Le langage comme outil cognitif pour les agents ouverts

Par **Thomas CARTA**

Sous la supervision de **Pierre Yves OUDEYER**

Co-supervision : **Olivier SIGAUD, Sylvain LAMPRIER**

Thèse présentée pour obtenir

le grade de Docteur

Université de Bordeaux

École Doctorale de Mathématiques et d'Informatique

Soumis le 1 Septembre 2025.

Composition du jury:

Dr. Georg MARTIUS	Professeur	Université de Tübingen	Rapporteur
Dr. Pierre-Luc BACON	Professeur Associé	Université de Montreal DIRO	Rapporteur
Dr. Edward HUGHES	Chercheur et Professeur Associé	Deepmind et The London School of economics	Examinateur
Dr. Martha WHITE	Professeure Associée	Université d'Alberta	Examinatrice
Pr Jean PONCE	Professeur	Ecole normale supérieure-PSL	Examinateur
Dr. Pierre-Yves OUDEYER	Directeur de Recherche	INRIA	Directeur
Pr. Olivier SIGAUD	Professeur	Université de la Sorbonne	Directeur
Sylvain Lamprier	Professeur	Université d'Anger	Invité

Contents

Acknowledgments vii

Abstract ix

Résumé Long xi

Introduction xiv

Glossary of mathematical symbols and abbreviations xix

I Background 1

1 Conceptual background 2
1.1 Machines in the world . . . 2
1.2 The Reinforcement Learning framework . . . 5
1.3 Language Modeling . . . 12

2 Literature overview 17
2.1 Linguistic agents . . . 17
2.2 Curiosity-driven exploration . . . 22

II Modelling the environment 29

3 Introduction 30

4 Grounded internalisation of environment rules 32
4.1 Motivations . . . 32
4.2 Contributions . . . 33
4.3 Related work . . . 33
4.4 GLAM: Grounding LLMs with online RL . . . 36
4.5 Experiments . . . 38
4.5.1 Q1. How fast can an LLM adapt and learn to solve tasks? . . . 40
4.5.2 Q2. Generalization to new objects . . . 47
4.5.3 Q3. Generalization to new tasks . . . 48

4.5.4 Q4. What is the impact of using RL vs Behavioral Cloning for grounding? 50
4.6 Discussion . . . 50

5 Abstract internalisation of environment rules 52
5.1 Motivations . . . 52
5.2 Contributions . . . 53
5.3 Related work . . . 53
5.4 Improving LLMs' world model with WorldLLM . . . 55
5.5 Experiments . . . 58
5.5.1 Q1. Environment modeling . . . 60
5.5.2 Q2. Impact of hypothesis generation . . . 63
5.6 Discussion . . . 65

6 Discussion 67

III Goal space exploration 69

7 Introduction 70

8 Navigate the goal space 72
8.1 Motivations . . . 72
8.2 Contributions . . . 73
8.3 Related work . . . 74
8.4 MAGELLAN: Measure and generalize learning progress with a Large Language Model . . . 75
8.5 Experiments . . . 79
8.5.1 The Little-Zoo environment as a testbed . . . 79
8.5.2 Q1. Competence estimation . . . 80
8.5.3 Q2. LLM agent training . . . 82
8.5.4 Q3. Generalization to new goals . . . 85
8.5.5 Q4. Adaptation to evolving goal space . . . 87
8.6 Discussion . . . 88

9 Generate new learnable goals 90
9.1 Motivation . . . 90
9.2 Contributions . . . 91
9.3 Related work . . . 91
9.4 AutotelicLLM: Generating creative and learnable goal with a Large Language Model . . . 93
9.5 Experiments . . . 96
9.5.1 Q1.Does AutotelicLLM generate diverse goals adapted to the agent abilities? . . . 97
9.5.2 Q2. Do the handcrafted categories necessary for generating diverse goals? . . . 99
9.6 Discussion . . . 100

10 Discussion 102

IV Efficient learning of new goals **104**

11 Introduction **105**

12 Using pure linguistic abstraction to generate a reward signal **107**
12.1 Motivation . . . 107
12.2 Contributions . . . 108
12.3 Related work . . . 109
12.4 EAGER: Asking and Answering Questions for Automatic Reward Shaping in Language-guided RL . . . 111
12.4.1 Problem statement . . . 111
12.4.2 EAGER . . . 112
12.5 Experiments . . . 115
12.5.1 Environment . . . 115
12.5.2 Q1. Sample efficiency . . . 117
12.5.3 Q2. Robustness . . . 119
12.6 Discussion . . . 122

13 Efficient learning of skills with hierarchical learning **123**
13.1 Motivation . . . 123
13.2 Contributions . . . 124
13.3 Related work . . . 125
13.4 HERAKLES: Hierarchical Skill Compilation for Open-ended LLM Agents . . . 127
13.5 Experiments . . . 131
13.5.1 Experimental setup . . . 131
13.5.2 Heterogeneous goal space . . . 133
13.5.3 Sample efficiency . . . 133
13.5.4 Generalization to new goals . . . 135
13.6 Discussion . . . 137

14 Discussion **138**

V Conclusion **140**

15 Conclusion **141**
15.1 Summary of contributions . . . 141
15.2 Limitations . . . 142
15.3 Future Directions . . . 143

Bibliography **145**

Appendices **209**

A Appendix of GLAM **210**
A.1 Environments . . . 210
A.1.1 BabyAI . . . 211
A.1.2 BabyAI-Text . . . 212
A.2 Additional results . . . 213

A.2.1 Per-task success rate . . . 213
A.2.2 Averaging success rate over the tasks . . . 213
A.2.3 Textual vs symbolic representation . . . 214
A.2.4 Impact of varying action space and distractors . . . 214
A.3 Evolution of actions distribution on evaluation prompts . . . 217
A.4 Generalization tests details . . . 224
A.4.1 Recapitulating results table . . . 224
A.4.2 Complementary tests for Q2 . . . 224
A.4.3 Complementary tests for Q3 . . . 225
A.5 Distributed experimental setup . . . 226
A.6 finetuning details . . . 227
A.6.1 PPO finetuning details . . . 227
A.6.2 Behavioral Cloning . . . 227
A.7 Confidence interval . . . 228
A.7.1 Confidence intervals for GFlan-T5, Flan-T5 and DRRN . . . 228
A.7.2 Confidence intervals for random agents . . . 229
A.8 Word substitutions for generalization tests . . . 229
A.8.1 Out of vocabulary . . . 229
A.8.2 Invented words . . . 230
A.8.3 Synonym actions . . . 230
A.8.4 Translation to French . . . 231

B Appendix of WorldLLM 232
B.1 The Playground-Text Environment . . . 232
B.2 WorldLLM details . . . 234
B.3 Generalization Tests . . . 237
B.4 Additional Results . . . 238
B.4.1 Evolution of likelihoods . . . 238
B.4.2 Analysis of the Metropolis algorithm in the Scientist . . . 238
B.4.3 Analysis of generated hypotheses . . . 240
B.4.4 A deeper study RL-ALP . . . 240

C Appendix of MAGELLAN 244
C.1 Little-Zoo environment . . . 246
C.1.1 Environment mechanics . . . 246
C.1.2 Goal space generation . . . 246
C.1.3 Example of impossible goals . . . 247
C.1.4 Goal repartition . . . 248
C.2 Comparison of LP methods . . . 249
C.3 Implementation details . . . 250
C.3.1 LLM-based RL agent . . . 250
C.3.2 MAGELLAN . . . 251
C.3.3 Baselines . . . 252
C.3.4 Compute budget . . . 253
C.4 Additional results . . . 254
C.4.1 Ablations on the MAGELLAN architecture . . . 254
C.4.2 Q1. Competence estimation properties . . . 257

C.4.3 Q2. Training an LLM-based RL agent with MAGELLAN 260
C.4.4 Q3. MAGELLAN's generalization abilities 261
C.4.5 Q4. Leveraging the generalization abilities when facing new goals . . 263

D Appendix of AutotelicLLM **267**
D.1 Helper functions for the algorithm 267
D.1.1 $\mathcal{F}_\Lambda$ the filter function 267
D.1.2 Goal and subgoals embedding 268
D.1.3 Filtering valid goals 268
D.2 Hyperparameters 268
D.2.1 PPO AGENT 268
D.2.2 Autotelic architecture hyperparameters 269
D.3 Example of reward functions generated by the LLM 269
D.4 Prompt for the LLM 274

E Appendix of EAGER **281**
E.1 Tasks description and assumptions used for the different method of reward shaping 281
E.2 Training details 282
E.2.1 QA architecture and pre-training 282
E.2.2 Agent architecture and training 284
E.2.3 RIDE 284
E.2.4 ELLA 285
E.2.5 EAGER and RIDE 285
E.3 More results on EAGER 286
E.4 Commented EAGER algorithm 290
E.5 Hyperparameters tables 291

F Appendix of HERAKLES **292**
F.1 Environment detail 294
F.1.1 Achievement tree 294
F.1.2 Modifications 294
F.1.3 Goal difficulty 295
F.1.4 Observations 295
F.2 Complementary Results 297
F.2.1 Evolution of success rate 297
F.2.2 Generalization results 298
F.2.3 Evolution of high-level sampling strategy 299
F.3 Hierarchical Agent 301
F.3.1 HERAKLES implementation 301
F.3.2 HERAKLES ablations 302
F.4 High-level policy 304
F.4.1 High-level policy architecture 304
F.4.2 Prompts for the High-Level policy 304
F.5 Low-level policy 305
F.5.1 Low-level policy architecture 306
F.5.2 Low-level training 307

F.5.3 Ablation of the AWR buffer size 308
F.6 Low-level success rate estimator 309
F.6.1 Prompt of the low-level success rate estimator 309
F.6.2 Training of the low-level success rate estimator 309
F.6.3 Composition of the set of skills 310
F.7 Implementation of MAGELLAN 311
F.7.1 Prompt of the high-level success rate estimator 311
F.8 Baselines details 312
F.8.1 FuN 312
F.8.2 POAD training 313
F.9 Hyper-parameters 314

Acknowledgments

First of all, I would like to thank my supervisors Pierre-Yves Oudeyer, Olivier Sigaud and Sylvain Lamprier. They guided me with kindness and encouraged me to give my best. Through many meetings they take the time to listen and discuss my ideas. Their different perspectives on the topics we discussed made our discussions extremely interesting and were invaluable when I had to write papers.

I would also like to take this opportunity to say how much I enjoyed my time with the Flowers team, undoubtedly one of the best places to do your thesis. The atmosphere was fantastic and it was always a pleasure to come to work. I would particularly like to thank Clément Romac, with whom I had the good fortune to collaborate. I am especially grateful for the time he spent on some of my bugs and, above all, for the scientific discussions we had. His extensive knowledge of the field, his rigour and his good spirit were a source of inspiration to me. I would also like to thank Loris Gaven, Guillaume Lévy, Guillaume Pourcel and Grgur Kovač, with whom I had the pleasure of working.

Finally, I would like to thank my family for believing in me and supporting me: my father Jean Paul, my mother Hélène and my sister Mathilde. Thank you also to my friends Juan, Gabriel, Antoine, Benjamin and so many others for the moments of sporting and gastronomic joy we have shared. Special thanks to Anaëlle for her unfailing support and constant help while I was writing my thesis, which was invaluable to me.

Abstract

Titre de la thèse décrivant avec les mots clés nécessaires la thématique précise étudiée et les travaux effectuées

Résumé :

Les êtres humains ont toujours inventé des outils pour étendre leurs capacités physiques et cognitives. Les premiers outils amplifiaient la force physique, tandis que des inventions plus récentes, telles que l'abaque, la Pascaline et la machine analytique de Babbage, facilitaient les tâches mentales. Le XXe siècle a vu l'apparition des ordinateurs programmables, concrétisant ainsi la vision d'Ada Lovelace d'une créativité artificielle. Aujourd'hui, les grands modèles linguistiques (LLM) poursuivent cette trajectoire, permettant des découvertes et des générations avancées. Malgré les progrès réalisés, les systèmes d'IA actuels ne fonctionnent que dans des domaines définis par l'homme, où les problèmes et les stratégies sont prédéfinis. L'ambition à long terme est l'intelligence artificielle générale (AGI), où les agents agissent efficacement même sans objectifs prédéfinis. Pour y parvenir, il faut des agents ouverts : des systèmes qui explorent, apprennent et réutilisent de manière autonome les connaissances tout au long de leur vie. Les êtres humains illustrent ce processus, guidés par la curiosité, la motivation intrinsèque et des objectifs qu'ils se fixent eux-mêmes. Le langage joue un rôle central. Il permet l'abstraction, le raisonnement et la planification, ce qui en fait un moyen puissant pour la découverte ouverte chez les agents artificiels. Pour fonctionner, un agent ouvert doit : (1) explorer des environnements de manière autonome, (2) s'adapter à des règles en évolution et (3) acquérir efficacement de nouvelles compétences à mesure que la complexité augmente. Cette thèse exploite le langage pour faire progresser les agents ouverts dans trois dimensions : la modélisation de l'environnement, l'exploration des objectifs et l'apprentissage efficace. Premièrement, les LLMs sont utilisés à la fois comme agents d'apprentissage par renforcement incarnés et comme générateurs d'hypothèses pour améliorer les modèles du monde. Deuxièmement, le langage est utilisé pour évaluer, généraliser et générer des objectifs, ce qui favorise un apprentissage évolutif. Troisièmement, de nouvelles méthodes utilisent le langage pour la mise en forme des récompenses et le contrôle hiérarchique, ce qui permet un apprentissage efficace des politiques et la réutilisation des compétences. Dans l'ensemble, cette thèse soutient que le langage n'est pas seulement un outil de communication, mais aussi un moteur cognitif. En intégrant les LLMs dans les architectures des agents, nous nous rapprochons de systèmes autonomes capables d'apprendre en continu, d'explorer de manière autonome et de faire des découvertes adaptatives tout au long de leur vie.

Mots-clés :

Unité de Recherche INRIA, 33000 Bordeaux, France.

Title of the thesis describing with the necessary keywords the precise topic studied and the research undertaken

Abstract:

Humans have always invented tools to extend their physical and cognitive abilities. Early tools amplified bodily force, while later inventions like the abacus, Pascaline, and Babbage's Analytical Engine supported mental tasks. The 20th century brought programmable computers, realizing Ada Lovelace's vision of machine creativity. Today, large language models (LLMs) continue this trajectory, enabling advanced discovery and generation. Despite progress, current AI systems succeed only in human-defined domains, where problems and strategies are pre-specified. The long-term ambition is Artificial General Intelligence (AGI), where agents act effectively even without predefined goals. Achieving this requires open-ended agents: systems that autonomously explore, learn, and repurpose knowledge across their lifetimes. Humans exemplify this process, guided by curiosity, intrinsic motivation, and self-generated goals. Language plays a central role. It enables abstraction, reasoning, and planning, making it a powerful medium for open-ended discovery in artificial agents. To function, an open-ended agent must: (1) explore environments autonomously, (2) adapt to evolving rules, and (3) learn new skills efficiently as complexity grows. This thesis leverages language to advance open-ended agents across three dimensions: environment modelling, goal exploration, and efficient learning. First, LLMs are used both as embodied reinforcement learning agents and as hypothesis generators to improve world models. Second, language is employed to evaluate, generalize, and generate goals, supporting scalable curriculum learning. Third, novel methods use language for reward shaping and hierarchical control, enabling efficient policy learning and skill reuse. Overall, the thesis argues that language is not only a communication tool but a cognitive engine. By embedding LLMs into agent architectures, we move closer to autonomous systems capable of continuous learning, self-directed exploration, and adaptive lifelong discovery.

Keywords:

Research Unit INRIA, 33000 Bordeaux, France.

Résumé Long

Le progrès humain a été stimulé par l'invention d'outils qui étendent les capacités naturelles. Les premiers instruments, tels que les éclats de pierre, les lances et les dispositifs mécaniques comme les poulies, ont amplifié la force physique. Au fil des siècles, cette trajectoire s'est étendue aux domaines cognitifs. L'abaque, la Pascaline et le moteur analytique de Babbage ont marqué les premiers efforts visant à soutenir le raisonnement. Au XXe siècle, les travaux théoriques de Turing et l'émergence des ordinateurs programmables ont confirmé la vision d'Ada Lovelace d'une machine capable de créativité. Aujourd'hui, l'intelligence artificielle, en particulier les grands modèles linguistiques (LLM), poursuit dans cette voie, accélérant la découverte et augmentant la pensée. Les systèmes modernes font preuve de capacités remarquables, notamment l'utilisation fluide du langage, la génération d'images et des applications scientifiques telles que la prédiction des protéines. Cependant, ces systèmes ne fonctionnent que dans le cadre de tâches et de stratégies spécifiées par l'homme. Leur champ de découverte reste limité par des directives externes. L'ambition à long terme de ce domaine est l'intelligence artificielle générale (AGI). Contrairement aux systèmes actuels, l'AGI agirait efficacement sans objectifs prédéfinis. Pour atteindre cet idéal, formulé depuis la conférence de Dartmouth en 1956, il faut créer des « agents ouverts » (open-ended agents en anglais). Ces agents doivent s'engager dans une exploration autonome, accumuler des connaissances au fil du temps et affiner leurs compétences à des niveaux de complexité croissants. Ils doivent également réutiliser et réinterpréter les découvertes antérieures, permettant ainsi des cycles récursifs de progrès, une caractéristique essentielle de l'apprentissage humain. Les êtres humains sont des modèles exemplaires d'apprentissage ouvert.

Dès la petite enfance, nous acquérons des capacités perceptives et motrices ; pendant l'enfance, nous développons le langage et les interactions sociales ; à l'âge adulte, certains atteignent la maîtrise dans des domaines qui élargissent les connaissances collectives. Une caractéristique déterminante de l'intelligence humaine est la capacité à se fixer soi-même des objectifs. Poussés par la curiosité et une motivation intrinsèque, les humains s'adaptent avec souplesse aux nouveaux défis. Reproduire cet apprentissage motivé par la curiosité dans des systèmes artificiels est essentiel pour développer des agents qui vont au-delà de l'exécution réactive. Des études récentes ont fourni des définitions formelles de l'exploration ouverte (open-endedness). Elles mettent l'accent sur la génération d'artefacts nouveaux et apprenables. Le langage est particulièrement puissant à cet égard. Au-delà de la communication, il structure le raisonnement, permet l'abstraction et soutient la formation d'hypothèses. Les perspectives

théoriques de Piaget et Vygotsky considèrent le langage comme un instrument cognitif essentiel à l'exploration et à la planification ouvertes.

Pour un agent artificiel, trois exigences fondamentales se dégagent. Premièrement, il doit explorer son environnement de manière autonome, en découvrant et en poursuivant de nouveaux objectifs. Deuxièmement, il doit intérioriser des structures en évolution et s'adapter à des dynamiques changeantes. Troisièmement, il doit apprendre efficacement, en veillant à ce que l'acquisition de compétences reste accessible à mesure que la complexité augmente. Ces capacités garantissent la poursuite des progrès tout au long d'un apprentissage de longue durée. Cette thèse répond à ces exigences en tirant parti du langage. Ses contributions sont structurées en quatre parties.

La partie I présente les concepts fondamentaux nécessaires à la compréhension des contributions suivantes. Le chapitre 1 passe en revue les idées clés de l'apprentissage par renforcement, de l'autotélisme, de la motivation intrinsèque et des grands modèles linguistiques (LLM). Le chapitre 2 passe en revue la littérature pertinente sur les agents linguistiques en général, puis se concentre davantage sur l'exploration motivée par la curiosité.

La partie II aborde le problème de la modélisation de la dynamique environnementale. Le chapitre 4 présente une méthode dans laquelle un LLM est déployé comme agent d'apprentissage par renforcement incarné, en ancrant sa sémantique par l'interaction. Dans le chapitre 5, nous adoptons une approche complémentaire dans laquelle le LLM sert de moteur d'hypothèses, construisant et affinant des modèles de l'environnement à partir des transitions observées. La section 5.5.1 explore comment les améliorations de la modélisation du monde peuvent servir de signaux de motivation intrinsèque pour une exploration systématique.

La partie III se concentre sur l'espace des objectifs. Le chapitre 8 présente une méthode d'évaluation de la compétence d'un agent sur des objectifs exprimés linguistiquement à l'aide d'un LLM. L'étude montre que les régularités linguistiques permettent un apprentissage évolutif et une généralisation à travers de grands espaces d'objectifs. Le chapitre 9 propose un travail préliminaire sur un mécanisme complémentaire permettant de générer de nouveaux objectifs adaptés aux capacités actuelles de l'agent, guidé par des signaux de progression de l'apprentissage.

La partie IV est consacrée à l'amélioration de l'efficacité de l'apprentissage des politiques. Le chapitre 12 présente une méthode basée sur le langage pour la mise en forme des récompenses via des requêtes structurées à compléter, qui contribue à atténuer les difficultés liées à la rareté des récompenses dans les environnements conditionnés par le langage. Le chapitre 13 explore un cadre d'apprentissage par renforcement hiérarchique dans lequel un LLM fonctionne comme un contrôleur de haut niveau sélectionnant parmi les compétences de bas niveau apprises. Ces compétences sont compilées et réutilisées de manière incrémentielle, ce qui permet un apprentissage et une généralisation efficaces avec des exigences informatiques raisonnables.

Dans l'ensemble, cette thèse soutient que le langage, lorsqu'il est utilisé comme outil cognitif et structurel, peut soutenir les capacités fondamentales requises pour l'apprentissage ouvert.

En intégrant les LLM dans l'architecture d'agents ouverts, nous visons à nous rapprocher de systèmes qui, comme les humains, peuvent apprendre en continu, découvrir de manière autonome et s'adapter de manière significative tout au long de leur vie.

