch the 32 states, feed them to the network and obtain the 32 actions' probability before sampling from them and choosing one action per environment. However, as explained in Section 3.2, our method requires $|\mathcal{A}|$ forward passes on a potentially very large and computationally expensive LLM in order to compute actions' probability for a single environment. Hence, we now need $32 \times |\mathcal{A}|$ forward passes for a single step in all environments, which can easily become a huge bottleneck in our training process.

To overcome this, we deploy for each of our experiments in Section 4 4 instances of our LLM all running in parallel. We load and use LLMs through the Hugging Face Transformers Python library[8]. Our method relies on a simple client-server architecture where the RL script acts as a client sending requests to LLMs. This client communicates with a master server which dispatches the call over multiple servers (i.e. one per LLM). Once each LLM has computed its subset of the call, the master gathers results and sends the response to the RL client. We use Pytorch Distributed[9] with the GLOO backend for communication (hence possible both on CPU-only and GPU setups). We wrap all these in a Python library called *Lamorel* which can dispatch calls over the deployed LLMs from a single line of code in the RL loop asking for actions' probability for all environments. Using this method, we observe a quasi-linear scaling with the number of deployed LLMs.

Once transitions have been collected, we update our LLM using the PPO loss. For this, *Lamorel* helps parallelize the gradients' computation with a Distributed Data Parallelism[10] setup where forward and backward passes over transitions are also dispatched on the different instances of our LLMs. Then, *Lamorel* helps gather gradients and update each LLM (as well as their value head) the same way. In addition, *Lamorel* also helps define a custom computational graph linked to the LLM. We use this to add MLPs on top of our Flan-T5 model for the value head (see experiments with action heads in Section B.3).

When using **Flan-T5 780M**, each LLM instance is distributed (Vertical Model Parallelism[11]) 2 Nvidia A100 80GB GPUs requiring thus a total of 8 Nvidia A100 80GB GPUs to run an experiment (2 GPUs $\times 4$ LLM instances). For **Flan-T5 80M** and **Flan-T5 3B**, we respectively use 1 Nvidia V100 32GB and 4 Nvidia A100 80GB per LLM instance.

In total, to conduct experiments and ablations we use 160 GPU.hours on the Nvidia V100 32G and 18880 GPU.hours on Nvidia A100 80GB.

F finetuning details

F.1 PPO finetuning details

We reused PPO's hyperparameters from Ramamurthy et al. [2022] and did not perform any further tuning (see Table 7). We used an Adam [Kingma and Ba, 2014] optimizer with the hyperparameters listed in Table 8). For additional heads, we used MLPs with 3 hidden layers of 1024 units with Sigmoid activation.

[8] `https://huggingface.co/docs/transformers/index`

[9] `https://pytorch.org/docs/stable/distributed.html`

[10] `https://pytorch.org/tutorials/intermediate/ddp_tutorial.html`

[11] Layers are spread across GPUs (`https://huggingface.co/docs/transformers/v4.15.0/parallelism`)

Table 7: PPO hyperparameters

Variable	Value
Number of transitions collected between two updates	1280 (32 environments $\times 40$ steps in each environment)
Number of epochs per update	4
Batch size	64
Entropy loss coefficient	0.01
Value function loss coefficient	0.5
Discount factor	0.99
lr	1×10^{-6}
λ factor of the Generalized Advantage Estimator	0.99
Clipping parameter ϵ	0.2
Maximum gradient norm	0.5

Table 8: Adam hyperparameters

Variable	Value
Learning rate	1×10^{-6}
ϵ	1×10^{-5}
β_1	0.9
β_2	0.999

During our experiments, we observe that the output of LLMs with the language modeling heads tend to be small ($\mathbb{P}_{LLM}(a_i|p) < 10^{-9}$) as the vocabulary is large (32128 tokens) meaning that probability of tokens is very small and therefore the product of tokens probability is even smaller. Thus, once the softmax step is performed as per Equation (2), the probability distribution over the possible actions is close to uniform, preventing the pretrained LLM to use its useful bias when interacting with the environment. To tackle this issue, we use a variable temperature τ in the softmax. τ is equal to the maximum probability returned by the LLM over the action space. So the distribution over the possible actions is given by:

$$\mathbb{P}(a_i|p) = \frac{e^{\mathbb{P}_{LLM}(a_i|p)/\tau}}{\sum_{a_j \in \mathcal{A}} e^{\mathbb{P}_{LLM}(a_j|p)/\tau}}$$

with $\tau = \max_{\{a_j \in \mathcal{A}\}} \mathbb{P}_{LLM}(a_j|p)$.

F.2 Behavioral Cloning

In Section 4.4, we show how grounding using RL differs from BC. For this, we finetune **Flan-T5 780M** on 400.000 transitions collected on the *Go To <object>* task. As indicated in Table 5, GFlan-T5 obtains a 0.81 success rate on the 1000 test episodes of the *Go To <object>* task. Hence by finetuning Flan-T5 to imitate GFlan-T5, one could expect an on-par performance (or worse, but not better). We therefore use GFlan-T5 to collect 400.000 transitions and finetune Flan-T5 using them. However, the stochasticity in the GFlan-T5 policy leads to deceptive transitions in the dataset (potentially harmful for BC). We thus also assess whether using optimal transitions to finetune Flan-T5 leads to better results than GFlan-T5. To collect optimal trajectories, we use the bot provided by BabyAI and also gather 400.000 transitions on the *Go To <object>* task.

For finetuning, we use Causal Language Modeling with the same prompt as the one given to our LLM agents in Section 4 as input and the performed action as label. We use the same learning rate as the one used by Rae et al. [2021] to generate Flan-T5 (i.e. 5×10^{-4}) and perform a single epoch on the 400.000 examples.

G Confidence interval

In sections 4.2 and 4.3, we perform several generalization tests. For each test we report the success rate over 2 seeds tested on 1000 episodes each. In the following, we explain how we get the 99% confidence interval.

G.1 Confidence intervals for GFlan-T5, Flan-T5 and DRRN

We model the success of an agent, trained with the seed i, on a task (i.e. episode with its associated task) using a Bernoulli variable $X^i \sim \mathcal{B}(p_i)$, with p_i the probability of success of the agent . The number of successes after doing n episodes is the random variable $Y_n^i = \sum_{k=0}^{n} X_k^i$ which follows a binomial law $\mathcal{B}(n, p_i)$. If n is large enough, the binomial distribution can be approximated by a normal distribution[12]. Thus we have

$$\begin{cases} p_i & \sim \mathcal{N}(p, \tau^2) \\ Y_n^i | p_i & \sim \mathcal{N}(n\, p_i, n\, p_i(1-p_i)) \end{cases} \tag{4}$$

where p is the mean success rate and τ the variance.

Moreover, one property of normal random variables is that if

$$\begin{cases} V & \sim \mathcal{N}(V_0, \Sigma_V) \\ U|V & \sim \mathcal{N}(U_0 + XV, \Sigma_{U|V}) \end{cases} \tag{5}$$

for any X,then

$$\begin{pmatrix} U \\ V \end{pmatrix} \sim \mathcal{N}(\begin{pmatrix} U_0 + XV_0 \\ V_0 \end{pmatrix}, \begin{pmatrix} X\Sigma_V X^T + \Sigma_{U|V} & X\Sigma_V \\ \Sigma_V X^T & \Sigma_V \end{pmatrix}) \tag{6}$$

Hence we obtain $U \sim \mathcal{N}(U_0 + XV_0, X\Sigma_V X^T + \Sigma_{U|V})$.