Introduction

Throughout history, humans have continually invented tools to overcome their physical limitations, extend their bodily capabilities, and transcend natural constraints. Early examples include stone splinters used in place of fingernails, spears as bodily extensions, and mechanical devices such as pulleys and steam engines that magnified human force. Over time, this tool-making trajectory expanded beyond the physical realm. With the invention of the abacus, followed by the Pascaline—the first mechanical calculator of the 17th century—and Babbage's Analytical Engine, humans began to design instruments to assist with cognitive tasks. This trend accelerated during the 20th century, with the theoretical foundation laid by Turing and the emergence of the first programmable computers. Machines increasingly took on tasks previously considered uniquely human, echoing Ada Lovelace's visionary insight that:

"*the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.*" Lovelace & Menabrea (1843)

In the contemporary era, with groundbreaking advances in artificial intelligence, in particular deep learning methods among which large language models (LLMs), humans are continuing this trajectory—augmenting their cognitive capabilities through tools that accelerate and amplify discovery.

Despite these remarkable accomplishments—ranging from fluent language and image generation to protein structure prediction—modern artificial intelligence systems have achieved success only within domains explicitly guided by human supervision. In these contexts, humans have defined both the problems to be solved and the strategies for solving them. The underlying structure of discovery remains human-directed.

Looking forward, a major ambition of the field is to build systems capable of assisting with problems whose solutions are unknown and whose solution paths are undefined. Achieving this kind of Artificial General Intelligence (AGI)—where agents operate effectively even when the goals are unspecified—is a guiding ideal of artificial intelligence research, dating back to the seminal 1956 Dartmouth Conference.

A fundamental challenge in the pursuit of general artificial intelligence lies in the development of *open-ended agents*—systems capable of autonomous exploration, cumulative discovery,

and lifelong acquisition of increasingly complex skills. Such agents must not only engage with their environments to uncover new knowledge, but also possess the capacity to abstract, reinterpret, and repurpose previous discoveries to catalyze further progress. This recursive, self-sustaining learning cycle is a hallmark of human cognition and development.

Humans are exemplary open-ended learners across the entire span of life (Gottlieb & Oudeyer, 2018). From early infancy, we acquire perceptual and motor skills that enable us to interact with our surroundings. As toddlers, we develop sophisticated forms of communication and social interaction. In adulthood, some individuals achieve mastery in domains such as science, the arts, or sports, thereby contributing to the advancement of human knowledge and culture. Over time, we construct vast repertoires of concepts, strategies, and skills drawn from a virtually unbounded set of possibilities.

One of the most distinctive attributes of human intelligence is the capacity to self-generate objectives. Rather than acting solely under external instruction, we continuously invent new goals and challenges, guided by intrinsic motivations and internal feedback mechanisms—particularly signals of learning progress. This ability to engage in curiosity-driven, self-directed learning enables humans to adapt flexibly and creatively to dynamic environments. Emulating these capabilities in artificial agents is a key milestone toward realizing systems that are not merely reactive executors, but autonomous explorers capable of lifelong discovery.

Recent works by Sigaud et al. (2024) and Edward et al. (2024) have contributed formal definitions of open-endedness, emphasizing the need for agents to produce sequences of novel and learnable artifacts. Language, by virtue of its abstraction and compositional structure, offers a particularly rich medium for generating and manipulating such artifacts.

Rooted in foundational theories from Piaget and Vygotsky, language is viewed not merely as a communication tool, but as a powerful cognitive instrument. It enables us to conceptualize, reason, and generalize; to structure our thoughts; and to formulate hypotheses about the world. These characteristics make language especially suited for supporting open-ended discovery and planning in artificial agents.

An open-ended agent must operate in two interconnected spaces: the environment and the goal space. First, it must continually internalize the evolving rules and structures of its environment, adapting to new dynamics as they emerge. Second, it must be capable of intrinsically motivated exploration of its environment, enabling the autonomous discovery and pursuit of novel goals. Third, and critically, it must be able to acquire new skills efficiently. As the complexity of pursued goals increases, learning time must remain tractable to avoid stagnation. This implies the need for increasingly effective learning mechanisms as the agent matures.

Document structure

In this thesis, we present a body of work that leverages the unique properties of language to address these fundamental capabilities of open-ended agents. The thesis is organized into

four parts, the first presenting the concepts and the literature the following three targeting a specific dimension of open-endedness.

Part I introduces the foundational concepts required to understand the subsequent contributions. Chapter 1 reviews key ideas in reinforcement learning, autotelism, intrinsic motivation, and LLMs. Chapter 2 surveys relevant literature, on linguistic agent in general, then more focus on curiosity driven exploration.

Part II addresses the problem of modelling environmental dynamics. Chapter 4 presents a method where an LLM is deployed as an embodied reinforcement learning agent, grounding its semantics through interaction. In Chapter 5, we adopt a complementary approach in which the LLM serves as a hypothesis engine, constructing and refining models of the environment from observed transitions. We also explore how improvements in world modelling can serve as intrinsic motivation signals for systematic exploration.

Part III shifts focus to the goal space. Chapter 8 introduces a method for evaluating an agent's competence on linguistically stated goals using an LLM. The study shows that linguistic regularities enable scalable curriculum learning and generalisation across large goal spaces. Chapter 9 proposes a preliminary work on a complementary mechanism for generating new goals tailored to the agent's current capabilities, guided by signals of learning progress.

Part IV is dedicated to improving the efficiency of policy learning. Chapter 12 presents a language-based method for reward shaping via structured fill-in-the-blank queries, which helps mitigate the challenges of sparse rewards in language-conditioned environments. Chapter 13 proposes an un published work that explores a hierarchical reinforcement learning framework in which an LLM operates as a high-level controller selecting among learned low-level skills. These skills are incrementally compiled and reused, enabling effective learning and generalisation with reasonable computational demands.

Altogether, this thesis argues that language, when used as a cognitive and structural tool, can support the core capabilities required for open-ended learning. By integrating LLMs into the architecture of open-ended agents, we aim to move closer to systems that, like humans, can learn continuously, discover autonomously, and adapt meaningfully throughout their lifetimes.

Collaborations

This research is the result of numerous collaborations and rich exchanges that have greatly contributed to both its development and my own scientific growth. I have benefited immensely from regular discussions with members of the FLOWERS team, whose insights have deepened my understanding of many core aspects of this work. In particular, I would like to thank Clément Romac (INRIA), with whom I collaborated on several projects presented in this manuscript (Chapters 4, 5, 8, and 13). Our exchanges were instrumental in refining my conceptualisation of language models as agents and in shaping my understanding of language as a cognitive tool. Together, we also co-authored a popular science article in collaboration

with François Lassagne (*Pour la Science*) aimed at disseminating our work on grounding LLM-based agents (Chapter 4) to a broader audience.

I also had the opportunity to collaborate with Loris Gaven (INRIA) and Guillaume Lévy (SURELI), at that time two Master's students whom I co-supervised. Working alongside them on a daily basis allowed me to explore new dimensions of LLM-based research, particularly in estimating an agent's competence (Chapter 8) and generating hypotheses about environmental dynamics (Chapter 5). Furthermore, my collaboration with Guillaume Pourcel (University of Groningen) and Grgur Kovač (INRIA) on the use of LLMs for autotelic agents (Chapter 9) significantly enriched my perspective on intrinsically motivated learning.

I am also grateful to Cédric Colas (MIT) for our in-depth discussions on autotelic agents, his thoughtful feedback on various aspects of this thesis, and his invaluable support throughout our collaboration. In addition, I collaborated with Mohamed Salim Aissi (ISIR) on a project—outside the scope of this manuscript—investigating the sensitivity of fine-tuned LLMs to prompt overfitting. Another collaboration not presented in this thesis took place during an internship at IBM Research, where I worked with Daiki Kimura (IBM), Don Joven Ravoy Agravente (IBM), and Michiaki Tatsubori (IBM) on applying LLMs to dataset-based question answering tasks.

Finally, I would like to express my sincere gratitude to my supervisors, Pierre-Yves Oudeyer (INRIA), Olivier Sigaud (ISIR), and Sylvain Lamprier (LERIA), whose guidance, expertise, and critical feedback were instrumental throughout the development of this work. Their deep knowledge and thoughtful advice have been a constant source of inspiration and support.

Publications

Part of the material of this research has been presented elsewhere in conference papers, journal articles, pre-prints:

- Eager: Asking and answering questions for automatic reward shaping in language-guided RL (NeurIPS 2021) Chapter 12 Carta et al. (2022)
- Grounding large language models in interactive environments with online reinforcement learning (ICML 2023) Chapter 4 Carta et al. (2023)
- Autotelic LLM-based exploration for goal-conditioned RL (NeurIPS 2024-IMOL workshop) Chapter 9 Pourcel et al. (2024a)
- MAGELLAN: Metacognitive predictions of learning progress guide autotelic LLM agents in large goal spaces (ICML 2025) Chapter 8 Gaven et al. (2025)
- WorldLLM: Improving LLMs' world modeling using curiosity-driven theory-making (RLDM 2025) Chapter 5 Levy et al. (2025)
- HERAKLES: Hierarchical Skill Compilation for Open-ended LLM Agents Chapter 13 Carta et al. (2025)

Not discussed here:

- Codeplay: Autotelic learning through collaborative self-play in programming environments (NeurIPS 2023-IMOL workshop) Teodorescu et al. (2023)
- Les algorithmes des IA peuvent-ils comprendre notre monde ? Romac et al. (2024)
- Leveraging Visual Handicaps for Text-Based Reinforcement Learning Chaudhury et al. (2024)
- SAC-GLAM: Improving Online RL for LLM agents with Soft Actor-Critic and Hindsight Relabeling (NeurIPS 2024-IMOL workshop) Gaven et al. (2024)
- Reinforcement Learning for Aligning Large Language Models Agents with Interactive Environments: Quantifying and Mitigating Prompt Overfitting (NAACL-Finding) Aissi et al. (2025b)

Glossary of mathematical symbols and abbreviations

Table 1: Abbreviations

Abbrev.	**Meaning**
ACL	Automatic Curriculum Learning
AGI	Artificial General Intelligence
ALP	Absolute Learning Progress
ASI	Artificial Super Intelligence
AWR	Advantage-Weighted Regression
BC	Behavioral Cloning
DCD	Dual Curriculum Design
DPO	Direct Preference Optimization
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
EK	Expert Knowledge
GCRL	Goal-Conditioned Reinforcement Learning
GPT	Generative Pretrained Transformers
HER	Hindsight Experience Replay
HL	High-level
HRL	Hierarchical Reinforcement Learning
IPO	Implicit Preference Optimization
JEPA	Joint Embedding Predictive Architecture
LAA	Language-Augmented Agent
LC-RL	Language-Conditioned Reinforcement Learning
LL	Low-level
LLM	Large Language Model
LLM-MA	Multi-Agent LLM

Continued on next page

Table 1: Table 1 (continued)

Abbrev.	**Meaning**
LP	Learning Progress
LSTM	Long Short-Term Memory network
MDP	Markov Decision Processes
MLE	Maximum Likelihood Estimation
MLP	Multi-layer perceptron
ND	Narrow trajectory Distribution
NLM	Neural Language Model
NLP	Natural Language Processing
PLR	Prioritised Level Replay
POMPD	Partially Observable Markov Decision Processes
PPO	Proximal Policy Optimization
QG	Question Generation
QA	Question Answering
RAG	Retrieval-Augmented Generation
RL	Reinforcement Learning
RLHF	Reinforcement Learning from Human Feedback
RNN	Recurrent Neural Network
SE	Sample efficiency
SLM	Statistical Language Model
SR	Success rate
TD	Temporal difference
UED	Unsupervised Environment Design
UVFA	Universal Value Function Approximators
VAE	Variational Autoencoders
VLM	Visual Language Model
WD	Wide trajectory Distribution

Table 2: Symbols

Symbol	Meaning
$\mathcal{M}$	A MDP
$\mathcal{S}$	State space of the MDP
s	A state in $\mathcal{S}$
$\mathcal{A}$	Action space of the MDP
a	An action in $\mathcal{A}$
$\mathcal{T}$	Transition function of the MDP
$\mathcal{R}$	Reward function of the MDP
r	A reward
G	Discounted return of the MDP
ρ_0	MDP initial state distribution
$\Delta\mathcal{A}$	Probability distribution over the action space $\mathcal{A}$
π	Policy
$V^\pi(s)$	State value function under policy π for state s
$Q^\pi(s,a)$	Action-value function under policy π for state-action pair (s,a)
Π	The space of policies
π^*	Optimal policy
$\mathcal{G}$	Goal space of the MDP in GCRL
g	A goal from the goal space
$\mathfrak{G}$	Goal generator function
$\mathcal{E}$	Embedding function
$\mathcal{F}_\Lambda$	Filtering function of archive Λ
$V^*(s)$	Optimal value function
$Q^*(s,a)$	Optimal action-value function
$J(\theta)$	Optimization objective of a RL policy gradient algorithm parametreized by θ
$\mathcal{V}$	The vocabulary
Ω	Observation space in a POMPD
O	The observation function mapping a state to an observation
o	An observation
C	Competence function of a policy on a goal
$\mathbb{P}$	Probability
$\mathbb{P}_{LLM}$	Probability computed by the LLM
$\mathbb{LP}_{LLM}$	Log probability of $\mathbb{P}_{LLM}$
$\mathbb{P}^{St}$	Statistician's distribution
$\mathbb{P}^{Sc}$	Scientist's proposal distribution
$\mathcal{I}$	Instruction space
ϕ	Prompting function

Continued on next page

Table 2: Table 2 (continued)

Symbol	**Meaning**
Γ	The set of all trajectories collected
τ	An episode of the MDP
$\eta_{\mathcal{G}}$	A task selector that selects a goal based on Γ
γ	The discount factor
$\mathcal{U}$	Uniform distribution
D	Difficulty of a goal
L	Learnability of a goal
CL	Cumulative learnability
nm_g	Name of the goal g
c	Counting function for n-gram
$\mathbf{Q}$	Query representations in attention mechanism
$\mathbf{K}$	Key representations in attention mechanism
$\mathbf{V}$	Value representations in attention mechanism
$\mathbf{W}_Q$	Query projection matrix for attention mechanism
$\mathbf{W}_K$	Key projection matrix for attention mechanism
$\mathbf{W}_V$	Value projection matrix for attention mechanism
BCE	Binary cross-entropy
Div	Diversity of goals contained in an archive
T_{exp}	The maximum number of experiences done in the environment (WorldLLM)
$\mathcal{H}_y$	The set of containing all the hypotheses
H_y	A set of hypotheses
$\mathcal{H}_{ist}$	The history of the agent
h_c	The completion probability of an auxiliary objective in EAGER
$\mathcal{L}$	Loss function
$\mathcal{L}_{BC}$	Log-likelihood of BC methods
ω^g	The linguistic instruction of goal g
(q, a^*)	Pair of question and expected answer
$\tilde{A}$	A set of possible answers
$\mathcal{Q}$	An active set of questions
L	Number of tokens
$\mathcal{B}$	A buffer
S_c	Crafter score

Part I

Background

Chapter 1

Conceptual background

This thesis is about the use of language to support learning in open-ended agents. Before delving into our contributions, we provide the necessary tools to present our work. We first provide a hig-level perspective on artificial intelligence, then we give some insights on two central concepts: Reinforcement Learning (RL) and language modelling, in particular Large Language Models (LLMs). The first provides the theoretical structure that allows agents to interact with an environment and learn from these interactions. The second has made it possible to harness the specific properties of language, among which a strong generalisation and natural space to develop thoughts, to build better open-ended learning agents. These aspects are presented more deeply in Chapter 2.

1.1 Machines in the world

In order to ground our analysis, we begin by presenting a brief history of AI, particularly in the context of agentic AI, one of the fields of research where reinforcement learning is now a fruitful area that promises to pave the way for open-ended agents.

Symbolic-Connectionist

Before considering how systems endowed with artificial intelligence can be effectively integrated into complex environments, it is important to reflect on the nature of human intelligence and the mechanisms through which a machine can acquire it. In two seminal essays, Turing proposed foundational insights: first, that *machine intelligence* could emerge from an iterative process of generating and evaluating candidate solutions (Turing, 2004); and second, that machines might attain human-like intelligence if they were educated in a manner analogous to human children: *"Instead of trying to produce a program to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education, one would obtain the adult brain."* (Turing, 1950).

These two complementary perspectives on the development of intelligence resonate with two foundational paradigms in artificial intelligence research. The first, known as the symbolic approach, posits that intelligence arises from manipulating abstract symbols,

and that replicating this symbolic system in machines would suffice to achieve intelligent behavior (Newell & Simon, 1976). This school of thought enjoyed early success with systems such as the Logic Theorist (Newell & Simon, 1957), MYCIN (Shortliffe, 1976), and DENDRAL (Buchanan & Feigenbaum, 1978), which demonstrated proficiency in narrow domains such as theorem proving, medical diagnosis, and chemical analysis. Symbolic systems also enabled progress in instruction following (Winograd, 1974), and in strategic games such as checkers (Samuel, 1959). However, this paradigm has been criticized for its labor-intensive knowledge engineering requirements (Felgenbaum, 1977) and its limited capacity for adaptability in dynamic environments (Elkan & Greiner, 1993).

The second paradigm, connectionism, draws inspiration from the neuroscience and psychology work of Thorndike (1933) and Hebb (1949), and models cognition as emerging from networks of interconnected processing units, analogous to neurons. McCulloch & Pitts (1943) introduced the notion of artificial neurons arranged in fixed networks, arguing that intelligence arises from the structure and connectivity of such networks. This idea culminated in the Perceptron (Rosenblatt, 1957), an early neural network capable of simple pattern recognition. The development of the backpropagation algorithm by Rumelhart et al. (1986)—extending Rosenblatt's work—enabled the tuning of network parameters and sparked renewed interest in neural architectures. The combination of algorithmic advances with increases in computational power and access to large datasets laid the foundation for the modern era of *deep learning* (LeCun et al., 2015), a dominant expression of the connectionist approach.

While traditionally seen as competing paradigms, the symbolic and connectionist approaches are now increasingly viewed as complementary. Their integration, giving rise to *neuro-symbolic* methods, enables hybrid architectures that marry the robustness and flexibility of neural models with the interpretability and structure of symbolic systems. These hybrid systems have achieved notable success in complex tasks that demand both abstraction and data-driven learning, including automated theorem proving (Minervini et al., 2018), code synthesis (Parisotto et al., 2017), and visual question answering (Mao et al., 2019), domains which naturally have a strong internal structure but are also so vast and complex that it is impossible for humans to define all of their internal rules by hand.

More recently, the emergence of Large Language Models LLMs—a profound success of the connectionist paradigm—has demonstrated their capacity for symbolic manipulation through purely data-driven learning. LLM-powered Autonomous Agents (LAAs) embody this convergence, performing sophisticated reasoning and decision-making without explicitly programmed rules. Techniques such as ReAct (Yao et al., 2022b) allow these agents to interleave reasoning and acting through structured steps (e.g., *thought*, *action*, *observation*), positioning the LLM as a central planner (Lin et al., 2023a). Besides, studies such as Xiong et al. (2024) show that LAAs can leverage external knowledge structures such as knowledge graphs to approximate human-like reasoning, while effectively scaling with large datasets and using in-context learning without retraining.

This emerging neuro-symbolic synthesis, embodied by LLM-based agents, appears particularly suited to the challenge of open-ended learning. An agent capable of open-ended development must generate and pursue novel artifacts, which inevitably requires moving beyond the

constraints of any predefined symbolic framework. Nevertheless, human intelligence—the best example of open-ended intelligence we have at hand—relies deeply on the manipulation of symbols and formal rules, as exemplified in domains like mathematics. LLM-based agents thus present a compelling model for simulating human cognitive processes. While some researchers, such as Sergey Levine, caution that these systems represent only a projection of human intelligence onto the surface of language (Levine, 2025), their performance suggests a promising path forward for building machines that learn and reason in increasingly human-like ways.

Embodied machines

Open-ended agents must learn skills through experience gathered via continuous interaction between their physical body and their surrounding environment; they are thus inherently *embodied*. Furthermore, to function effectively, such agents must respond in real-time to dynamic environmental changes, qualifying them as *situated*. These characteristics are central to the theory of *embodied cognition*, which posits that cognition is not isolated from the physical body but rather shaped by it. As such, organisms with different bodily structures perceive and conceptualize their environment in fundamentally distinct ways (Lakoff & Johnson, 1981).

In opposition to symbolic artificial intelligence paradigms, Brooks and colleagues advanced the field by designing robots with minimal reliance on internal representations. These robots operated via a *subsumption architecture*, in which sensor inputs directly activated behavior-generating mechanisms. For example, Brooks's robot Allen could perform three basic behaviors: *Avoid* (obstacle avoidance), *Wander* (random exploration), and *Explore* (goal-directed movement). These behaviors were organized into layers that interacted competitively. For instance, while wandering, encountering an obstacle would activate the *Avoid* behavior to prevent collisions. Simultaneously, if a goal was detected, *Explore* could override *Wander* to maintain course. Despite the apparent simplicity of this reactive system, it produced emergent behaviors that appeared purposeful and adaptable. According to Brooks, such systems do not require internal models of the world, since they can "use the world as its own model" (Brooks, 1990).

Nevertheless, this reactive approach has been criticized for its limitations. In particular, Brooks's robots lacked internal memory and world models, making them ill-suited for tasks requiring long-term planning and abstract reasoning (Clark & Grush, 1999). Addressing these concerns, the field of cognitive and developmental robotics emerged, aiming to create embodied agents capable of autonomous decision-making and conceptual learning (Cangelosi & Schlesinger, 2018). Contrary to the purely reactive paradigm, this field investigates how internal representations—such as world models—can emerge from continuous sensorimotor interactions, taking into account the agent's motor capabilities (Gibson, 1979; Weiller et al., 2010). A key research focus lies in understanding how embodied learners develop grounded concepts, rooted in immediate perceptions and actions, and how such concepts enable flexible behavior. For instance, Cangelosi & Riga (2006) presented a simulated robotic model capable of generating novel action concepts through the integration of sensorimotor categories and language learning.

Reinforcement learning (RL), inspired by early work on animal behavior (Thorndike, 1898), offers another framework for developing embodied and situated agents. In RL, agents learn to perform sequences of actions in an environment to maximize cumulative reward, guided by feedback from reward functions that encode task success or failure (Sutton & Barto, 2018c). Through repeated trial-and-error interactions with the environment, agents adapt and refine their behavior. While early RL approaches focused on solving isolated tasks, modern extensions address multi-goal RL, enabling agents to learn across diverse goal distributions (Kaelbling, 1993; Schaul et al., 2015). These goals can be expressed through specific states, visual observations (Nair et al., 2018), or even natural language (Luketina et al., 2019b). With the advent of deep learning, Deep Reinforcement Learning (DRL) emerged as a powerful methodology capable of training agents to tackle increasingly complex challenges. DRL combines neural network function approximation with reinforcement learning algorithms and has achieved remarkable success in domains such as video games (Mnih et al., 2015b; Silver et al., 2017), robotic manipulation (OpenAI et al., 2019), and real-world control systems (Bellemare et al., 2020). These advancements underscore the potential of DRL as a foundational approach for developing open-ended, skill-acquiring artificial agents.

1.2 The Reinforcement Learning framework

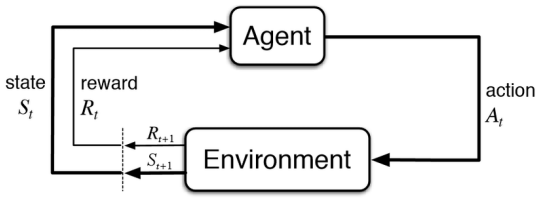


Figure 1.1: The reinforcement learning framework from Sutton & Barto (2018b).

Now that we have briefly reviewed the history of artificial intelligence, and in particular that of artificial agents, we provide a general overview of the main concepts of reinforcement learning.

Markov Decision Processes

This framework describes a sequential decision-making process, where an agent interacts with an environment. At each time step, the agent finds itself in a given state, it takes an action, a reward is then provided by the environment and the state is updated. In this framework, the agent learns to take actions in order to maximize the reward. This process is depicted in Figure 1.1.The RL agent learns through interaction with its environment, aiming to maximize long-term reward. In *episodic* settings, the agent begins at an initial *state* and executes a sequence of actions until that makes it transition to a new state until it reaches a terminal condition—either a maximum step limit or a terminal state—after which the episode resets.

The mathematical framework for formulating the agent learning problem is the *Markov Decision Process* (MDP). An MDP is defined as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0\}$. $\mathcal{S}$ is the state space, with initial state $s_0 \sim \rho_0(\mathcal{S})$ following the *initial state distribution*. $\mathcal{A}$ is the action space and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function mapping state-action pairs to scalars, providing the optimization target for the agent. In some cases, rewards depend only on states: $\mathcal{R} : \mathcal{S} \to \mathbb{R}$. $\mathcal{T}$ is the *transition function* giving the probability distribution over next states: $\mathcal{T}^a_{ss'} = p(s'|s, a)$ for $s' \in \mathcal{S}$, $s \in \mathcal{S}$, and $a \in \mathcal{A}$. The transition function reflects the *Markov assumption*: given the present state and action, the distribution of probabilities over the future state is independent of the past. Formally, at step t:

$$p(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_0, \ldots, s_t, a_t). \tag{1.1}$$

We use *return* to denote the cumulative measure of reward that the agent tries to maximize. In an episode of horizon T, we apply a discount factor to trade off immediate versus future rewards, so the discounted return from time t onward is:

$$G_t = \sum_{i=t}^{T} \gamma^{i-t} \mathcal{R}(s_i, a_i), \tag{1.2}$$

where $\gamma \in [0, 1]$ is the discount factor. When $\gamma = 1$, all rewards are equally weighted (undiscounted case), while $\gamma < 1$ exponentially decreases the importance of future rewards.

Reinforcement Learning algorithms

A *policy* is a function that maps states to actions, either deterministically $\pi : \mathcal{S} \to \mathcal{A}$ or stochastically $\pi : \mathcal{S} \to \Delta\mathcal{A}$, where $\Delta\mathcal{A}$ represents the probability distribution on actions. An agent executes policy π by taking action $a = \pi(s)$ in the deterministic case or sampling $a \sim \pi(\cdot|s)$ when in state s in the stochastic case. Although rewards indicate what is good in the immediate sense, the ultimate goal of RL agents is to learn a policy that maximizes expected long-term return. Thus, many RL algorithms rely on the definition of *value functions* and/or *action-value functions* to evaluate long-term performance.

Given a policy π, we define its associated value function $V^\pi(s)$ as the expected return when starting from state s and following π,

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid s_0 = s \right]. \tag{1.3}$$

It evaluates the quality of being in a given state under a specific policy π.

The action-value function $Q^\pi(s, a)$ is defined as the expected return when starting from state s, taking action a, and following π onward:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid s_0 = s, a_0 = a \right]. \tag{1.4}$$

By developing the expectation in the previous definitions we obtain the relations between the value function and the action-value function (proof in Puterman (2014)):

$$\forall s \in \mathcal{S}, V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q^{\pi}(s,a), \tag{1.5}$$

$$\forall (s,a) \in \mathcal{S}. \times \mathcal{A}, Q^{\pi}(s,a) = \sum_{s'} \mathcal{T}^{a}_{ss'} [\mathcal{R}(s,a,s') + \gamma V^{\pi}(s')]. \tag{1.6}$$

The optimal functions $V^*(s)$ and $Q^*(s,a)$ are defined as the functions that reach the maximum over the space of policies Π:

$$\forall s \in \mathcal{S}, V^{*}(s) = \max_{\pi \in \Pi} V^{\pi}(s), \tag{1.7}$$

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, Q^{*}(s,a) = \max_{\pi \in \Pi} Q^{\pi}(s,a)). \tag{1.8}$$

The value and action-value functions form metrics that measure the quality of the current policy. Thus, the optimal policy $\pi^* \in \Pi$ is defined as the one that maximizes these two functions:

$$\pi = \pi^* \Leftrightarrow V^{\pi} = V^* \Leftrightarrow Q^{\pi} = Q^* \tag{1.9}$$

The size of the state space often prevents a direct calculation of V and Q. Fortunately, these functions satisfy the *Bellman equations*, which decompose recursively into current reward and future discounted value (Bellman, 1966):

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim \mathcal{T}(\cdot|s,a)} [\mathcal{R}(s,a) + \gamma V^{\pi}(s')], \tag{1.10}$$

$$Q^{\pi}(s,a) = \mathbb{E}_{s' \sim \mathcal{T}(\cdot|s,a)} [\mathcal{R}(s,a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')} Q^{\pi}(s',a')]. \tag{1.11}$$

By applying the *Bellman equations* with the optimal policy π^* we get:

$$V^{*}(s) = \max_{a} \mathbb{E}_{s' \sim \mathcal{T}(\cdot|s,a)} [\mathcal{R}(s,a) + \gamma V(s')], \tag{1.12}$$

$$Q^{*}(s,a) = \mathbb{E}_{s' \sim \mathcal{T}(\cdot|s,a)} [\mathcal{R}(s,a) + \gamma \max_{a'} Q^{*}(s',a')]. \tag{1.13}$$

These equations form the backbone of reinforcement learning, as they make it possible to measure the quality of a policy through dynamic programming.

Beyond evaluation, a key consideration in RL is policy improvement. RL algorithms learn the optimal policy using interaction data collected between a policy (for instance, the one of the agent or from a demonstrator) and the environment under a *behavioral policy*. When the *target policy* (being learned) matches the behavioral policy (generating the data) the setting is called *on-policy* learning—the algorithm improves the same policy used for data collection. When these policies differ the setting is called *off-policy* learning—the algorithm learns a different policy than the one that generated the training data.

Another fundamental categorization concerns how agents access the environment dynamics. *Model-based* RL assumes that the agent has access to (or can learn) a model of the environment, enabling planning by simulating state transitions from any given state. In contrast, *model-free* RL improves policies by learning directly from experience without constructing an explicit dynamics model. Model-based methods are analogous to mental simulation or deliberative planning, whereas model-free methods resemble skill acquisition through repeated real-world experience.

One of the most important breakthroughs in RL is the development of Q-learning, an off-policy, model-free algorithm designed to learn the optimal action-value function (Watkins & Dayan, 1992a). In its tabular form, Q-learning maintains a table of Q-values for every state-action pair (s, a) which are initialized arbitrarily. At each time step, the agent selects an action a based on its current Q-values, observes a reward r and the next state s', and then updates the Q-value for the state-action pair (s, a) using the following rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right], \tag{1.14}$$

where α is the learning rate, which determines the extent to which newly acquired information overrides old information. Q-learning is guaranteed to converge to the optimal Q-values, provided that every state-action pair is visited infinitely often (Watkins & Dayan, 1992b).

The Q-learning update can be framed as minimizing the Temporal Difference (TD) error at each step. This is equivalent to minimizing the following loss function:

$$\mathcal{L} = \mathbb{E}_{s,a\sim\rho(\cdot)} \left[\left(\underbrace{r + \gamma \max_{a'} Q(s',a')}_{\text{TD Target}} - Q(s,a) \right)^2 \right], \tag{1.15}$$

where $\rho(\cdot)$ is the distribution in the dataset. This perspective is crucial for extending Q-learning to complex problems where a table is infeasible. DRL extends this idea by using deep neural networks as function approximators to represent value or action-value functions, such as $Q(s,a) \approx Q(s,a;\theta)$, where θ are the network parameters. This allows RL to scale to high-dimensional state and action spaces.

The seminal Deep Q-Network (DQN) algorithm, which achieved superhuman performance on Atari games, introduced two key innovations to stabilize training (Mnih et al., 2015a). Experience replay breaks temporal correlations by storing past transitions in a buffer and training on random mini-batches. A separate target network, which is updated periodically, provides stable TD targets for learning updates, preventing oscillations during training. These two techniques proved highly effective for stabilizing DRL, and many subsequent algorithms have been built upon this foundational work (Van Hasselt et al., 2016; Wang et al., 2016b; Bellemare et al., 2017).

Policy gradient

Rather than directly depending on value functions for generating a policy (for instance by using $\pi(s) = \arg\max_{a\in\mathcal{A}} Q(s,a)$), policy gradient methods directly define a policy as a

continuous function parametrized by θ mapping states to actions $\pi_\theta : \mathcal{S} \to \Delta\mathcal{A}$. Commonly this parametrization is done by a neural network. The objective function is defined as the expectation of the state value function in the initial state s_0:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[V^{\pi_\theta}\right]. \tag{1.16}$$

The objective $J(\theta)$ can be estimated by sampling trajectories from the current policy, and the policy gradient $\nabla_\theta J(\pi_\theta)$ is used to update the parameters via gradient ascent. Policy gradient methods offer several advantages over value-based approaches. First, they naturally accommodate continuous action spaces, avoiding the discretization required by most value-based methods. Second, they often exhibit more stable convergence properties. Finally, they inherently learn stochastic policies, which facilitate exploration and are beneficial in complex environments with multiple near-optimal actions (Lehmann, 2024).

REINFORCE (Williams, 1992) is the first policy gradient method and operates as an on-policy algorithm that relies on Monte Carlo estimation of returns using full episode data to update θ:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t\right]. \tag{1.17}$$

Intuitively, the score function $\nabla_\theta \log \pi_\theta(a_t|s_t)$ indicates how to adjust parameters to increase the probability of action a_t in state s_t, weighted by the return G_t. This naturally increases the likelihood of high-return actions. However, this approach suffers from high variance because the returns sampled across different trajectories can vary substantially.

A common variation of REINFORCE addresses this variance issue by subtracting a *baseline* value from the return G_t while preserving the unbiased nature of the gradient estimator. This leads to the use of the *advantage* function $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, which replaces G_t in the gradient estimate and $V^\pi(s)$ is a baseline function. The advantage function measures how much better or worse an action is compared to the average expected return from that state. Based on this insight, *actor-critic* methods aim to leverage the strengths of both value-based and policy-based approaches. They achieve this by maintaining two distinct components, an "*actor*" (the policy) to choose actions and a "*critic*" (the value function) to evaluate those actions (Mnih et al., 2016). They are often represented by separate neural networks.

In this research, we mainly use Proximal Policy Optimization (PPO), which is an on-policy algorithm under the actor-critic framework (Schulman et al., 2017b). With vanilla actor-critic methods, one can update the policy too aggressively in a single step, causing the new policy to deviate too far from the old one and leading to performance collapses. The key idea of PPO is to impose a constraint on policy updates to ensure stability and prevent drastic changes. The surrogate objetive therefore becomes:

$$J^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right], \tag{1.18}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the likelihood ratio between the new and old policies. This ratio provides a measure of how much the policy has changed between updates. By monitoring

this ratio using the clipping mechanism, PPO ensures that updates are not too drastic, which helps prevent instability in the learning process. PPO also uses multiple epochs of mini-batch updates on the same batch of data, making it more sample-efficient than traditional policy gradient methods.

The other policy gradient method used in this research is *Advantage-Weighted Regression* (AWR) (Peng et al., 2019). AWR assigns one network to regress onto target values for the value function, and another network to regress onto weighted target actions for the policy. The objective function is:

$$J^{AWR} = \mathbb{E}_{s,a\sim\mathcal{D}}\left[\log \pi_\theta(a \mid s)\ \exp\left(\frac{1}{\beta}\left(R^{\mathcal{D}}_{s,a} - V^{\mathcal{D}}(s)\right)\right)\right], \tag{1.19}$$

where $V^{\mathcal{D}}(s)$ is first fit by the entire dataset $\mathcal{D}$ collected from interaction or offline data. The advantage is computed as $A(s,a)^{\mathcal{D}} = R^{\mathcal{D}}_{s,a} - V^{\mathcal{D}}(s)$, and β is a temperature parameter that controls the strength of the advantage weighting. Due to its off-policy nature, AWR can effectively utilize experience data collected from various sources, enabling agents to learn from data generated by different policies such as previous policy versions, exploratory policies, or expert demonstrations to enhance the current behavioral policy.

Goal-conditioned Reinforcement Learning

Human intellects can learn to reach goals one by one, developing skills that can be generalized across different tasks. While agents are able to learn a specific task by optimizing long-term rewards in standard RL, real-world tasks are rarely that straightforward, and goals can be more dynamic and complex. The rise of multi-task learning and advances in RL motivated the emergence of Goal-Conditional Reinforcement Learning (GCRL), where an agent is required to learn to master multiple tasks simultaneously.

The modeling of GCRL augments the standard MDP framework with an additional *goal space* $\mathcal{G}$. The reward function becomes goal-dependent: $\mathcal{R} : (\mathcal{S} \times \mathcal{A} \times \mathcal{G}) \to \mathbb{R}$, meaning that the agent's reward depends not only on the current state and action but also on the specific goal it is pursuing. Similarly, the policy becomes goal-conditioned: $\pi : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$, enabling the agent to select actions based on both its current state and its target goal. This framework allows the agent to generalize across multiple tasks or goals without requiring complete retraining.

Several methods have emerged within the GCRL paradigm. Universal Value Function Approximators (UVFA) (Schaul et al., 2015) extend traditional value functions to handle multiple goals by conditioning on both state and goal, enabling a single network to represent value functions across diverse objectives. Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) addresses the sparse reward problem by treating failed attempts as successful experiences for unintended goals, effectively learning from every trajectory regardless of the original outcome.

Hierarchical Reinforcement Learning

While traditional RL addresses the question *how do I learn to achieve a specific goal*?, agents can struggle in large state-action spaces with long temporal horizons. In such scenarios, the agent can exhibit inefficient exploration behavior, requiring extensive sampling across numerous episodes while receiving sparse or non-informative reward signals. One solution is the "divide and conquer" approach, which Hierarchical Reinforcement Learning (HRL) implements (Sutton et al., 1999; Precup & Sutton, 2000). Instead of asking the agent to reach a goal directly, HRL examines the *temporal* structure of sequential decision-making problems, addressing the complexity by decomposing the problem (goal) into *modular* and *composable* sub-problems (subgoals). Modularity means that the solution to a sub-problem can be reused as a building block, while composability means that these sub-problems can be recombined to solve a wide range of more complex problems.

Formally, this framework extends the standard MDP by introducing a goal space $\mathcal{G}$ and establishing a multi-level decision hierarchy (Klissarov et al., 2025). At the higher level, a meta-policy $\pi_{\text{high}}(g \mid s)$ determines which subgoal $g \in \mathcal{G}$ to pursue given the current state s. This subgoal is then communicated to the lower level, where a goal-conditioned policy $\pi_{\text{low}}(a \mid s, g)$ generates primitive actions to accomplish g. This low-level component directly leverages the GCRL framework introduced earlier. The execution proceeds as follows: once a subgoal is selected, the low-level policy operates for k time steps or until the subgoal is satisfied, generating experience tuples $(s_t, g, a_t, r_t, s_{t+1})$. π_{low} can be learned either by imitation learning (Le et al., 2018; Team et al., 2024) or through RL by getting a reward $r_{\text{low}}(s, g)$ (Bagaria et al., 2021). This second formulation is what we adopted in this manuscript.

Thus, hierarchical decomposition effectively creates multiple coupled MDPs operating at different temporal abstractions. Here we consider two levels: a high-level MDP $(\mathcal{S}, \mathcal{G}, \mathcal{T}_{\text{high}}, \mathcal{R}_{\text{high}}, \gamma_{\text{high}})$ where actions correspond to subgoal selection, and a low-level goal-conditioned MDP $(\mathcal{S} \times \mathcal{G}, \mathcal{A}, \mathcal{T}, \mathcal{R}_{\text{low}}, \gamma_{\text{low}})$ where actions are primitive motor commands. The high-level learns strategic subgoal sequencing while the low-level masters subgoal achievement.

HRL enables more efficient space exploration by allowing agents to pursue achievable subgoals (Klissarov et al., 2025; Machado et al., 2017; Bagaria et al., 2021), promoting diverse exploration through the agent's skill repertoire, and facilitating exploration at higher abstraction levels than primitive actions. Moreover, since subgoals are modular and composable, skills learned at lower levels can be transferred and recombined across different tasks (Konidaris & Barto, 2007; Veeriah et al., 2021; Li et al., 2024a; Wang et al., 2023b). Additionally, it could allow one to identify the origins of undesired behaviors at appropriate abstraction levels (Klissarov et al., 2025; Amodei et al., 2016). However, HRL also presents several challenges. In particular, HRL is a non-stationary framework as the hierarchical agent has to learn, potentially simultaneously, both the low-level policy and the high-level policy (whose optimality depends on the low-level) (Klissarov et al., 2025; Nachum et al., 2018a). Finally, learning a large quantity of behaviours for the low-level policy is costly which, if done naively, can require more interactions than solving the original task directly (Klissarov & Precup, 2021; Barreto et al., 2020).

1.3 Language Modeling

While reinforcement learning has demonstrated remarkable success in sequential decision-making tasks, language modeling presents a distinct yet equally compelling challenge in artificial intelligence. More than a mere tool for information exchange, human languages exhibit remarkable complexity, characterized by their symbolic nature, structured grammar, cultural transmission, compositional properties, and expressiveness.

Language model tasks can be broadly categorized into two types: discriminative tasks and generative tasks. The former focuses on extracting or classifying information from text without producing free-form continuations. Examples include classification tasks such as sentiment analysis and topic detection, and sequence labeling tasks like span extraction and named entity recognition. Generation tasks instead involve predicting subsequent words given a preceding context, with or without additional conditioning inputs. These include open-ended text completion, summarization, translation, and dialogue generation, as seen in Generative Pretrained Transformers (GPT)-based conversational systems (Radford et al., 2018; Achiam et al., 2024). In this section, we first present a short summary of the last decades progress on language modeling, then we expand on the transformer architecture that has deeply impacted this field and that we use in our works.

A Short History of (Recent) Language Models

Language models fundamentally operate as sequential modeling systems, where linguistic sentences are represented as sequences of words or *tokens*—discrete units obtained through tokenization, which decomposes text into smaller, manageable components. The history of language modeling is rich and can be traced back more than a century to Saussure's work on the structure of language (de Saussure, 1983) or Markov's statistical study (Markov, 1913) of the occurrence of vowels and consonants in *Eugene Onegin* (Pushkin, 2010). Although these older works have influenced our current understanding and modeling of language, in this section, we will focus only on more recent modeling methods that are more relevant to the works presented in this thesis.

Developed in the 1990s, Statistical Language Models (SLMs) were addressing contextually relevant properties of natural language from a probabilistic and statistical perspective. They were based on *n-gram* models (Stolcke, 2002; Frederick, 1999; Rosenfeld, 2002), that approximates the probability of a word in a sequence by using the n previous words. For a sequence of k words we have:

$$\mathbb{P}(w_k, w_{k-1}, ..., w_1) \approx \Pi_{j=1}^{k} \mathbb{P}(w_j | w_{j-1}, ..., w_{j-n}). \tag{1.20}$$

The method estimates the probability $\mathbb{P}(w_j | w_{j-1}, ..., w_{j-n})$ by using *Maximum Likelihood Estimation* (MLE).

$$\mathbb{P}(w_j | w_{j-1}, ..., w_{j-n}) = \frac{c(w_j, w_{j-1}, ..., w_{j-n})}{\sum_w c(w_j, w_{j-1}, ..., w_{j-n})}, \tag{1.21}$$

where $c(w_j, w_{j-1}, ..., w_{j-n})$ is a function counting the number of time a n-gram appears in a dataset.

This requires counting the occurrences of subsequences in a training corpus and storing them to compute predictions. However, n-gram models face significant limitations: as n grows, the number of required counts increases exponentially, leading to prohibitive storage demands, which restricts n to small values (typically 2 or 3). Moreover, many n-gram sequences are rare or absent from the training data, causing zero-probability estimates and exacerbating overfitting.