By identification with Equation 4, we have

$$Y_n^i \sim \mathcal{N}(np, (n\tau)^2 + n\, p_i(1-p_i)) \tag{7}$$

We can rewrite it using the random variable SR_i the success rate of the the agent (trained with seed i) during the test time (over n trajectories).

$$SR_i = \frac{Y_n^i}{n} \sim \mathcal{N}(p, \tau^2 + \frac{p_i(1-p_i)}{n}) \tag{8}$$

Because $\frac{p_i(1-p_i)}{n} \leq \frac{0.25}{n} \xrightarrow[n\to\infty]{} 0$ and n is large, we can neglect this term with respect to τ in equation above (we verify at the end that we rightfully neglected it) and obtain:

$$SR_i \sim \mathcal{N}(p, \tau^2) \tag{9}$$

Using the maximum likelihood estimation for normal random variables, we get with a 99% confidence interval:

$$\hat{p} \pm\ 2.58 \frac{\hat{\tau}}{\sqrt{s}} \tag{10}$$

$$\begin{cases} \hat{p} & = \frac{1}{s} \sum_{i=1}^{s} SR_i \\ \hat{\tau}^2 & = \frac{1}{(s-1)} \sum_{i=1}^{s} (\hat{p} - SR_i)^2 \end{cases} \tag{11}$$

with s the number of seeds used, $\hat{p}$ the estimator for p and $\hat{\tau}^2$ the unbiased sample variance.

[12]using the Berry-Essen theorem, the approximation is good enough if $n > 9\frac{p(1-p)}{p}$ and $n > 9\frac{p}{p(1-p)}$

G.2 Confidence intervals for random agents

As previously mentioned, we model the success of an agent using a Bernoulli variable $X \sim \mathcal{B}(p)$, with p the probability of success. The measured success rate after doing n episodes is the random variable $SR_n = \frac{1}{n}\sum_{k=0}^{n} X_k$ which also follows Bernoulli's law $\mathcal{B}(p)$. Following Hoffending's inequality, we have:

$$\mathbb{P}(|SR_n - p| > \varepsilon) < 2\exp\left(-2n\varepsilon^2\right) = \delta \tag{12}$$

with δ the error.

Thus if we use $n = 1000$ episodes to measure the success rate and we want a confidence of 99% ($\delta = 0.01$) with $\varepsilon = \sqrt{|\frac{1}{2n}\ln\frac{\delta}{2}|}$, we get $\varepsilon = 0.05$.

H Word substitutions for generalization tests

For the generalization tests given in the sections 4.2, 4.3, and D, we use the dictionaries given below to substitute some words by others.

H.1 Out of vocabulary

To generate descriptions with out-of-vocabulary nouns and adjectives, we modify the prompt by substituting words as per Table 9.

Table 9: Out-of-vocabulary substitutions for Nouns and adjectives

Original Word	New Word
key	chair
ball	table
box	car
red	vermillion
green	jade
blue	cyan
purple	violet
yellow	golden
grey	silver

H.2 Invented words

Similarly to Section H.1, we apply the substitutions indicated in Table 10.

Table 10: Invented substitutions for Nouns and adjectives

Original Word	New Word
key	dax
ball	xolo
box	azfe
red	faze
green	jatu
blue	croh
purple	vurst
yellow	gakul
grey	sil

H.3 Synonym actions

In 11, we choose the synonym actions to avoid as much as possible to reuse already used words in the finetuning (only "left" and "right" cannot be changed). To verify that Flan-T5-Large considers these words as synonyms we ask it: *"Answer the following yes/no question by reasoning step-by-step. Are <original action> and <synonym action> synonymous?"*. We retain the synonym only if it considers that this is the case.

Table 11: Synonym actions

Original Words	**Synonyms**
turn left	rotate left
turn right	rotate right
go forward	move ahead
pick up	take
drop	release
toggle	switch

H.4 Translation to French

We give in Table 12 the chosen translation for the french environment (the adjectives are given in the feminine form as all the objects are feminine).

Table 12: French translation

English	**French**
turn left	tourner à gauche
turn right	tourner à droite
go forward	aller tout droit
pick up	prendre
drop	lâcher
toggle	basculer
go to a/the adj n	aller à une/la n adj
steps	pas
You see a <*object*> <*location*>	Tu vois une <objet><location>
You see a(n) *open/closed* door <*location*>	Tu vois une porte *ouverte/fermée* <location>
You carry a <*object*>	Tu portes un <objet>
key	clef
ball	balle
box	boîte
red	rouge
green	verte
blue	bleue
purple	violette
yellow	jaune
grey	grise