Later, Neural Language Models (NLMs) emerged, utilizing neural networks to estimate word probabilities in sequences through dense vector representations of tokens. NLMs are no longer constrained by the limitations associated with small n values in SLMs. Multilayer perceptrons (MLPs) have been used to implement such methods (Bengio et al., 2003). However, they require learning similar semantic interpretations for all permutations of words in the input, so they are limited to very short contexts and have mainly been used for learning word representations (Mikolov et al., 2013). For modelling longer sequences, they were quickly replaced by sequential models such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) have been implemented to learn distributed representations of words and capture sequential dependencies (Kombrink et al., 2011; Sundermeyer et al., 2012). A remarkable contribution is presented in Radford et al. (2017), where researchers trained a byte-level LSTM model to predict the next byte, demonstrating that the model can learn high-level concepts including sentiment when provided with sufficient training data, model capacity, and training time. This work represents one of the first demonstrations of emergent complex behavior in language models. In the same year, the Transformer architecture was proposed (Vaswani et al., 2017) combining a set predictor model (Santoro et al., 2017) with an attention mechanism (Bahdanau et al., 2015; Kim et al., 2017). Unlike RNN-based structures that process sequences sequentially, transformers consider a set of inputs on which they apply the attention mechanism. All of these processes are permutation-invariant, allowing models to directly access any position in the input sequence, greatly improving both training efficiency (as they enable parallel computation) and the ability to capture long-range dependencies.

The introduction of the Transformer architecture (Vaswani et al., 2017) has fundamentally changed the landscape of natural language processing research. Based on this architecture, *pre-trained models* such as BERT (Devlin et al., 2019), which employs masked language modeling, have demonstrated remarkable performance across a wide range of downstream tasks including question answering and language inference. Subsequently, the Transformer architecture has enabled the development of LLMs such as GPT (Radford et al., 2018, 2019), and their successors, which exhibit emergent capabilities including few-shot learning, reasoning, and complex text generation when scaled to billions of parameters (Wei et al., 2022a). These models have fundamentally transformed our understanding of what is possible with language modeling, moving from narrow task-specific applications to general-purpose language understanding and generation systems. Given that in this thesis, numerous studies use language models based on this architecture, below we provide a detailed explanation of its architecture and mechanisms.

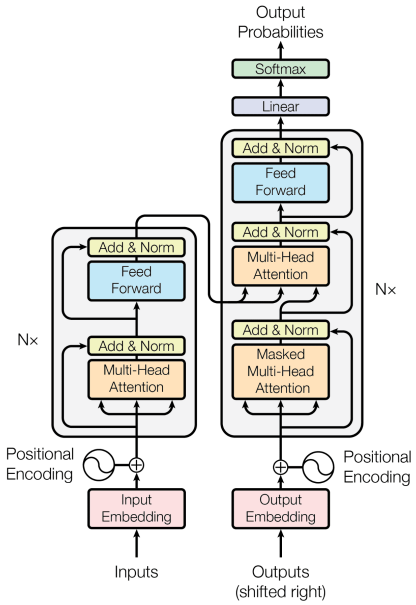


Figure 1.2: Classical illustration of the Transformer architecture from Vaswani et al. (2017). The original design follows an encoder–decoder structure, as shown here, whereas the GPT series retains only the (causal) decoder component.

Transformer

The Transformer is an architecture primarily designed for sequence-to-sequence tasks (Vaswani et al., 2017). As shown in Figure 1.2, it adopts an encoder-decoder structure where both components consist of stacked layers. Each encoder layer contains two sub-layers: a multi-head *self-attention* mechanism followed by a feed-forward network (MLP), with residual connections and layer normalization (Ba et al., 2016) applied around each sub-layer. The decoder shares the same structure but includes an additional third sub-layer that performs multi-head cross-attention over the encoder's output. Furthermore, the decoder's self-attention is masked to prevent attending to future positions, maintaining the auto-regressive property necessary for sequence generation.

The objective of the transformer architecture is to contextualize, through the layers of the model, the representation of tokens in order to refine their meaning in the input context. The core structure and the main innovation of Transformer lie in the design of the multi-head self-attention block. For a single-head attention, denote its input as $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the model dimension. The attention mechanism computes three matrices: queries $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, keys $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, and values $\mathbf{V} = \mathbf{X}\mathbf{W}_V$, where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$ are learned parameter matrices. For each input $\mathbf{x}_i \in \mathbb{R}^d$, we have its corresponding query vector $\mathbf{q}_i = \mathbf{x}_i\mathbf{W}_Q$, key vector $\mathbf{k}_i = \mathbf{x}_i\mathbf{W}_K$, and value vector $\mathbf{v}_i = \mathbf{x}_i\mathbf{W}_V$. The attention score of vector $\mathbf{x}_i$ over $\mathbf{x}_j$ is calculated as $\text{score}(i,j) = \frac{\langle \mathbf{q}_i, \mathbf{k}j \rangle}{\sqrt{d_k}}$, and the attention weights are obtained by applying softmax normalization across all positions, along dimension j. The output for position i is then computed as:

$$\mathbf{o}_i = \sum_{j=1}^{n} \alpha_{i,j} \mathbf{v}_j,$$

where $\alpha_{i,j}$ represents the normalized attention weight between positions i and j, thus $\mathbf{o}_i \in \mathbb{R}^{d_k}$. In the case of multi-head attention, we parallelize the calculation of single-head attention across h heads and concatenate all outputs to formulate the final representation. If the desired output dimension is d and we have h heads, then $d_k = d/h$. The advantage of using multi-head attention is that it allows the model to jointly attend to information from different representation subspaces, enabling it to capture diverse types of relationships such as syntactic dependencies, semantic similarities, and positional patterns simultaneously within a single layer.

From the formulation of the multi-head attention mechanism, one can observe that it is inherently a permutation-equivariant operation, which means that permuting the input sequence will result in the output being permuted in the same way. However, this property is problematic for language modeling, as permuting word positions in a sentence can result in completely different meanings or even ungrammatical constructions. To preserve sequential positional information, the Transformer incorporates positional encoding, which injects positional information into the sequence. The original Transformer uses sinusoidal positional encodings:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right),$$

where pos is the position index, i is the dimension index, and d is the model dimension. These encodings are added to the input embeddings, enabling the model to distinguish between tokens at different positions.

The attention mechanism of the Transformer enables it to capture long-range dependencies effectively, as each position can directly attend to any other position in the sequence, with positional information preserved through positional encoding. Moreover, its parallelizable nature makes it highly efficient for GPU computation, allowing simultaneous processing of all positions. In contrast, traditional RNN-based methods require sequential processing, limiting parallelization efficiency and presenting gradient vanishing or explosion issues.

Reinforcement Learning for Large Language Models

Although language had been incorporated into reinforcement learning (RL) for several years as a means of representing instructions (Mei et al., 2016b; Oh et al., 2017), advances in language modelling itself did not initially depend on RL techniques. The advent of LLMs, however, has considerably deepened the interaction between the two fields. On the one hand, RL has played a pivotal role in improving the alignment, safety, and usability of LLMs (Ouyang et al., 2022a; Ahmadian et al., 2024; Guo et al., 2025); on the other hand, the representational and reasoning capabilities of LLMs have opened promising directions for enhancing decision-making processes in RL. In this thesis, we concentrate on the latter perspective. Nevertheless, given the bidirectional nature of this interplay and its growing relevance, we first provide a concise overview of how RL has been applied to the training and fine-tuning of LLMs.

In the wake of a large body of works interested in reducing the exposure bias, i.e. the divergence between the distribution of the training data and the distribution of the test data

(which stems from the words generated by the model itself) (Ranzato et al., 2016; Goyal et al., 2016; Yu et al., 2017; Che et al., 2017; Scialom et al., 2020), Reinforcement Learning from Human Feedback (RLHF) has emerged as a central paradigm for aligning LLMs with human intentions. While pretrained LLMs are highly proficient at next-token prediction, they are not intrinsically designed to follow instructions or sustain dialogue. A major breakthrough in this direction was *InstructGPT* (Ouyang et al., 2022a), which demonstrated how reinforcement learning could be leveraged to fine-tune models toward producing responses aligned with user requests. The key innovation was to train a reward model from human preference data: annotators ranked model completions by quality, and these rankings served as supervision for the reward model. The LLM was then optimized using PPO (Schulman et al., 2017b), where the learned reward function acted as the optimization objective. This approach was anticipated by Christiano et al. (2017), who proposed preference-based training as a practical alignment strategy. Since then, RLHF has become a cornerstone technique in the development of widely deployed conversational systems, enabling models to be more helpful, safer, and better aligned with human values. Following Ouyang et al. (2022a), numerous refinements have been proposed. Ahmadian et al. (2024) showed that several components of PPO are unnecessary in an RLHF context and advocated for REINFORCE as a more efficient alternative. Direct Preference Optimization (DPO) (Rafailov et al., 2023) and Implicit Preference Optimization (IPO) (Azar et al., 2024) go further by eliminating both reinforcement learning and the explicit reward model, directly learning from human feedback. More recently, Guo et al. (2025) proposed an approximation for advantage computation in PPO, observing that reinforcement learning fosters the emergence of reasoning behaviors in LLMs. Complementarily, Yue et al. (2025) showed that RLHF biases generation toward rewarded trajectories, improving sampling efficiency but at the cost of reduced breadth in reasoning capabilities. Importantly, most of these methods were initially designed for single-turn interactions, where a model generates a response immediately evaluated by feedback.

Chapter 2

Literature overview

Before discussing the body of work presented in this manuscript, this chapter provides an overview of the literature on the role of language as a cognitive tool, exploration and curiosity in AI. We first look at how both humans and artificial agents use language to navigate their environment. We then focus on a fundamental aspect of open-endedness, namely the concept of curiosity-driven exploration and the methods used by humans and artificial agents to achieve it.

2.1 Linguistic agents

Numerous animal species exhibit complex communication systems, including chemical signalling in ants, visual displays in fireflies, and auditory calls in birds and mammals. These systems enable animals to reproduce, locate food sources, or warn of danger. For example, bees perform dances to communicate the location of floral resources, and vervet monkeys emit distinct alarm calls for different predators such as snakes, eagles, or leopards (Moore & Palazzolo, 2024). Yet, despite their functionality, these forms of communication lack the combinatorial, representational, and creative power of human language.

Understanding the emergence of human language requires investigating both its unique features and the cognitive capacities it presupposes. While early hominins such as *Homo erectus* (circa 1 million years ago) had vocal tracts capable of producing simple phonemes (Capasso et al., 2008), and *Homo habilis* (circa 500,000 years ago) showed embryonic development of Broca's area (Balzeau et al., 2014), language in its fully modern form likely emerged only around 100,000 years ago. This gradual evolution is often associated with archaeological evidence of symbolic behaviour, such as figurative art, ceremonial burials, and the construction of dwellings, all indicating the development of recursive and representational thought (Goldberg, 1999; Vyshedskiy, 2019).

Three essential features distinguish human language from other communication systems. First, it exhibits *double articulation* (Martinet, 1960): meaningless phonemes combine into morphemes and words, which can in turn be structured into syntactically governed sentences. Second, human language is inherently *representational*. Words are not mere signals for emotions or immediate reactions; they are arbitrary signs referring to abstract or remote entities. A sentence like "Bob is in the garden" conveys information that is detached from

the speaker's immediate context. Third, language is marked by *creativity*, allowing humans to generate and understand an infinite number of novel utterances. These characteristics culminate in what is often termed *recursive language*—a hallmark of modern linguistic capacity.

Although cognition can occur independently of language—as demonstrated by non-human animals and pre-linguistic children (Sperber et al., 1995; Allen & Bekoff, 1999)—language significantly enhances cognitive capabilities. According to the *constitutive thesis*, language is not merely a vehicle for thought but an essential component of it. Carruthers proposes that language evolved to allow individuals to exchange internal representations, a capability foundational to social cognition and theory of mind (Carruthers, 1998). He argues that language enables the phenomenon of inner speech, paralleling how the mind visualizes images. Clark, on the other hand, advocates for a *supra-communicative view*, wherein language forms an external medium for processing and solving problems (Clark, 1998). In a more radical perspective, Dennett suggests that the human brain is reshaped by its continuous interaction with language (Dennett, 1993), and Leiber likens natural language to a "programming language" that configures cognitive functions (Leiber, 1975). These perspectives align with the *requirement thesis*, positing that certain high-level cognitive abilities depend on the presence of language.

Language also functions as a powerful tool for categorisation. By labelling similar experiences with shared words, humans segment and structure the world into discrete categories (Waxman & Markow, 1995). For instance, associating the term "car" to a range of similar 4 wheels motor vehicles and "air plane" to flying man-made object with wings enables generalization and conceptual blending (e.g. we can easily imagine the category of "flying cars" as cars with wings). Without such linguistic compression, reasoning across infinite specific instances would be cognitively infeasible. The recursive and compositional nature of language allows individuals to generate and comprehend sentences never previously encountered, thus supporting what Chomsky describes as 'linguistic productivity' (Chomsky, 1957). To better understand the emergence of language structure, simulations with humans in laboratories on the emergence of language were conducted, where participants had to create or learn an artificial language (Gómez & Gerken, 2000; Fedzechkina et al., 2012). These experiments simulate the process of language evolution across generations using methods such as Iterated Learning (Kirby et al., 2008) and communication games (Müller & Raviv, 2024). They showed that the need for effective language transmission exerted a pressure that made it simpler and more compositional (Kirby et al., 2008; Raviv et al., 2019). These studies show the balance between learnability and expressivity pressures in shaping language : language must be simple to learn in order to survive and be transmitted to invidiuals across generations (Chater & Vitányi, 2003; Ferrer-i Cancho et al., 2013) while being expressive enough to discriminate different meanings (Skyrms, 2010).

Another significant cognitive contribution of language lies in the formation of relational structures. Through metaphors and analogies, language enables the construction of new mental models. For example, early representations of the atom as 'plum puddings' facilitated the understanding of unfamiliar phenomena through familiar imagery (Hon & Goldstein, 2013). Despite their inaccuracy, such metaphors illustrate how language supports conceptual innovation and abstraction (Hesse, 1988; Lakoff & Johnson, 1981).

The developmental acquisition of these linguistic cognitive tools is well described in Vygotsky's sociocultural theory. Language initially serves as an external communication medium between the child and caregivers, who describe, model, and scaffold the child's experience of the world. This external language is gradually internalised, first as "private speech" directed at the self (thoughts spoken out loud as described by Piaget), and later as fully internal "inner speech" (Lupyan, 2005) (where the language is not audible and occurs in the mind). According to Fernyhough, inner speech manifests in two forms: *expanded inner speech*, which preserves features of dialogue such as syntax and turn-taking, and *condensed inner speech*, which is highly compressed and integrates multiple perspectives simultaneously (Fernyhough, 1996, 2004; Alderson-Day & Fernyhough, 2015).

Language as a Cognitive Tool in AI Agents

As previously discussed, language functions as a cognitive tool in humans by facilitating abstraction, categorisation, and relational reasoning. These same cognitive abilities are increasingly relevant to the development of artificial agents. Language enables artificial agents not only to receive instructions but also to generalise knowledge across novel contexts. For instance, Hill et al. (2020a) demonstrated that reinforcement learning agents conditioned on natural language goals—such as "take the red circle" or "take the blue square"—could generalise to unseen combinations like "take the blue circle." This compositional generalisation, enabled by linguistic structure, was not observed in agents trained without language. Similar findings show that language allows agents to abstract over properties like size or shape (Jiang et al., 2019), model object interactions (Lynch & Sermanet, 2021), and sequence temporally structured goals, such as "open the green door after opening the blue box"(Chevalier-Boisvert et al., 2019). Furthermore, approaches like FiLM (Perez et al., 2017b) illustrate how linguistic inputs can modulate visual representations, altering how a convolutional neural network extracts features. This reflects Dennett's view of language as a transformative input, capable of reshaping an agent's internal representations and cognitive strategies.

Beyond perception and generalisation, language also plays a role in supporting reasoning processes within artificial systems. As argued by proponents of the *constitutive thesis*, language can serve as a medium for thoughts. This idea finds a parallel in LLMs which, when scaled sufficiently, exhibit emergent capabilities for human-like reasoning (Webb et al., 2023). Although zero-shot reasoning may be inconsistent, recent prompting strategies have significantly improved performance. Chain-of-Thought prompting (Wei et al., 2022c), for example, introduces explicit intermediate steps, while instructing models to "think step by step" (Kojima et al., 2022) has been shown to enhance logical consistency. These approaches resemble Carruthers' concept of inner speech, where linguistic scaffolding enables the decomposition and exploration of complex problems (Yao et al., 2023a). In this light, LLMs approximate Clark's idea of language as a cognitive workspace: a practical arena for problem-solving rather than mere representation (Clark, 1998). Recent frameworks like ReAct (Yao et al., 2022b) further exploit this capacity by enabling agents to interleave reasoning and actions through structured steps such as *thought*, *action*, and *observation*, positioning the LLM as a central cognitive planner (Lin et al., 2023a). Moreover, Xiong et al. (2024) shows that language-augmented agents can integrate external symbolic structures,

such as knowledge graphs, and leverage in-context learning to mimic human-like reasoning without additional training.

The process of concept acquisition in AI agents can also mirror developmental mechanisms described by Vygotsky. Initially, an agent engages in dialogue with a social partner, who provides linguistic descriptions of its behaviour and environment. In the IMAGINE framework (Colas et al., 2020b), this interaction allows the agent to acquire a compositional vocabulary, which it later reuses to formulate novel goals. As in human development, the external linguistic input is gradually internalised, giving rise to a form of inner speech that facilitates goal-setting, planning, and flexible problem-solving.

In summary, language serves as a multifaceted cognitive tool not only in humans but also in artificial agents. It supports abstraction, generalisation, categorisation, and reasoning—key faculties for any system striving for open-ended intelligence. The parallels between human and artificial cognitive development are striking: in both, language restructures internal representations (Dennett), enables problem-solving (Clark), and underpins internal reflection (Carruthers). In AI agents, repeated exposure to language enables the emergence of generalisation and planning capacities. When equipped with LLMs and structured prompting strategies, these agents can develop simplified forms of inner speech. Such capabilities are indispensable for open-ended agents, who must not only interact effectively with their environments but also explore, adapt, and generate novel artefacts in response to dynamic conditions.

A world of LLM agents

Following the rapid advancement of large language models (LLMs), agents powered by such models have proliferated, with numerous variants emerging in just a few years (Xi et al., 2025). Although none of these systems can yet be considered truly open-ended, they nonetheless exhibit the core characteristics of agents: the ability to interact with their environment, to plan actions over short- and medium-term horizons, and, to a limited extent, to explore. Their performance often derives from carefully engineered architectures and prompt designs, which constrain them to regions of the environment where human expertise can be effectively leveraged. Nevertheless, these agents are capable of locally emulating traits conducive to open-endedness, whether through environment modelling, goal-space exploration, or rapid acquisition of new capabilities. In this section, we provide a concise review of this emerging and rapidly evolving class of agents.

Similar to Clark's conception of language as a cognitive workspace (Clark, 1998), LLM agents employ language as an integrative medium to combine multimodal perceptions, process them through reasoning patterns, and generate actions. Such agents can be abstracted into three main components: a perceptual block, a cognitive block, and an action module (Xi et al., 2025). Beyond traditional textual inputs, the perceptual block may process visual and auditory information, which is mapped into the language space of the LLM via various techniques. For visual data, many approaches employ captioning to transcribe images into descriptive text (Huang et al., 2019; Pan et al., 2020), where the agent receives textual descriptions of its surroundings.

The cognitive block generally comprises two main facets: memory and reasoning. Memory may be implemented by concatenating prior observations directly into the prompt, although this is limited by the LLM's input length. To overcome this, compression techniques can summarise observation histories, or external databases can serve as persistent memory stores. A prominent family of approaches, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020), retrieves relevant memories based on recency, relevance, or importance (Zhong et al., 2024; Lin et al., 2023b; Hu et al., 2023), enabling the agent to maintain behavioural consistency over time. The reasoning facet builds on the emergent capabilities of LLMs, which can perform zero-shot reasoning on complex tasks. For example, in GLAM, the agent can infer that it should move forward when instructed to approach a ball described as being in front of it. Such capabilities scale with model size (Webb et al., 2023), but can be enhanced through prompting strategies such as Chain-of-Thought reasoning (Wei et al., 2022c; Kojima et al., 2022; Webb et al., 2022), which elicit structured internal reasoning sequences. In addition, LLM agents can decompose high-level tasks into simpler subgoals, akin to human planning (Wang et al., 2023b; Ahn et al., 2022; Gramopadhye & Szafir, 2023). As demonstrated in ReAct (Yao et al., 2023b), agents can also employ reflective steps during which they analyse their latest actions and observations in relation to their objectives in order to adapt their plans (Li et al., 2023a; Qian et al., 2024).

Finally, the action module enables interaction with the environment, most often by issuing natural-language commands that are then translated into executable actions. For instance, the instruction "extend the robotic arm" can be mapped to a sequence of motor control commands. Many LLM agents are also trained to leverage external tools—such as APIs—to expand their operational capabilities, much like humans extend their abilities through tool use. Among the most effective applications of tool use is program synthesis, where an LLM composes multiple simple functions into pipelines capable of achieving complex objectives (Qin et al., 2024; Wang et al., 2023b).

As previously discussed, a defining characteristic of LLM agents is their exceptional capacity for modelling language. This enables them to exchange complex information structures and coordinate effectively with external agents. Consequently, they are well-suited for deployment in multi-agent systems (Guo et al., 2024), where each LLM agent can be assigned distinct roles with specific characteristics, capabilities, and constraints. For example, an LLM multi-agent (LLM-MA) system may simulate a debate, with one agent advocating for a proposal, another opposing it, and several others acting as a jury. Such systems have been employed for societal simulations (Park et al., 2023; Gao et al., 2023) and integrated into games to study the emergence of cooperative and competitive strategies (Wang et al., 2023c; Xu et al., 2024). Research has also explored the incorporation of theory of mind mechanisms to enhance collaboration between agents in gaming environments (Li et al., 2023b; Fan et al., 2024; Lică et al., 2025). The application of LLM-MA frameworks is not limited to modelling social interactions; they are also employed for complex problem-solving tasks. The primary motivation is to leverage the collective capabilities of specialised agents that, while operating individually, can collaborate to solve intricate problems more efficiently. Examples include systems designed for medical diagnosis (Tang et al., 2024c) or for conducting scientific experiments (Zheng et al., 2023), where multiple agents perform complementary tasks such as strategic planning, document retrieval, robotic operation, and data analysis. In these scenarios, agents frequently interact with human collaborators to optimise workflows, such as

in the synthesis of complex materials. While current multi-agent LLM systems do not exhibit true open-endedness, they can contribute meaningfully to open-ended research projects, demonstrating exploratory capabilities within domains constrained by human guidance. Given these strengths, it is reasonable to expect that LLM-based multi-agent systems will increasingly collaborate with humans, particularly in scientific domains, where the inherently open-ended nature of inquiry aligns closely with their operational potential.

In the preceding discussion, we examined the conventional definition of an LLM agent as a combination of a language model and an associated architecture, including the prompt. This naturally raises the question of what, in this context, truly constitutes the agent. One position is that the language model itself forms the agent's core, underpinning all agentive capabilities—both reactive and proactive—while the surrounding engineering merely enables interaction with an otherwise inaccessible world (Andreas, 2022; Wong et al., 2023). In this view, a linguistic model trained on diverse data, capable of representing a broad spectrum of agents, only acquires an operational identity once its profile is specified, determining intrinsic attributes and behavioural patterns. While this question remains open, its implications are significant for the design of open-ended agents. If the essence of the agent lies entirely in the model and its pretraining, then improving performance would require better data (despite the saturation of high-quality text corpora) or architectural innovations to overcome the limitations of next-token prediction. Conversely, if what truly constitutes the agent resides primarily in the prompt and surrounding engineering (how the task is described to the agent, how its responses are templated, how information are extracted from agent answer, ...) , there is a risk of reproducing the shortcomings of symbolic AI, where manual encoding of "open-ended" behaviours yielded limited success (Russell & Wefald, 1991; Guha & Lenat, 1994). The reality likely lies between these extremes: LLMs can be regarded as neuro-symbolic reasoners (Fang et al., 2024; Xiong et al., 2024), whose architecture shapes performance, and where novel approaches—such as Darlow et al. (2025)—may advance language modelling. Nevertheless, integrating such models into broader frameworks for perception and reasoning inevitably requires deliberate engineering, particularly in the way input data are structured. For instance, in the context of RL agents, where they actively collect data that will serve to update either their core model (see Chapter 4) or their current prompt Yao et al. (2023b).

2.2 Curiosity-driven exploration

Supervised and unsupervised learning paradigms are typically constrained to optimising predefined objective functions using static, pre-collected datasets. In contrast, active learning agents autonomously generate and collect data through interaction with their environment, thereby shaping their learning trajectory (Thrun, 1995). This principle is particularly prominent in reinforcement learning frameworks, where the agent's policy determines the data it encounters and subsequently uses for self-improvement (Sutton & Barto, 2018b). Open-ended agents—designed to operate in unconstrained and evolving environments—amplify this paradigm, as the data they collect can vary radically depending on the paths they choose to explore (Jiang et al., 2019). Consequently, the mechanisms of exploration and data acquisition become central challenges in the design of such agents. In this section, we first examine how exploratory behaviours manifest in human development, especially during early

childhood. We then analyse how agents—biological and artificial—construct internal models of their environment to predict the consequences of actions (*forward models*) and to select actions that achieve specific outcomes (*inverse models*). Subsequently, we consider the types of spaces available for exploration and review various exploration strategies that have been proposed in the literature for human and artificial agents.

Exploration in Humans

Human exploration has long been studied through the lens of developmental psychology, particularly in the foundational work of Piaget (Piaget, 1931) and Vygotsky (Vygotsky, 1934). These perspectives view children not as passive recipients of knowledge, but as intrinsically motivated, active learners who construct their understanding of the world through interaction and experimentation. Gopnik (Gopnik, 2020) argues that the unusually long childhood period in humans is an evolutionary adaptation to the exploration-exploitation dilemma. During this phase, children prioritize exploration over exploitation, allowing them to form a rich and flexible model of their environment. Empirical studies support this view: younger children have been shown to infer unusual causal relationships more readily than older children and adults (Lucas et al., 2014), to distinguish true causes from spurious correlations (Gopnik et al., 2001), to infer hidden variables (Schulz & Sommerville, 2006), and to strategically explore to disambiguate mechanisms (Bonawitz et al., 2012; Schulz et al., 2008). They also demonstrate sophisticated inferential strategies, modulating their generalisations based on whether they believe the data was randomly or intentionally sampled (Denison et al., 2013). For a comprehensive overview of these findings, see the review by Schulz (2012).

While the precise cognitive and neural mechanisms that drive human curiosity are not yet fully understood (Gottlieb & Oudeyer, 2018; Gopnik, 2020), several theoretical models have been proposed to account for observed exploratory behaviour. One influential hypothesis posits that humans are intrinsically motivated to maximise their own learning progress (Baldassarre et al., 2014; Oudeyer & Kaplan, 2007b; Ten et al., 2021b), gravitating towards tasks that promise substantial improvements in performance. Other models, grounded in information theory, suggest that exploration is driven by a desire to maximise information gain—leading to more accurate representations of environmental structure (Addyman & Mareschal, 2013; Ruggeri et al., 2024). Empowerment, defined as the ability to maintain control and increase future action possibilities (Salge et al., 2013; Ay & Polani, 2008), has also been shown to guide human behaviour (Brändle et al., 2023). Notably, behaviours aimed at maximising empowerment have been linked to the state of 'flow'—a deeply engaging cognitive state that emerges when an individual's skills are well matched to the demand of a task (neither to easy or difficult) (Melnikoff et al., 2022; Csikszentmihalyi, 1997). Furthermore, humans appear to be drawn to novel stimuli (Ellis, 1950; Smock & Holt, 1962; Poli et al., 2022), a preference that aligns closely with exploration strategies aiming to increase state entropy and diversify experience.

Modelling the environment

Active exploration requires an agent not only to select actions but also to interpret the consequences those actions produce in the environment. This dynamic interaction between

decision and feedback underpins the agent's capacity to learn from its surroundings. Crucially, effective exploration depends on the agent's ability to predict the outcomes of its actions or to identify actions that will bring it closer to the desired environmental state. In this context, the agent's capacity to manipulate or reshape its environment becomes fundamental, as it provides the foundation upon which meaningful exploration—and, by extension, learning—can occur. To efficiently acquire new skills and adapt to novel situations, both humans and animals rely on a *world model*—an internal cognitive framework that enables the anticipation of future environmental states based on current actions (Craik, 1944). This internal model supports planning, reduces uncertainty and enhances decision-making, particularly in complex or unfamiliar scenarios. In artificial intelligence, considerable effort has been devoted to developing analogous mechanisms in artificial agents. By equipping them with learned or structured world models, researchers aim to enable predictive reasoning and adaptive behavior, thereby closing the gap between reactive and goal-directed artificial cognition.

A fundamental form of a world model is the *forward model*, which predicts future states of the environment based on the current state and the agent's chosen action. This concept has long been explored in optimal control theory with model predictive control (Bryson & Ho, 1969) or linear quadratic regulator (Kálmán, 1960), often relying on physics-based simulators. In the 1960s, efforts emerged to optimize continuous action sequences by backpropagating along policy trajectories (Kelley, 1960; Dreyfus, 1962). However, these approaches required environments with analytically defined state transitions and accurately computable inverse derivatives—conditions met in applications such as flight path optimization. Subsequent research introduced approximate forward models (Schmidhuber, 1990; Jordan & Rumelhart, 1992), many of which employed feedforward or recurrent neural networks (Nguyen & Widrow, 1990). In parallel, probabilistic models based on Gaussian processes (Deisenroth & Rasmussen, 2011) or Bayesian neural networks (Gal et al., 2016; Depeweg et al., 2016) demonstrated success in modeling environments with structured and observable states. These models found practical application in frameworks like Dyna (Sutton, 1991), where learned environmental dynamics support offline policy updates without continuous interaction. More recently, several works have trained forward models from state-action trajectories obtained via environment interaction (Ha & Schmidhuber, 2018; Chua et al., 2018; Hafner et al., 2019; Hansen et al., 2022). Nevertheless, such methods typically require direct access to the environment and an explicit reward function. To address these limitations, LeCun et al. (LeCun, 2022) introduced the Joint Embedding Predictive Architecture (JEPA), a self-supervised model that learns world dynamics from unlabeled video data, thereby removing the need for explicit state-action inputs. Building on this framework, V-JEPA (Assran et al., 2025) demonstrates that such models can be repurposed for planning and learning tasks. Additionally, H-JEPA (LeCun, 2022) proposes a hierarchical modeling scheme in which predictive granularity varies according to task complexity, advocating for a single, dynamically reconfigurable world model engine. The DynaLang model (Lin et al., 2024) further extends this paradigm by introducing multimodal conditioning through language, enabling the model to incorporate textual descriptions, instructions, and environmental rules.

Despite the utility of forward models, they exhibit notable limitations when applied to open-ended exploration. Consider the case of a robotic arm: numerous equivalent trajectories may result in a successful grasp, making it challenging to identify novel behaviors or learn diverse skills. Forward models often waste resources evaluating redundant action sequences

that yield similar outcomes, thereby hindering exploration. A more effective alternative is to learn an *inverse model*, which maps desired outcomes to action sequences, thus enabling direct exploration of the goal space and facilitating skill acquisition (Baranes & Oudeyer, 2013). This approach is conceptually aligned with the problem of inverse kinematics in developmental robotics (Vijayakumar et al., 2005; Sigaud et al., 2011), and shares similarities with goal-conditioned reinforcement learning (Kaelbling, 1993), where agents learn to achieve varied objectives by generalizing across goals.

Exploration spaces

In the previous section, we detail how agents could model their environment in order to act more effectively within it. In order to learn one or more models of the world, an agent must be able to collect new data and therefore explore its environment. While a classic RL agent has no choice but to explore the immediate environment in which it acts, more complex setups that favour open-endedness open up other spaces for exploration. In this section we discuss three spaces that an agent can explore:

1. **The action space**, where the agent must decide, in the immediate term, which action will most effectively promote exploration within the current environment.
2. **The goal space**, a higher-level decision space in which the problem shifts from selecting actions to selecting target states to be achieved.
3. **The space of environments**, where the agent—or an external teacher—must select a MDP that will maximize the agent's potential for progress.

In the following section, we focus on the latter three spaces, examining their unique characteristics and the specific challenges they pose for the design of exploration strategies.

The first exploration space for an agent is the *action space*. Each action choice leads to the sampling of a distinct trajectory, thereby enabling the discovery of new states. The objective in this setting is to select actions that yield data useful for improving a forward model, an inverse model, or both. These models can be updated through supervised learning, unsupervised learning, or RL. In this form of exploration, the states that most effectively update the agent's models are those that have never been previously encountered. In the context of RL, this has motivated the design of policies that maximise the discovery of novel states (Burda et al., 2019a) or that drive the agent toward states where its predictive models are most inaccurate. Although this exploration space is arguably the most intuitive for an agent, it presents inherent limitations. Many trajectories may be redundant, converging to the same states, thereby reducing exploration efficiency. Moreover, when the objective shifts toward maximising behavioural diversity or generating novel artefacts, such myopic exploration no longer aligns with the overarching goals. In these cases, an alternative and often more suitable exploration space emerges: the *goal space*.

The second exploration space of interest is the *goal space*. While environmental exploration often focuses on discovering novel states and behaviours, this objective can be reframed as the direct exploration of the space of possible goals. In psychology, goals are defined as

cognitive representations associated with a future outcome that an individual commits to achieving (Elliot & Fryer, 2008). Analogously, in RL, goals can be formalised as a set of constraints on one or more consecutive states that the agent seeks to satisfy (Colas et al., 2022c). These constraints may specify a precise environmental configuration, such as a specific video game frame, or a temporally extended condition, for example, "pick up a red cube after opening a blue door." Representing a goal requires two components: a compact representation that encodes the success criteria (e.g., a vector encoding object positions or a natural language description of a desired behaviour (Hill et al., 2020a; Luketina et al., 2019b)) and a goal-achievement function that determines whether the goal has been reached (Colas et al., 2022c). Language, with its compositionality and capacity for systematic generalisation, has emerged as a promising space for goal representation, being used to shape policies (Harrison et al., 2018; Tasrin et al., 2021), define rewards (Bahdanau et al., 2019; Blukis et al., 2019; Goyal et al., 2020), and set intrinsic goals, as in IMAGINE (Colas et al., 2020a).

In the previous two paragraphs we describe classical setting where an agent is in a MDP possibly augmented with a goals space and can explore both of these spaces. However, such approach is restricted to a MDP. Even classic open-ended RL environments such as Minecraft (Fan et al., 2022) or NetHack (Küttler et al., 2020) remain bound by their predefined parameters, preventing the emergence of entirely novel behaviours, such as learning to cook, that lie outside their design constraints. To develop genuinely open-ended agents capable of discovering an unbounded range of artefacts, it is necessary to extend exploration to the *space of environments* itself (Jiang et al., 2023), in such setting the open-endedness would come from the diversity of MDP to which the agent is confronted (Sigaud et al., 2024). Early research on this problem focused on systems capable of continuous, self-improving exploration. For instance, Schmidhuber (2003) and Schmidhuber (2011) proposed searching the space of solver programs to progressively discover more general problem solvers. Similarly, Clune (2019) advocated for the creation of an AI-generating algorithm (AI-GA) in which increasingly complex problems and agents co-evolve in a rich ecosystem. More recently, adversarial settings have gained traction, where a teacher proposes tasks just beyond the learner's current capabilities. A notable example is POET (Wang et al., 2019), which employs a population of minimax adversaries to generate terrains for a 2D walker. Such approaches have demonstrated improved robustness and adaptability (Portelas et al., 2020d). Another prominent framework is Unsupervised Environment Design (UED) (Dennis et al., 2020; Jiang et al., 2021c,b), which optimises environment parameters—often using a minimax regret objective—to produce new training scenarios. By operating at the meta-level of environment generation, these methods enable exploration beyond the intrinsic limitations of any single environment.

Exploration strategies

Once the exploration spaces have been characterised, the next step is to define strategies for navigating them effectively. Existing approaches can be broadly classified into three main categories. *Undirected exploration* strategies rely on stochasticity or heuristic rules to promote state diversity without explicitly modelling the environment or the agent's knowledge. *Knowledge-based exploration* strategies exploit discrepancies between predicted and observed outcomes to guide exploration toward states that improve the agent's models.

Finally, *competence-based exploration* strategies focus on the agent's ability to achieve goals, directing exploration toward tasks or situations that maximise learning progress. This taxonomy provides a structured framework for analysing and comparing exploration methods across different spaces and agent architectures.

The simplest form of exploration is *random* or *uniform exploration*, which distributes interest evenly across the choice space. It can be effective in simple environments with dense rewards or straightforward, consistent action–outcome mappings (Baranes & Oudeyer, 2013; Benureau & Oudeyer, 2016). However, in complex environments characterised by non-linearity, stochasticity, and redundancy, motor-based undirected exploration often proves insufficient, making goal-based exploration more advantageous (Moulin-Frier & Oudeyer, 2013). For example, in a robotic arm scenario, grasping an object can be achieved through a vast number of action combinations; random action-space exploration is therefore unlikely to yield a wide range of interesting behaviours. Conversely, exploring the *goal space*—in this case, the behavioural repertoire of the arm—can lead to the discovery of a large diversity of potentially useful skills. Mastering specific goals builds competence and environmental control, unlike indiscriminate action sampling (Mirolli & Baldassarre, 2013). Despite its simplicity, random goal exploration has shown surprising efficiency, fostering novelty search and even outperforming directed action-space strategies in some settings (Benureau & Oudeyer, 2016; Colas et al., 2018). Its low computational cost makes it a common component in hybrid strategies, where it is used to gather initial data, function at intermediate levels in hierarchical exploration, or operate within epsilon-greedy frameworks to balance randomness and direction (Forestier et al., 2022; Sutton & Barto, 2018b; Colas et al., 2019). Nevertheless, its effectiveness diminishes in domains where the outcome space has a hierarchical structure that requires prerequisite states to access certain goals—a limitation that can be mitigated through structured goal representations or natural language encodings (Forestier et al., 2022; Colas et al., 2022c).

Knowledge-based exploration encompasses a broad and heterogeneous class of intrinsically motivated strategies, distinguished by the diverse ways in which learning situations are linked to an agent's internal knowledge and the variety of knowledge representations employed. A common approach equips agents with predictive causal forward models of the effects of their actions, coupled with metacognitive monitoring to track outcome prediction errors, thereby directing exploration toward actions with high predictive discrepancy (Saegusa et al., 2008). Relatedly, Pathak et al. (2017b) reward states in proportion to the prediction variance of an ensemble of N world models. Learning progress strategies take a different perspective, favouring actions for which prediction accuracy improves over time (Schmidhuber, 1991a; Oudeyer et al., 2007; Kim et al., 2020), while other methods exploit inverse dynamics models to guide exploration (Pathak et al., 2017b; Haber et al., 2018; Raileanu & Rocktäschel, 2020). A distinct subclass, *count-based* approaches, explicitly represents the frequency of visited states to bias exploration toward under- or over-visited regions of the state space (Bellemare et al., 2016; Tang et al., 2017). Zhang et al. (2021) propose quantifying state familiarity by comparing the agent's representation of a state to that of a randomly initialised agent, scaling the resulting novelty bonus by visitation count. Beyond explicit counting, representation learning techniques such as autoencoders and Variational Autoencoders (VAEs) model the structure of observed states, enabling more abstract measures of interestingness. For instance, Bougie & Ichise (2020) derive intrinsic rewards from reconstruction errors in

context-based autoencoders. Generative models like VAEs (Nair et al., 2018; Pong et al., 2020; Castanet et al., 2025) estimate valid state distributions and latent space statistics, allowing likelihood-based novelty detection; in such frameworks, prediction error naturally decreases with repeated exposure, indirectly reflecting state frequency while also identifying entirely novel states. More recently, several works argue that comparing states solely in terms of representation overlooks their underlying dynamics. FB (Touati & Ollivier, 2021) address this by learning a forward and backward representations of a reward-free MDP as a way to summarize all optimal policies via explicit formulas. More recently LIBERTY (Wang et al., 2023d) and EME (Wang et al., 2024c) tackle this problem by approximating the bisimulation metric, thereby considering both the representation and the value of a state—two states are thus deemed similar if they share not only similar embeddings but also similar behaviour, for example, the distribution of future states under a given policy conditioned on being in state s.

Competence-based exploration strategies assign interestingness according to an agent's level of competence, without necessarily relying on explicit models of world dynamics, which makes them directly applicable to model-free learning systems without auxiliary tasks. They are particularly suited to intrinsically motivated agents pursuing self-imposed goals, as competence naturally corresponds to goal achievement. A simple heuristic in such settings is to sample goals based on the agent's ability to achieve them, as in Bougie & Ichise (2020), where agents are rewarded for perceived incompetence in a given state, thereby seeking actions that maximise incompetence and broaden their repertoire of skills. Florensa et al. (2018) proposed an alternative approach, employing adversarial networks to generate goals that the agent evaluates by their probability of being reached, enabling prioritisation of intermediate-difficulty goals while avoiding those that are trivial or impossible. In multi-task settings with unreachable distractor tasks, Santucci et al. (2013) demonstrated that a competence prediction-error measure was most effective, enabling mastery of all achievable tasks in minimal time. Other research has investigated modular goal spaces, where agents are incentivised to explore regions in which competence is either improving or deteriorating (Stout & Barto, 2010a; Forestier et al., 2022; Colas et al., 2019). Relatedly, Baranes & Oudeyer (2013) and Moulin-Frier et al. (2014) applied competence progress-based strategies to progressively modularised goal spaces, adapting exploration as the structure of the goal space evolved. Complementing these intrinsic strategies, the UED paradigm (**?**) offers a framework in which a teacher varies the parameters of an underspecified environment to generate distributions over environments for a student learner, enabling teaching modes such as domain randomisation, minimax regret, or adversarial environment generation, as in the PAIRED algorithm (**?**). Extensions such as Dual Curriculum Design (DCD) (Jiang et al., 2021b) unify UED with Prioritised Level Replay (PLR) (**?**), allowing the student to learn either from teacher-generated environments (UED) or from strategically selected past experiences (PLR), thus integrating environment design with curriculum learning for more effective exploration.

Part II

Modelling the environment

Chapter 3

Introduction

Open-ended agents are expected to continuously explore their environment and challenge their own capabilities at the frontier of their current competence. This mirrors Vygotsky's concept of the Zone of Proximal Development in developmental psychology, wherein development is driven by tasks situated just beyond the agent's existing abilities (Vygotsky, 1934). It is important to note that these agents can progress even in seemingly static environments, partly thanks to the transformative artefacts they produce, which can fundamentally alter environmental dynamics. For example, once humanity mastered the use of fire, new situations such as cooking meat became possible. To exploit the emergent affordances, agents must internalize new rules and adapt accordingly. An historical example of this necessity is the *aeolipile*—an early version of the steam engine attributed to Heron of Alexandria in the 1st century AD. Although it demonstrated the principles of steam propulsion, its potential remained unrealized for centuries due to a lack of theoretical understanding of thermodynamics (Rosenberg, 2010). This illustrates the importance of representational structures (even partial) that can capture and generalize the underlying dynamics to exploit the new emerging opportunities.

Language, with its inherent capacity for abstraction and generalization, presents a compelling framework for modeling environmental structures and processes. In this context, LLMs, trained on vast corpora, encode extensive prior knowledge about the world and have been increasingly employed to assist in RL tasks (Ahn et al., 2022; Yao et al., 2022b). However, their training paradigm—supervised learning over a static dataset—stands in stark contrast to the open-ended nature of agents that incrementally acquire knowledge through ongoing interaction. As a result, LLMs often exhibit brittle knowledge and fail to adapt to the idiosyncrasies of specific environments. Nevertheless, the general knowledge embedded in LLMs provides valuable heuristics for initial exploration and reasoning. Building on this intuition, we propose two approaches that leverage LLMs to acquire and model environment dynamics in an open-ended manner to adapt them to the specificity of the environment.

In Chapter 4, we introduce GLAM, a gradient-based method wherein an LLM is deployed as an embodied agent and fine-tuned via reinforcement learning. This allows the model to adapt to environmental dynamics and develop what we term "functional grounding," in which symbols acquire meaning through their capacity to predict and influence environmental behavior.

In Chapter 5, we present WorldLLM a theory-driven approach inspired by human inductive reasoning. Here, an agent uses an LLM to generate and iteratively refine hypotheses about the environment based on observed transitions. The progressive refinement of this world model encourages the agent to seek out rare and informative transitions, thereby pushing the boundaries of its current understanding and extending its cognitive reach.

Together, these methods explore complementary pathways toward equipping language-based agents with the ability to learn and adapt within dynamic environments, moving a step closer to the ideal of open-ended intelligence.

Chapter 4

Grounded internalisation of environment rules

4.1 Motivations

The recent rise of Transformer-based 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 & 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., 2023; 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 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 & 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 & 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 & 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).

4.2 Contributions

In this chapter, 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 chapter 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 RL to finetune LLMs for natural language generation tasks (Stiennon et al., 2020; Ouyang et al., 2022b; 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?

4.3 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

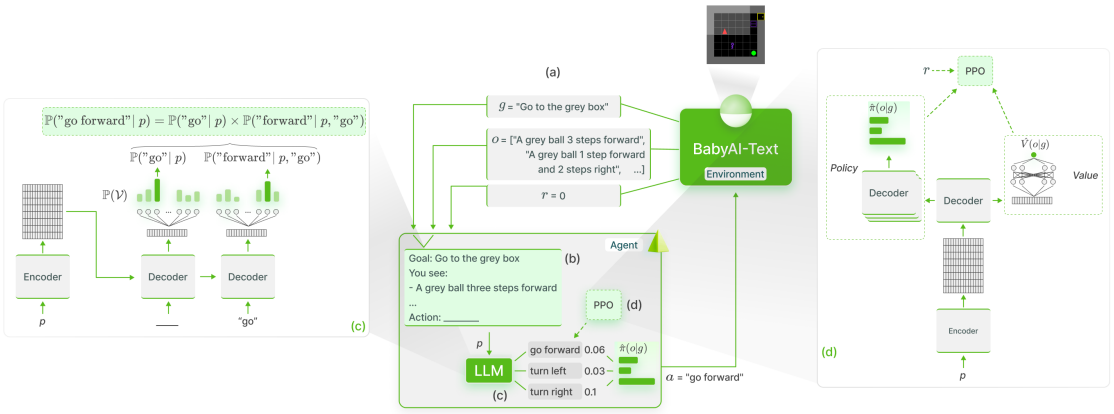


Figure 4.1: **The GLAM method: we use an LLM as agent policy in an interactive textual RL environment (BabyAI-Text) where the LLM is trained to achieve language goals using online RL (PPO), enabling functional grounding**. (a) BabyAI-Text provides a goal description for the current episode as well as 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).

language instruction (Luketina et al., 2019a). 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., 2019; **?**; 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., 2020b) 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 A.2.4 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 5.1).

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., 2023) 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. (2022a) 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 DQN 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., 2022b; Ramamurthy et al., 2022). In this 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., 2017a) to finetune their LLMs using a reward function learned on a dataset of collected human interactions. With this technique, Ouyang et al. (2022b) 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.

4.4 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 5.1 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).

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 $\mathcal{M} = (\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{G}, O, \gamma)$ 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, $O : \mathcal{S} \mapsto \mathcal{V}^N$ 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.1.2). We leverage BabyAI-Text in Section 4.5 to assess our grounding method.

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 A.3 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 4.5.

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. (2022); 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:

$$\mathbb{P}_{LLM}(a_i|p) = \prod_{j=0}^{|a_i|} \mathbb{P}_{LLM}(w_j|p, w_{<j}) \tag{4.1}$$

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 5.1). 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:

$$\mathbb{P}(a_i|p) = \frac{e^{\mathbb{LP}_{LLM}(a_i|p)}}{\sum_{a_j \in \mathcal{A}} e^{\mathbb{LP}_{LLM}(a_j|p)}}. \tag{4.2}$$

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., 2017a) that both learns a policy $\hat{\pi} : O \times \mathcal{G} \mapsto \mathbb{P}(\mathcal{A})$ and a value function $\hat{V} : O \times \mathcal{G} \mapsto \mathbb{R}$ 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 4.4, 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 5.1). 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.

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 A.5 for more details.

4.5 Experiments

BabyAI-Text environment

We design a set of experiments in BabyAI-Text aiming to provide answers for the scientific questions introduced in Section 4.2. 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[1]. 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 4.5 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 this section 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 A.3 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>*

• 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 A.2.3, 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 4.12) on the zero-shot abilities of the resulted trained agents mostly inspired from (**?**) 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 A.7).

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

4.5.1 Q1. How fast can an LLM adapt and learn to solve tasks?

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 4.2. 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 A.3. By plotting the evolution of the distribution over possible actions in Figure A.6, 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 A.3), GFlan-T5 exhibits a drastically better sample efficiency with Symbolic-PPO almost reaching 0.4 after 1.5 million steps. Figure A.6 and Table A.1 highlight how GFlan-T5 leverages its knowledge about the relationships between entities to learn navigation tasks in less than a hundred updates.

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 A.2.3 for further results).

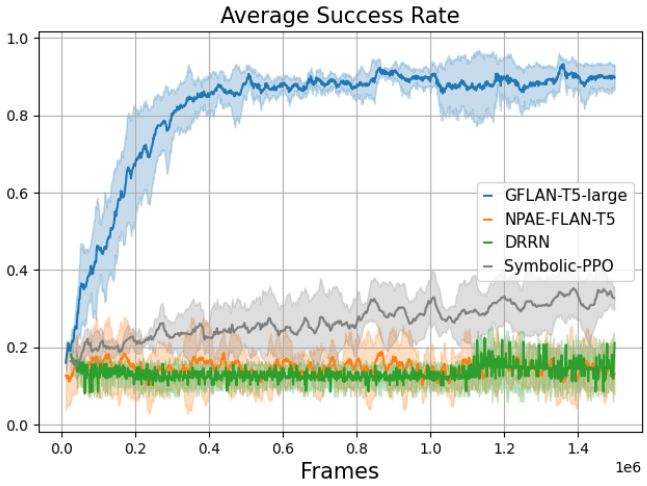


Figure 4.2: **Q1. Sample efficiency**: Evolution over 4 seeds of the average success rate and standard deviation on all Q1 tasks. The details of average success rate calculation over the goals is given in Appendix A.2.2

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 (SE) measure:

$$SE = \frac{1}{T}\sum_{t=0}^{T} SR_t \tag{4.3}$$

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

In order to better understand the impact of the different variables in our experiments, in the rest of this section we conduct experiments on the effect of: pretraining, the agent size, the dimension of the action space, the number of distractor. We also study the robustness of our method to domain specific vocabulary.

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 4.3)

• The full LLM is pretrained and new action heads are used: AFlan-T5 (Figure 4.4)

• 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 4.5)

• 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 4.6)

• All LLM's weights are randomly initialized and action heads are used: **NPA-Flan-T5** (Figure 4.7)

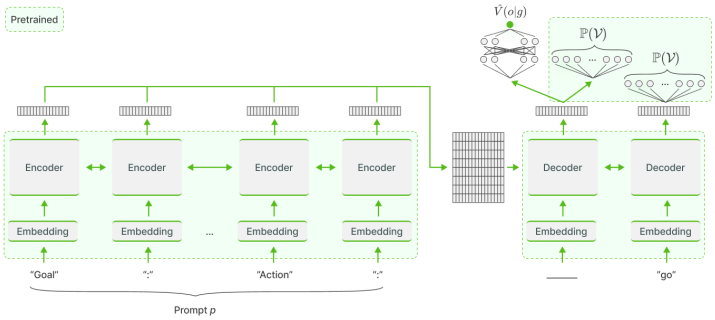


Figure 4.3: GFlan-T5: We use the Flan-T5 architecture and add a value head. We initialize the agent with the pretrained weights (framed in green in the diagram) including its language modeling heads to compute action probabilities. The weights of the value head are initialized randomly. GFlan-T5 stands for: grounded Flan-T5.

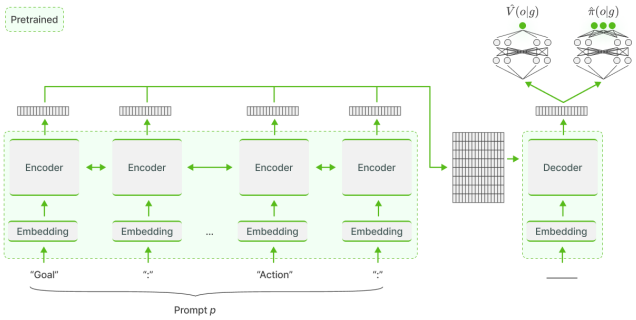


Figure 4.4: AFlan-T5: We use the Flan-T5 architecture but replace the language modeling heads with action heads (that return the probability for each action) and add a value head. We initialize the embedding, the encoder and decoder parts of the agent with the pretrained weights (framed in green in the diagram) and the other weights randomly. AFlan-T5 stands for action heads Flan-T5.

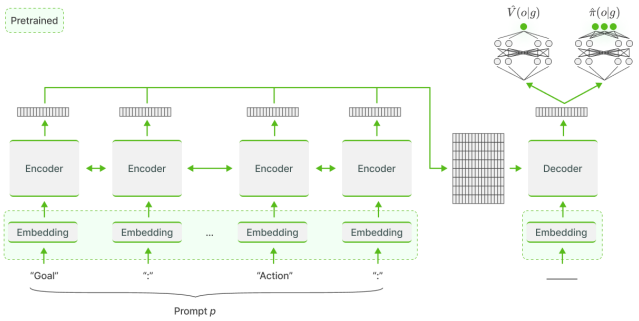


Figure 4.5: NPAE-Flan-T5: We use the Flan-T5 architecture but replace the language modeling heads by action heads and add a value head. We initialize the embedding with the pretrained weights (framed in green in the diagram) and the other weights randomly. NPAE-Flan-T5 stands for: non-pretrained with action heads and pretrained embedding Flan-T5.

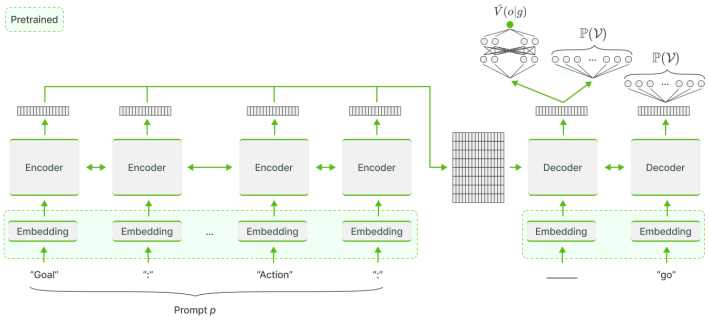


Figure 4.6: NPE-Flan-T5: We use the Flan-T5 architecture and add a value head. We initialize the embedding with the pretrained weights (framed in green in the diagram) and the other weights randomly. NPE-Flan-T5 stands for: non-pretrained with pretrained embedding Flan-T5.

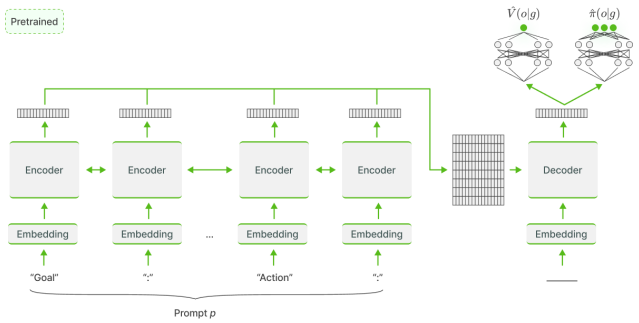


Figure 4.7: NPA-Flan-T5: We use the Flan-T5 architecture but replace the language modeling heads with action heads and add a value head. We initialize all the weights randomly. NPA-Flan-T5 stands for: non-pretrained with action heads Flan-T5.

Figure 4.8 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.

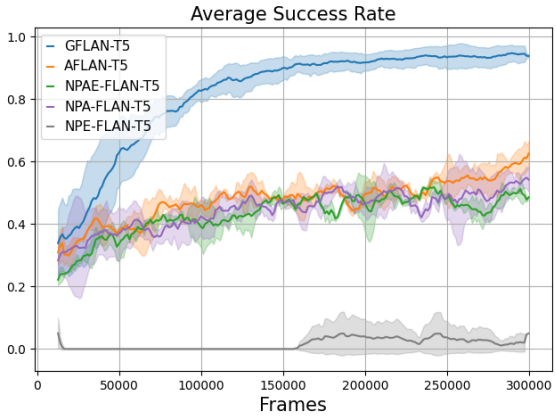


Figure 4.8: Average success rate of varying pretrained weights and scoring method with standard deviation over two random seeds. We train all LLMs on the *Go to <object>* task in 1 room, with 8 distractors, the 6 canonical actions and using Flan-T5 large (780 million parameters) as architecture.

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., 2022b). 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 4.9 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. (2022b): *"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.

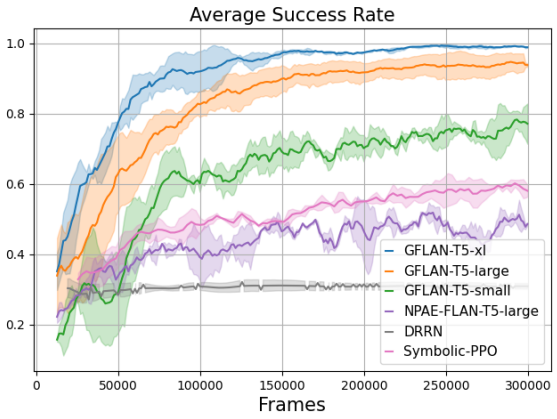
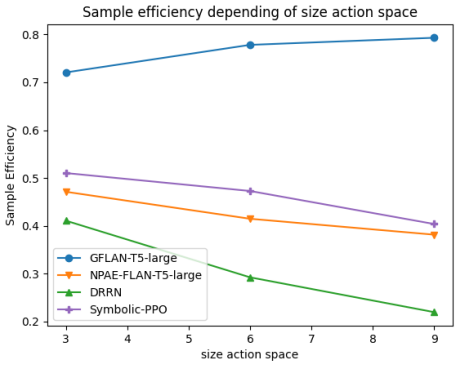
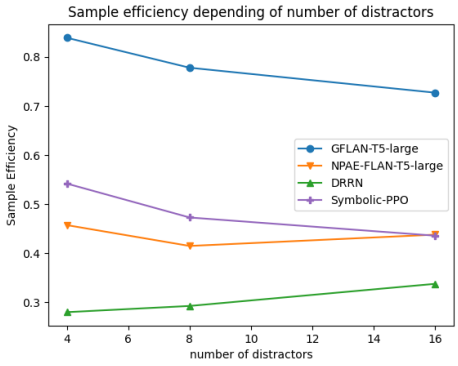


Figure 4.9: Impact of the size of the LLM on online RL finetuning. We conduct the tests with the *Go to <object>* task in 1 room, with 8 distractors. We measure the evolution of average success rate over 2 seeds with standard deviation for GFlan-T5 **small** (80 million parameters), GFlan-T5 **large** (780 million parameters) and GFlan-T5 **XL** (3 billion parameters). DRRN, NPAE-Flan-T5-large and Symbolic-PPO are given as baselines.

Impact of the dimension of the action space



(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 4.10: Impact of the aciton space size and number of distractors on the sample efficiency measure (Equation (4.3)). We report results averaged over 2 seeds for training on the *Go To* task.

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 4.10a. 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.

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 (4.3)) 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 4.10b). 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 show 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 A.2.4 and A.2.4 respectively.

Robustness to domain-specific vocabulary

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 4.11 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.

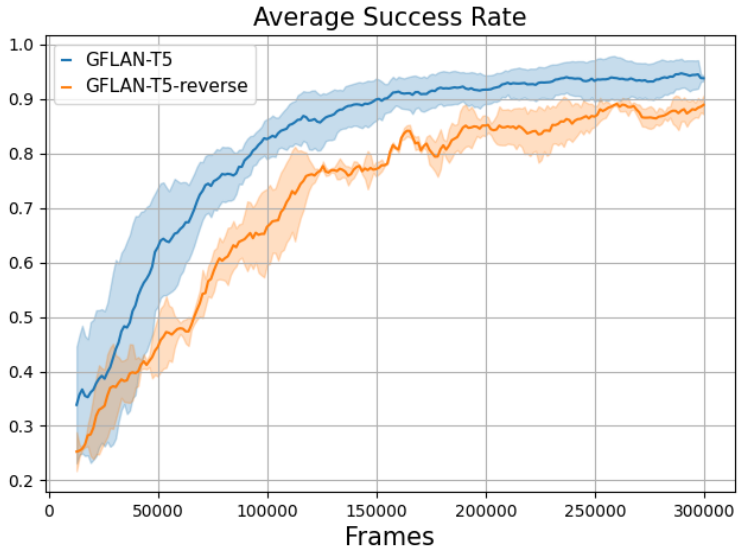


Figure 4.11: Comparison of the average success rate over training for GFlan-T5 on the *Go To* when the actions “turn left” and “turn right” are flipped ("-reverse"). The success rate is given over 2 seeds with standard deviation.

4.5.2 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")[2] and second, an environment with invented objects (made of an invented adjectives and an invented nouns such as *faze dax*).[3] We use the environment the agent has been finetuned on (i.e. without any word substitutions) as a control environment. Results in Figure 4.12 (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

[2]The out-of-vocabulary nouns are given in Appendix A.8.1.
[3]The invented objects are given in Appendix A.8.2.

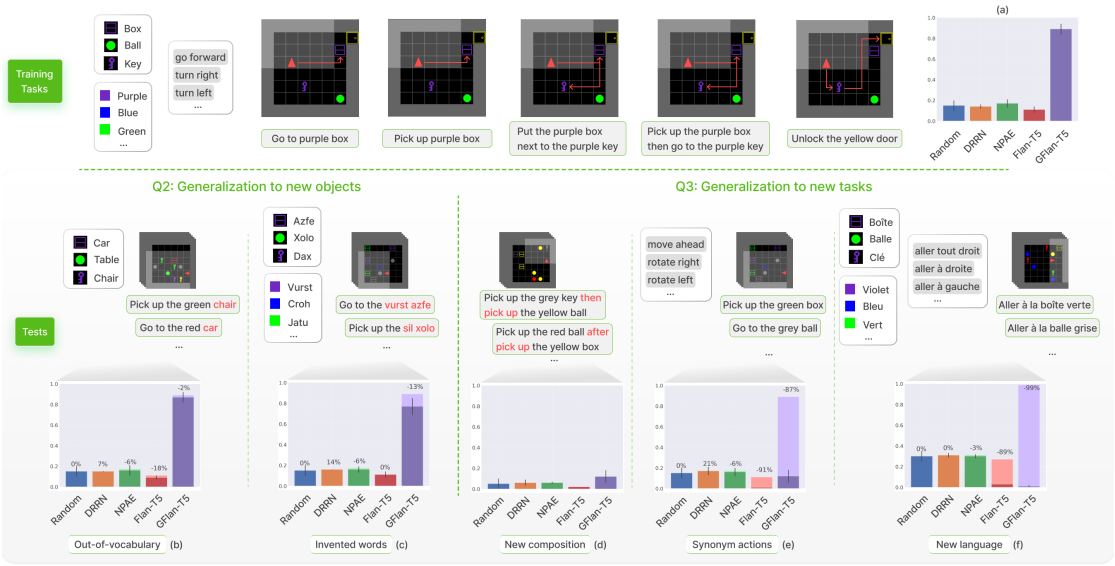


Figure 4.12: **Generalization tests**: We train all agents on a mix of 5 different tasks and evaluate their generalization abilities on 1000 test episodes (also containing a mix of these 5 tasks) (a). We compare them to two baselines: an agent choosing actions randomly (Random) and the zero-shot Flan-T5 (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). The details of average success rate calculation over the goals is given in Appendix A.2.2.

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)[4].

4.5.3 Q3. Generalization to new tasks

In this Section, we perform generalization tests as in Section 4.5.2, 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.

New composition of learned tasks:

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

[4]See Section A.1.2 for more details on the geometry.

Table 4.1: 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	

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 4.12 (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 A.3).

To better observe the dynamics of functional grounding of instructions containing the temporal symbols "then" and "after", we conduct an experiment using 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 4.2 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 A.3 (prompts 8, 9).

Table 4.2: 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	

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"[5]. 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 4.12

[5]A table giving all the used synonyms is given in Appendix A.8.3

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 A.4.2 and A.4.3.

4.5.4 Q4. What is the impact of using RL vs Behavioral Cloning for grounding?

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 A.6.2). 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 4.1, 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).

4.6 Discussion

In this work, 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.

Despite these contributions, the current study is not without limitations. Notably, the experiments were restricted to a purely textual environment, and the method's computational cost presents a barrier to scaling both the action space and the model size. These constraints limited our use to a single environment and relatively small models. However, we believe that addressing these efficiency challenges—potentially through access to greater computational resources—could unlock new opportunities. In particular, it would be possible to extend GLAM to more complex, multimodal environments, potentially leveraging recent advances in large vision-language models (e.g., (Alayrac et al., 2022)) and applying them to robotics scenarios (Lu et al., 2021; Ahn et al., 2022). Additionally, a promising direction for future work would involve studying how functionally grounding an LLM in one environment influences its generalisation abilities, plasticity, and capacity to acquire novel skills in other domains. Ultimately, these results contribute to the broader vision of departing from the *tabula rasa* RL paradigm by demonstrating that pretrained LLMs can serve as a foundation for more sample-efficient, cognitively capable agents.

Since the publication of this work, several follow-up studies have explored similar directions. POAD (Wen et al., 2024b) proposes a fine-grained token-level credit assignment mechanism by decomposing utterances into sub-actions, while ArCHer (Huang et al., 2023) employs a hierarchical RL structure with token-level credit approximation via REINFORCE (Williams, 1992). Other approaches, such as Dynalang (Lin et al., 2024), extend grounding to multimodal settings by training autoregressive models that predict future text and visual observations jointly, thus learning a unified world model from cross-modal experience.

In parallel, our work can be seen as part of a broader effort to align LLMs not just to human preferences—as in RLHF—but to the dynamics of interactive environments. RL-based approaches to LLM training have gained traction, notably with methods like DeepSeek-R1 (Guo et al., 2025), which focus on language-only tasks with static utterance scoring. Inspired by such methods, ReTool (Feng et al., 2025) trains LLMs to operate external tools. Nevertheless, recent critical analysis by (Yue et al., 2025) suggests that while these methods improve sampling efficiency toward correct answers, they may also narrow the model's reasoning diversity, potentially limiting their capacity for creative or novel inference.

While GLAM demonstrates the feasibility of gradient-based functional grounding, this approach is computationally intensive. In the next chapter, we propose an alternative strategy for aligning LLMs with their environment—one inspired by human inductive reasoning. Rather than directly optimizing the model through gradient updates, we explore how LLMs can generate and refine hypotheses about their environment, thereby incrementally improving their internal world models through self-supervised interaction.

Chapter 5

Abstract internalisation of environment rules

5.1 Motivations

LLMs possess broad knowledge about the world, but leveraging this knowledge for precise dynamics modeling remains challenging (Vafa et al., 2024). While LLMs can engage in general reasoning, they struggle to make accurate predictions in specific domains with structured observations and dynamics, such as physics simulations or video games. This limitation stems from the gap between their general capabilities and the need for grounded, domain-specific understanding (Carta et al., 2023; Cross et al., 2025; Bender & Koller, 2020).

The challenge of learning world models has been extensively studied in model-based RL, as depicted in Moerland et al. (2023). Traditional approaches focus on learning forward dynamics models through direct interactions, typically using neural networks to predict next observations given current states and actions. Recent work has shown promise in using LLMs for world modeling by fine-tuning them on embodied experiences (Xiang et al., 2023). However, such an approach is particularly computationally costly given the large amount of interaction data required as well as the cost of gradient-based finetuning of LLMs. Contrary to this, LLMs are known to possess general knowledge that can be adapted using careful prompting (Zhang et al., 2024a).

In a parallel line of work, theory-based RL takes inspiration from humans' inductive reasoning and intuitive theory-making to suggest focusing on inferring structured and compositional world models (e.g. programs) (Tsividis et al., 2021). Current approaches either generate programs to capture environment dynamics directly (Tsividis et al., 2021; Tang et al., 2024a; Li et al., 2024b) or use natural language rules as an intermediate representation (Piriyakulkij et al., 2024; Wang et al., 2024b). Leveraging the world model's structure as well as human-provided prior distributions over possible world models, theory-based methods are usually more sample-efficient than gradient-based ones. More importantly, the resulting world model often shows stronger generalization abilities. However, contrary to gradient-based approaches that usually do not require any expert-given knowledge, theory-based RL methods mostly rely on hand-crafted theory spaces. Finally, one major challenge in learning a world model lies in the data collection. Indeed, collecting transitions that cover the space of possible

outcomes in the environment is key for learning an accurate world model. Notably, both classic model-based RL approach and theory-based RL methods agree on the importance of active data collection focusing on evidence with low likelihood under the current model (Schmidhuber, 1991b; Pathak et al., 2017a; Burda et al., 2019b; Piriyakulkij et al., 2024).

5.2 Contributions

In this chapter, we present WorldLLM, a framework for autonomous improvement of an LLM's world modeling abilities. Our approach combines 1) probabilistic theory induction to produce hypotheses that are given in our LLM's prompt to improve its predictions and 2) curiosity-driven RL to explore the environment and collect transitions poorly predicted with the current hypotheses. Formally, our LLM's world model is the conditional probability $P(s_{t+1}|s_t, a_t, H_y)$, where s_t represents a state, a_t an action, and H_y a set of natural language hypothesized theories. This probability is computed by the LLM by giving it s_t, a_t and H_y in its prompt and taking the probability of s_{t+1} to follow this prompt. Our key insight is that natural language theories can help ground an LLM's broad knowledge into precise predictive power by providing domain-specific rules (Zhang et al., 2024a). Our approach consists of three interacting components: (1) our LLM that computes $P(s_{t+1}|s_t, a_t, H_y)$ by conditioning its predictions on both a state-action pair and the current hypotheses, (2) a theory generator that updates natural language hypotheses using Bayesian inference, and (3) a curiosity-driven reinforcement learning agent trained to collect evidence against the current hypotheses. Inspired by how humans, from children to scientists (Piaget, 1954; Gopnik, 1996), actively update their internal world model by performing experiments, our agent's exploration provides new evidence for hypothesis refinement, creating a virtuous cycle of improvement.

We demonstrate our approach in a video game environment where agents manipulate and combine objects, showing that WorldLLM successfully learns accurate predictive models while generating human-interpretable theories about environment dynamics. This work contributes to a growing body of research on improving LLMs' world modeling capabilities and grounding their knowledge in specific domains (Carta et al., 2023; Xiang et al., 2023; Patel et al., 2020). By combining ideas from theory-based RL, Bayesian inference, and active exploration, we provide a framework for learning structured, interpretable world models that leverage both the broad knowledge of LLMs and domain-specific experiences without any costly gradient-based learning. We examine empirically three aspects of our method answering three open scientific questions.

•**Q1. Environment modeling** How efficiently can WorldLLM model the environment?

•**Q2. Impact of hypothesis generation** What is the empirical impact of hypothesis generation compared to fine-tuning on collected transitions?

5.3 Related work

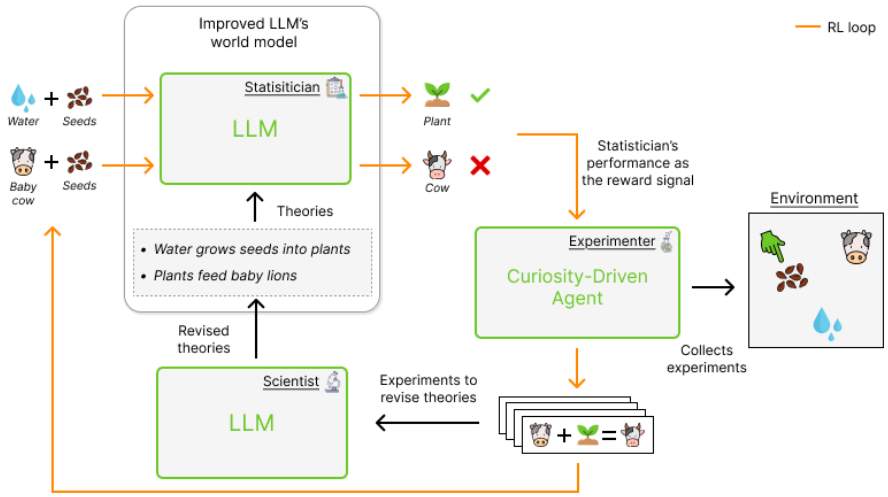


Figure 5.1: Our framework aims at improving an LLM's ability to predict the outcome of state-action pairs by augmenting it with natural language hypotheses about the world (*Statistician*). WorldLLM alternates between generating hypotheses (*Scientist*) and collecting evidence to update these hypotheses (*Experimenter*). The former uses Bayesian inference with an LLM as the proposal distribution. The latter uses curiosity-driven RL to train an agent to collect experiences that have a low likelihood under the *Statistician* with the current set of hypotheses.

Building forward models

Building forward models is a long-established research direction (Ha & Schmidhuber, 2018; Moerland et al., 2023). In addition to producing more interpretable agents, forward models were particularly used in RL to improve sample efficiency allowing the agent to train on imagined transitions or perform multi-step planning (Hafner et al., 2020, 2021, 2025; Schrittwieser et al., 2020). While these works learned the forward model at the same time as the RL policy, a line of work also studied how learning the forward model could be used as an intrinsic reward to foster the policy's exploration (Meyer & Wilson, 1991; Pathak et al., 2017a; Burda et al., 2019a; Oudeyer et al., 2007). WorldLLM employs a similar curiosity-driven scheme where our RL-based Experimenters use reward signals from prior work to promote exploration and collect transitions helping to improve the forward model. However, our method differs in the way the model is improved: instead of using a gradient-based approach to fine-tune the LLM-based forward model, as done in (Xiang et al., 2023), WorldLLM searches for natural language theories that improve the forward model when given in the prompt. While recent work studied how a forward model could leverage external textual information such as a manual explaining the environment (Zhang et al., 2024a; Dainese et al., 2023), our approach iteratively generates and refines its own explanations.

Inductive reasoning for world modeling

The importance of theories in shaping one's representations about the world was argued as central, in both children development and adults (Piaget, 1954; Gopnik, 1996). Building on this, the literature of theory-based RL has been advocating for augmenting artificial agents with inductive reasoning (Tsividis et al., 2021). In particular, (Tsividis et al., 2021; Tang

et al., 2024a; Li et al., 2024b) proposed approaches based on LLMs which generate programs representing the theories about an environment's dynamics. The use of programs allowed the authors to execute and compare how the proposed theories match empirical evidence. Parallel to this, (Piriyakulkij et al., 2024; Wang et al., 2024b) proposed to generate theories using natural language. While the former constrained the generated theories to programs which can be easily evaluated in their environment, the latter used the generated theories to condition a program generator to simulate the environment's dynamics. WorldLLM also builds on the theory-based RL literature and closely relates to (Piriyakulkij et al., 2024). Notably, WorldLLM also leverages Bayesian Inference with an LLM as the proposal distribution to generate the theories. However, the major difference lies in how the produced theories are used: while (Piriyakulkij et al., 2024) constrain the format of theories such that they can be directly tested in the environment, our work focuses on giving these theories in another LLM's prompt to improve its forward modeling capabilities. *In other words, our work strongly differs in its perspective of producing theories that improve our LLM's world modeling abilities instead of producing correct theories.* Nonetheless, the Bayesian Inference framework we used in our Scientist is largely inspired by (Piriyakulkij et al., 2024). Both our work and theirs also consider an active learning paradigm to choose which experiments to perform for theories' refinement. While their approach computes exact Information Gain of all possible experiments, ours relies on curiosity-driven RL agents and Information Gain approximates as rewards to overcome the intractability of exact Information Gain in most environments.

5.4 Improving LLMs' world model with WorldLLM

Problem statement

We consider an environment framed as a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T})$ with $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ the reward function and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ the transition function. In particular, as our main objective is to study world modeling abilities, we consider reward-less environments where no particular task must be solved, i.e. the reward function always returns 0. We focus on textual environments where, given a vocabulary $\mathcal{V}$ and a maximum sentence length N, observations $o \in \mathcal{S} \subset \mathcal{V}^N$ and actions $a \in \mathcal{A} \subset \mathcal{V}^N$ are represented in natural language.

In such a setting, we study how to rapidly learn an LLM-based parametric model $\mathbb{P}^{LLM}$ of the transition function $\mathcal{T}$ (also called forward model) given a budget of experiences T allowed to an agent. This parametric model outputs a probability distribution over all possible next states given a state-action pair. The natural approach from the RL literature to improve this forward model uses gradient-based updates (Ha & Schmidhuber, 2018; Hafner et al., 2020; Schrittwieser et al., 2020). However, the scale of modern LLMs makes such an approach particularly costly. LLMs' natural in-context learning abilities open up an alternative approach: providing natural language hypotheses in the prompt to improve its predictions. This method is less costly, known as more sample efficient (Le Scao & Rush, 2021), and provides human-interpretable updates. Consequently, the parameter we seek to optimize is a set of hypotheses $\mathcal{H}_y \subseteq \mathcal{V}^N$ that, when given as input to the forward model (in addition to state-action pairs), maximizes the likelihood of the experienced next states.

We define the global optimization objective as finding the smallest set of hypothesis $H_y \subseteq \mathcal{H}_y$ that maximizes the forward model's probability of all possible transitions in the environment:

$$\begin{aligned} max \quad & \int_{S\times A\times S} \mathbb{P}^{LLM}(s'|s,a,H_y)\, ds'\, da\, ds \\ min \quad & |H_y| \\ \text{subject to} \quad & H_y \in \mathcal{H}_y \end{aligned} \tag{5.1}$$

As the full transition space is unknown to our agent, it can only use the collected transitions within the budget of T_{exp} experiences. Note that it is essential for the agent to efficiently explore the environment within these T_{exp} experiences, as collecting only similar transitions would lead to hypotheses that predict these transitions well but not the remainder of the transition space. For evaluation, as computing a forward model's performance on the whole transition space is usually intractable, we define a hidden test set $\mathcal{D}_{test}$ containing representative transitions.

WorldLLM

In this section, we present WorldLLM, our methodology for addressing the challenge outlined in the preceding section. Our framework comprises three interconnected components: the Statistician, the Scientist, and the Experimenter. The interactions among these components are illustrated in Figure 5.1. The Statistician represents our LLM-based forward model (i.e. $\mathbb{P}^{LLM}$) and is used to evaluate the current hypotheses. The set of hypotheses H_y the Statistician uses is produced by the Scientist, another LLM. These hypotheses are derived from trajectories collected from interactions with the environment using the Experimenter. WorldLLM alternates for T_{exp} iterations between evidence collection from the Experimenter and hypotheses refinement from the Scientist. In the subsequent sections, we elaborate on how these modules contribute to our objective's optimization.

Statistician

We first detail the Statistician module whose role is to evaluate a set of hypotheses H_y^t on collected transitions. We start by renaming, for the sake of clarity, $\mathbb{P}^{LLM}$ by $\mathbb{P}^{St}$. As described above, the Statistician is a forward model computing the likelihood $\mathbb{P}^{St}(s'|s,a,H_y^t)$ of the next state s' given a state-action pair (s,a) and a set of hypotheses H_y^t. This likelihood is obtained by the LLM by giving it s, a and H_y^t in the prompt and taking the probability of s' to follow this prompt.

Scientist

To improve the Statistician's prediction as a forward model, the Scientist's role is to search through the space of possible sets of hypotheses. To efficiently explore this space, we take inspiration from (Wang et al., 2024b; Piriyakulkij et al., 2024) and use a Bayesian Inference framework. In particular, we rely on another LLM $\mathbb{P}^{Sc}$ as the proposal distribution which generates a candidate set of hypotheses following the distribution $\mathbb{P}^{Sc}(\hat{H}_y^{t,i}|\mathcal{D}_t,\hat{H}_t)$ given

a data set of collected transitions $\mathcal{D}_t$ and the current best hypotheses candidate $\hat{H}_y^{t,i}$. We assume a uniform prior (i.e. no particular prior) over the space of hypotheses. For this study, we chose the Metropolis algorithm, a variant of the Metropolis-Hastings algorithm, which iteratively generates candidate hypotheses for n_steps. The full algorithm is displayed in Algorithm 1 and more details are provided in Appendix B.2.

Algorithm 1 WorldLLM

Input: T total number of iterations, n_{steps} number of Metropolis steps at each iteration, $\mathbb{P}^{St}$ the Statistician's distribution, $\mathbb{P}^{Sc}$ the Scientist's proposal distribution, π the Experimenter
Initialize $H_y^t \leftarrow null$
for $t = 0$ to T_{exp} **do**
  Collect trajectories $\mathcal{D}_t \leftarrow$env(π)
  Evaluate current hypotheses $current_score = \sum_{(s,a,s')\sim\mathcal{D}_t} \log \mathbb{P}^{St}(s'|s,a,H_y^t)$
  Set $\hat{H}_y^t = H_y^t$
  for $i = 0$ to n_{steps} **do** ▷ Metropolis step
    Generate hypotheses $\hat{H}_y^{t,i} \sim \mathbb{P}^{Sc}(\cdot|\mathcal{D}_t, \hat{H}_y^t)$
    Evaluate candidate hypotheses $score = \sum_{(s,a,s')\sim\mathcal{D}_t} \log \mathbb{P}^{St}(s'|s,a,\hat{H}_y^{t,i})$
    Generate a random number $u \sim \mathcal{U}(0,1)$
    if $\log(u) < score - current_score$ **then** ▷ Accept or reject
      Update hypothesis $\hat{H}_y^t = \hat{H}_y^{t,i}$
      Update score $current_score = score$
  Set $H_y^t = \hat{H}_y^t$
  Update Experimenter π

Experimenter

Given a set of hypotheses produced by the Scientist, the Experimenter must collect transitions that will help assessing and refining the hypotheses (i.e. transitions that have a low likelihood under the Statistician). We explore two approaches for our Experiment: oracle agents, which are hard-coded policies that have access to the environment's inner dynamics, and curiosity-driven RL agents trained to collect transitions based on an intrinsic signal.

For the oracles, we propose the following policies:

- A policy that collects the same distribution of transitions as the test set's distribution (O-Ideal)
- An optimal policy that progressively moves from collecting simple transitions to collecting the hardest ones (O-Curriculum)
- An optimal policy that produces the sequence of actions necessary to collect the hardest transitions (O-Hardest)

While O-Hardest will collect complex transitions, it may not cover the full transition space. In comparison, O-Ideal matches the test set distribution. However, the iterative process of WorldLLM may require the Experimenter to focus for several iterations on particular subspaces of the transition space for the Scientist to produce accurate hypotheses. While O-Hardest and O-Ideal lack such a developmental trajectory, O-Curriculum focuses on specific transitions for an arbitrary number of collections. We also study the performance of a policy acting randomly (O-Random), which is unlikely to collect complex transitions.

Regarding our RL policies, we propose three agents trained using curiosity-based intrinsic rewards derived from $\mathbb{P}^{St}$ (Oudeyer & Kaplan, 2007c). First, we introduce *RL-LogP*, which uses the prediction error or surprisal (Burda et al., 2019a; Pathak et al., 2017a) as reward by receiving $r = -log\mathbb{P}^{St}(s'|s, a, H_y^t)$ for each collected transition (s, a, s'). However, this method is known to suffer from the "noisy TV problem" (Burda et al., 2019b) where unpredictable transitions give high rewards.

To address this, we inspired from methods relying on Learning Progress (Oudeyer et al., 2007; Lopes et al., 2012; Schmidhuber, 1991b) or information gain over the prediction error (Oudeyer et al., 2007; Lopes et al., 2012; Schmidhuber, 1991b) that reward transitions on which the forward model improves. We thus propose *RL-ALP*, which uses the absolute value of Learning Progress, allowing it to track both progress and forgetting in the forward model (Oudeyer et al., 2007; Baranes & Oudeyer, 2009):

$$r_{alp} = |\log \mathbb{P}^{St}(s'|s, a, H_y^{t-1}) - \log \mathbb{P}^{St}(s'|s, a, H_y^t)| \tag{5.2}$$

Finally, we also propose *RL-ALPEXP*, a variant of RL-ALP which partitions the transition space into subspaces and computes ALP over each subspace. However, the partitions are not automatically inferred (e.g. as in Baranes & Oudeyer (2013)) but hand-designed. It corresponds to a middle ground between the oracles and RL-LogP or RL-ALP as it has access to expert environment knowledge while still relying on RL. Further details are provided in Appendix B.1.

5.5 Experiments

Environment

To evaluate WorldLLM's efficiency, we use the Playground-Text environment from (Gaven et al., 2024), a textual environment which produces textual observations and receives textual commands. This environment generates a room that contains multiple objects the agent can interact with by grasping or releasing them. The agent can combine two objects, if an interaction is possible, by releasing one onto the other, as illustrated on Figure 5.2 b. For instance, releasing *water* on a *carrot seed* transforms them into *grown carrot*. In our experiments, we only consider 4 types of objects: *Water*, *Plants* (seed or grown) that require *water* to grow, *Small Herbivores* (baby or grown) that require a grown *plant* to grow, and *Big Herbivores* (baby or grown) that require 2 grown *plants* to grow. We evaluate WorldLLM's ability to discover, predict and explain how to grow *Plants*, *Small Herbivores* and *Big Herbivores*. We provide more details on the environment in Appendix B.1. For

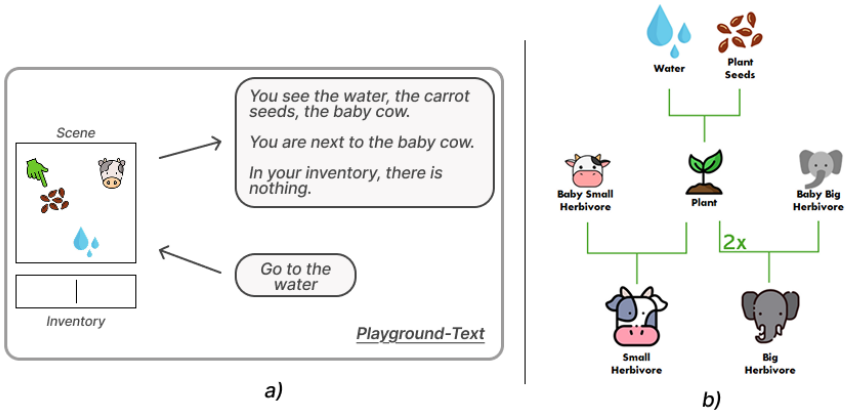


Figure 5.2: *Our experimental setup.* We use the Playground-Text environment which features objects the agent can interact with along with textual observations and actions (a). We focus on 4 object types from Playground-Text: *Water*, *Plants*, *Small Herbivores*, and *Big Herbivores*. We show in (b) the technology tree for combining objects.

analysis, we gather the possible transitions into 6 different types: *Standing* when moving on an object, *Holding 1* when grasping an object, *Holding 2* when grasping two objects at the same time, *Grow Plant* when releasing water on a seed, *Grow S. Herbivore* when releasing a grown plant on a baby small herbivore and *Grow B. Herbivore* when releasing two grown plants on a baby big herbivore. These types are the ones used by RL-ALPEXP.

Optimization

We use Phi-3-mini-4k-Instruct[1] for both the Statistician and the Scientist. For our Experimenter, we either use one of our hard-coded oracles or an RL agent. Regarding oracles, O-Ideal collects transitions that match the distribution of the above test set. O-Hardest performs the optimal policy to grow the *Small Herbivore* then the *Big Herbivore* accessible. O-Curriculum plays the optimal policy to grow a *Plant* up to 133 iterations, then plays the optimal policy to grow the accessible *Small Herbivore* up to 266 iterations, and finally grows the *Small Herbivore* then the *Big Herbivore* until the end. See Appendix B.2 for more details on the oracles. Concerning the RL agent, its policy is a Multi-Layer Perceptron with 2 hidden layers of 64 units. For simplicity, this policy uses a symbolic representation of Playground-Text's state as observation. In all experiments below, unless specified otherwise, we use 8 different random seeds. For each seed, we perform 400 iterations of the framework (i.e. T) where 150 transitions are collected per iteration and $n_{steps} = 5$ steps of the Metropolis algorithm are performed by the Scientist at each iteration. When the Experimenter is an RL agent, we collect 3600 transitions per iteration to train it and use PPO to update the policy, while only keeping the last 150 transitions for the Scientist to match the oracles' hyperparameters. Our test set $\mathcal{D}_{test}$ is composed of 120 *Standing*, 20 *Holding 1*, 7 *Holding 2*, 12 *Grow Plant*, 6 *Grow S. Herbivore* and 3 *Grow B. Herbivore*.

[1] https://huggingface.co/microsoft/Phi-3-mini-4k-instruct

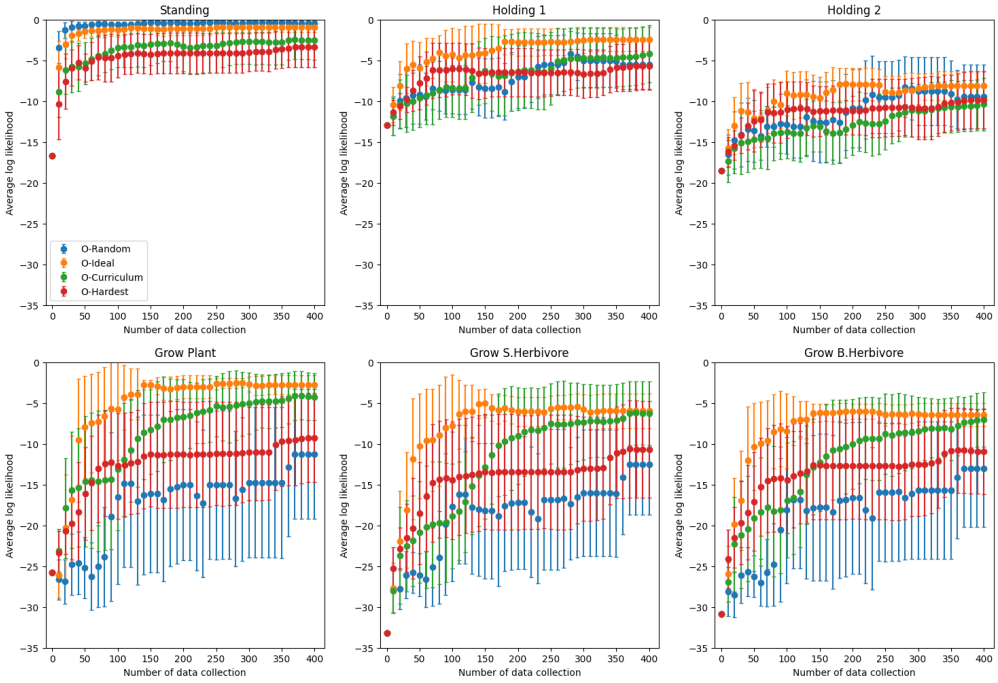


Figure 5.3: Evolution of log-likelihodds computed by the Statistician (using the Scientist's last accepted hypotheses) on the test set throughout WorldLLM's iterations when using different oracles as Experimenters. We average per transition type over 8 random seeds with the standard deviation represented by the error bars. We show at iteration 0 the log-likelihood from the Statistician without any hypotheses given.

5.5.1 Q1. Environment modeling

WorldLLM with an oracle Experimenter

We initiate the experiments by examining the performance of WorldLLM when using our oracles as Experimenters. In Figure 5.3, we show the evolution of log-likelihoods outputted by the Statistician using the last accepted set of hypotheses H_y^t throughout WorldLLM's iterations when using different oracles as Experimenters. These log-likelihoods are computed on the test set and averaged over transition types (and seeds).

One can observe that, regardless of the oracle used or transition type, giving the Statistician hypotheses always improves its performance compared to when no hypotheses are given. When comparing the oracles, one can observe that O-Ideal achieves the best overall performance. This outcome is expected, as the distribution of collected transitions is the same as the one we use for evaluation in the test set. O-Random obtains the poorest performance on complex transitions (i.e. growing an object) while being the best method on *Standing* transitions. This can easily be explained by the fact that moving actions are predominant in the action space. O-Hardest did not lead to strong performance, highlighting the need of diversity in collected transitions for the Scientist. Finally, O-Curriculum obtains results on par with O-Ideal. Results from Appendix B.4.2 indicate that O-Curriculum particularly helps the Scientist generating improved hypotheses.

Overall, these initial experiments demonstrate that WorldLLM is able to produce hypotheses that improve the predictive capabilities of the Statistician's LLM. Moreover, the different performance obtained by our oracles highlighted the importance of 1) diversity in the collected transitions and 2) a developmental collection strategy allowing the Scientist to progressively update the hypotheses.

WorldLLM with a curiosity-driven RL agent as Experimenter

We now move to curiosity-driven RL agents as Experimenters. Compared to the hard coded policies, we rely instead on one of our three intrinsic reward signals (RL-ALPEXP, RL-LogP, and RL-ALP), to collect transitions that drive the Scientist towards generating efficient hypotheses.

As in the previous section, we analyze the evolution of the Statistician's log-likelihoods over the test set. In order to ease the comparison of our three RL-based Experimenters to oracles, we introduce a coverage metric as the area under the evolution of log-likelihoods' curve of each method. We normalize this area using the area between the Statistician's log-likelihood without any hypotheses and the best performance achievable (i.e. 0). The complete formula is

$$\text{area} = 1 - \frac{1}{T_{exp}} \int_0^{T_{exp}} \frac{\log \mathbb{P}^{S_t}(\mathcal{D}_{test}|H_y^t)}{\log \mathbb{P}^{S_t}(\mathcal{D}_{test}|\{\emptyset\})} dt. \tag{5.3}$$

The best performing method possible is expected to be very close to 1 while an inept method's result will be near 0. A method whose result is below 0 means that the hypotheses degrade the initial performance of the Statistician.

Methods	Standing	Holding 1	Holding 2	Grow Plant	Grow S.H.	Grow B.H.
O-Random	0.97 ± 0.01	0.45 ± 0.16	0.40 ± 0.15	0.34 ± 0.24	0.44 ± 0.16	0.40 ± 0.17
O-Ideal	0.92 ± 0.03	0.72 ± 0.12	0.50 ± 0.06	**0.81 ± 0.06**	**0.77 ± 0.06**	**0.74 ± 0.04**
O-Curriculum	0.79 ± 0.15	0.48 ± 0.25	0.31 ± 0.15	0.67 ± 0.15	0.63 ± 0.13	0.60 ± 0.14
O-Hardest	0.74 ± 0.13	0.48 ± 0.20	0.39 ± 0.16	0.52 ± 0.21	0.57 ± 0.18	0.56 ± 0.18
RL-ALPEXP	0.95 ± 0.02	0.44 ± 0.16	0.40 ± 0.14	0.42 ± 0.25	0.52 ± 0.19	0.49 ± 0.19
RL-LogP	0.93 ± 0.01	**0.52 ± 0.07**	**0.53 ± 0.07**	0.62 ± 0.09	0.61 ± 0.07	0.60 ± 0.07
RL-ALP	**0.98 ± 0.01**	0.18 ± 0.08	0.17 ± 0.06	-0.07 ± 0.05	0.16 ± 0.04	0.08 ± 0.07

Table 5.1: This table presents the normalized area under the training curve for each algorithm and transition type. The normalization is performed by dividing the computed area by the area formed by the initial log-likelihoods, i.e. obtained without using hypotheses.

From results shown in Table 5.1, one can observe that RL-ALPEXP is able to leverage its environment knowledge to reach performance significantly better than O-Random but remaining worse O-Ideal. When given no expert knowledge, RL-ALP achieves poor performance while RL-LogP obtains results close or even better than oracle baselines.

To better understand the effect of the RL-based Experimenters on the Statistician's predictions, we studied the evolving distribution of collected transitions. We grouped *Holding 1* and *Holding 2* for simplicity and ignored *Standing* transitions which are predominant by nature. We show these results in Figure 5.4.

They show that RL-LogP manages to explore the environment and cover the whole transition space. The resulting collected distribution is near O-Hardest's one (which is showed in dashed lines). This performance is notably explained by the syntax used by Playground-Text for observations and actions that makes complex transitions naturally harder to predict for an LLM (e.g. moving to a *Big Herbivore* involves only this entity whereas growing a *Big Herbivore* implies an action where two *Plants* are released on a baby *Big Herbivore*). This natural ordering appears in the rewards obtained by the RL policy and favors complex transitions. RL-ALPEXP, which does not leverage such a reward ordering, progressively converges to collecting more complex transitions. When looking at RL-ALP, most of the collected transitions are *Standing* ones. As we have seen in the previous section, the diversity of the transitions given to the Scientist is key in WorldLLM. Consequently, the hypotheses produced with RL-ALP focus solely on explaining *Standing* transitions. A more detailed analysis of why RL-ALP failed is available in Appendix B.4.4.

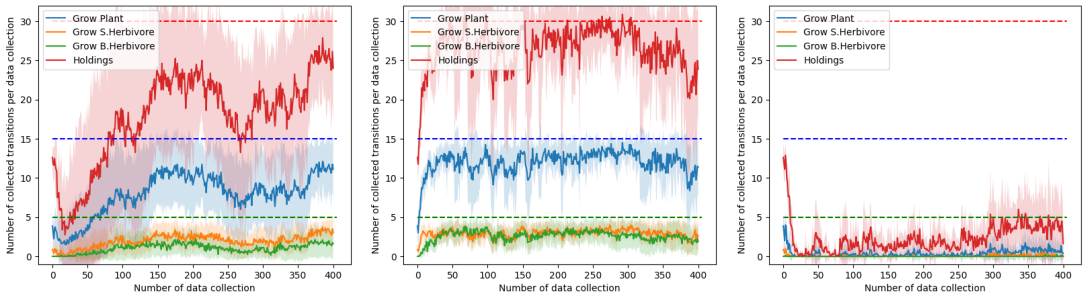


Figure 5.4: Proportion of collected transitions for RL-ALPEXP (left), RL-LogP (middle) and RL-ALP (right). The colored dashed lines correspond to the amount of transitions collected by O-Hardest.

This sections' results show that WorldLLM successfully leads to improvement of the LLM's forward modeling abilities when using curiosity-driven RL-based Experimenters. Moreover, we show that rewarding Experimenters for collecting transitions that help refine the hypotheses not only leads to efficient hypotheses but also leads to policies which solve our environment's full technology tree.

Evaluating the produced world model

We now focus on evaluating the usefulness of the produced world model (i.e. the Statistician equipped with the last accepted hypotheses). While previous sections indicate that WorldLLM succeeds in increasing the likelihood of the observed next states, these results do not guarantee that the observed next states receive the highest likelihood compared to other possible next states. We thus examine our LLM-based world model's capacity to correctly predict (i.e. generate) the next state on our test. In particular, for each state-action

pair in $\mathcal{D}_{test}$, we used our Statistician's LLM and constrained decoding to only generate the valid next states, given the current state, in Playground-Text. We then report in Table 5.2 the percentage of next states that belonged to the top-3 generated sequences.

Methods	Standing	Holding 1	Holding 2	Grow Plant	Grow S.H.	Grow B.H.
No Hypo.	0.82 ± 0.00	**1.00 ± 0.00**	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
O-Random	**1.00 ± 0.00**	0.48 ± 0.26	0.07 ± 0.19	0.18 ± 0.31	0.00 ± 0.00	0.04 ± 0.11
O-Ideal	**1.00 ± 0.00**	0.94 ± 0.15	0.00 ± 0.00	**0.94 ± 0.17**	0.25 ± 0.26	0.46 ± 0.37
O-Curriculum	**1.00 ± 0.00**	0.88 ± 0.31	0.00 ± 0.00	0.85 ± 0.28	**0.60 ± 0.41**	**0.54 ± 0.44**
O-Hardest	**1.00 ± 0.00**	0.86 ± 0.21	0.11 ± 0.28	0.29 ± 0.34	0.27 ± 0.37	0.21 ± 0.37
RL-ALPEXP	**1.00 ± 0.00**	0.71 ± 0.29	**0.34 ± 0.36**	0.40 ± 0.43	0.02 ± 0.06	0.08 ± 0.14
RL-LogP	**1.00 ± 0.00**	0.67 ± 0.30	**0.34 ± 0.25**	0.74 ± 0.32	0.02 ± 0.06	0.29 ± 0.42
RL-ALP	**1.00 ± 0.00**	0.20 ± 0.26	0.00 ± 0.00	0.03 ± 0.08	0.00 ± 0.00	0.00 ± 0.00

Table 5.2: Top 3 performance of Constrained Decoding for each algorithm by transition type. We also report the performance without any hypotheses given (No Hypo.).

The results show that no method performs best on all transition types. However, similarly to what was observed in previous sections, *O-Ideal*, *O-Curriculum*, *RL-ALPEXP* and *RL-LogP* appear to be the best performing Experimenters. More importantly, these results show a significant improvement over the performance without any hypotheses.

5.5.2 Q2. Impact of hypothesis generation

WorldLLM improves an LLM's forward modeling abilities by providing it natural language hypotheses about the world. In this section, we study how our approach compares to a more classic gradient-based approach where the LLM is directly fine-tuned on collected transitions. In addition to avoiding gradient computation through the LLM and being human interpretable, the theories produced by WorldLLM are expected to better generalize to changes in the environment.

We use the transitions collected by RL-LogP and fine-tune our Statistician's LLM on them using a Causal Language Modeling objective (i.e. we maximize the likelihood of the next state's tokens given the state-action pair). We name this new approach *F-LogP*. When evaluated on its top-3 constrained decoding accuracy over the test set as in Section 5.5.1, F-LogP obtains perfect results (i.e. accuracy of 1) over all transition types. In the following sections, we study the generalization abilities of both methods and provide a deeper analysis of their impact on log-likelihoods.

Generalization to changes in the environment

We now assess whether using natural language hypotheses in the prompt produces better generalization abilities for the LLM-based world model than fine-tuning it on collected transitions. For this, we create a generalization environment in which the syntax of observations is changed (see Appendix B.5). We use this new environment to report the top-3 constrained decoding accuracy on our test set as in previous sections.

Methods	Standing	Holding 1	Holding 2	Grow Plant	Grow S.H.	Grow B.H.
No Hypo.	0.28 ± 0.00	**0.85 ± 0.00**	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
O-Random	0.69 ± 0.16	0.45 ± 0.26	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
O-Ideal	0.73 ± 0.23	0.46 ± 0.19	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
O-Hardest	0.33 ± 0.33	0.71 ± 0.33	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
O-Curriculum	0.46 ± 0.34	0.72 ± 0.29	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
RL-ALPEXP	0.65 ± 0.16	0.42 ± 0.30	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
RL-LogP	0.43 ± 0.16	0.30 ± 0.23	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
RL-ALP	0.76 ± 0.18	0.61 ± 0.31	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
F-LogP	**0.88 ± 0.17**	0.24 ± 0.25	0.00 ± 0.00	0.07 ± 0.13	**0.21 ± 0.32**	**0.67 ± 0.44**

Table 5.3: Top 3 performance on Constrained Decoding for each algorithm by transition type on the test set using the generalization environment. We also report the performance without any hypotheses given (No Hypo.).

Results from Table 5.3 show a very important deterioration of the performance for all methods. F-LogP manages to maintain a non-zero accuracy on complex transition types while WorldLLM-based world models do not perform better than the Statistician without any hypotheses. These results invalidate our hypothesis regarding the generalization power of natural language theories. The next section provides additional insights on these results.

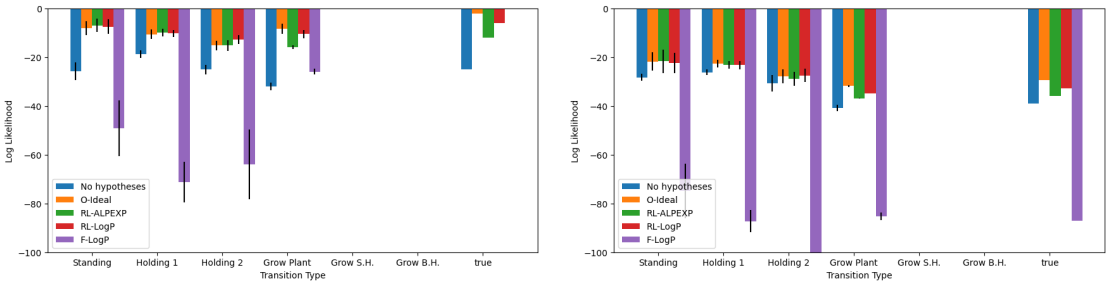
Effect on log-likelihoods

We now propose to study how both methods affect the Statistician's log-likelihoods. We show in Figure 5.5 the log-likelihoods (averaged by transition type) produced by the Statistician when facing an instance of a *Grow Plant* transition.

Looking at results when using the classic environment (a), WorldLLM increases the log-likelihood of all the possible transitions (no matter the Experimenter) while maintaining the correct transition more likely. In comparison, the Causal Language Modeling objective used for F-LogP increases the log-likelihood of the correct transition while significantly decreasing the log-likelihood of all the other transitions. These results illustrate the difference in the two optimization processes: while fine-tuning aims to increase the distance in log-likelihoods between the correct transition and the other ones, WorldLLM focuses on increasing the log-likelihood of the correct transition, regardless of the others.

When focusing on results with the generalization environment, WorldLLM leads to very few changes compared to when no hypotheses are given. On its side, F-LogP produces very low log-likelihoods for all transitions and keeps the correct one slightly above the others. Overall these results show that both approaches generalize poorly.

One explanation for WorldLLM's poor generalization performance lies in the hypotheses produced. We provided examples in Appendix B.4.3 which show that our Scientist notably used examples of transitions in the hypotheses (even though some hypotheses did mention abstract categories or even mentioned objects not in our environment). Such examples no longer align with the generalization environment's syntax. Multiple explanations can be given for this. First, our current experiments rely on a Scientist's LLM with limited capabilities. Second, while our objective in 5.1 minimizes the number of hypotheses, the current implementation of WorldLLM only focuses on maximizing the Statistician's log-likelihood. We argue that favoring short set of hypotheses could help abstract theories emerge. We leave these investigations for future work.



(a) No changes to the environment.

(b) Generalization environment.

Figure 5.5: Log-likelihoods (averaged by transition type) produced by the Statistician when facing an instance of a *Grow Plant* transition. We show how using the generalization environment from Section 5.5.2 affects the log-likelihoods (b).

5.6 Discussion

We propose WorldLLM, a method to improve LLMs' world modeling abilities by generating a set of natural language hypotheses that are given to an LLM when predicting the outcome of a state-action pair. Our method improves the LLM using human-interpretable theories with natural language while being less computationally costly than directly fine-tuning large-scale LLMs over collected data. Our approach improves model performance while preserving human interpretability—an essential property in the context of Artificial Super Intelligence (ASI) as discussed by Edward et al. (2024), where oversight and transparency are critical. WorldLLM is also based on an architecture that does not require any hand-crafted theory space. Our results showed that the adversarial framework between the Scientist and simple curiosity-driven RL Experimenters fuels an exploratory dynamic that covers the environment and allows the discovery of the underlying rules.

Nonetheless, our results show several limitations. First, the inner dynamics of Playground-Text are arguably simple. Moreover, the natural ordering created by the hardest transitions being longer and harder to predict is not expected in more complex environments. This should significantly affect RL-LogP's results. Studying WorldLLM's scaling to more complex environments (e.g. Mars (Tang et al., 2024b)) is therefore an important step. Secondly, our results were performed using a single LLM. This model showed its limitations on Playground-Text, especially in its capacity to generate high-level and abstract hypotheses when used in the Scientist. We believe that using a more efficient LLM in the Scientist in addition to taking into account the hypotheses length (e.g. adding a non-uniform prior in our proposal) should improve WorldLLM's efficiency. Then, we studied three simple curiosity-driven RL Experimenters. Studying more advanced methods such as autotelic approaches (**?**) could improve the quality of the collected transitions, which is key in our framework. Finally, our online Metropolis approach for the Scientist suffers from several limitations. Notably, it only considers the transitions collected at time t, creating potential forgetting when the Experimenter is non-stationary. This choice is motivated by the fact that 1) our LLM used as the proposal has a fixed and limited context window and thus cannot consider all transitions collected up to time t, and 2) computing the likelihood on a large set of transitions is costly. This could be improved by either keeping track of past data (e.g. using a buffer containing transitions from different timesteps) or considering more advanced Bayesian inference methods more suited for online data such as particle filters. The latter would also overcome another current limitation of our approach: a single particle is used and may lead our Scientist to get stuck in local optima (as hinted by Appendix C.4).

Another notable limitation is the disconnection between the evolving world model and the Experimenter's policy learning. In our current setup, hypotheses are used solely to refine the LLM's predictive capacity, without feeding back into the agent's decision-making process. However, in the broader context of open-ended learning, coupling world models with agent policies is essential. Directly integrating hypotheses to inform which transitions are worth exploring could drastically improve sample efficiency, as opposed to relying solely on reward signals to shape behavior.

In conclusion, WorldLLM offers a compelling and interpretable method for bootstrapping LLMs with causal and structural knowledge of their environment. While the current formulation has limitations in terms of scalability, inference efficiency, and integration with agent policies, these challenges also reveal promising directions for future research. The method sets the foundation for more adaptive, grounded, and explainable agents—paving the way toward the development of autonomous systems capable of self-directed, open-ended learning.

Chapter 6

Discussion

In this part, we have introduced two complementary, language-based approaches for enabling agents to model and understand the environments in which they operate. These methods serve as foundational tools in the broader pursuit of building open-ended autonomous systems capable of continual learning and adaptation.

In Chapter 4, we presented GLAM, a method for functionally grounding LLM-based agents by allowing them to interact with their environment through a reinforcement learning (RL) process. We defined *functional grounding* as the process by which internal linguistic symbols acquire meaning through alignment with the relational structure and dynamics of the external environment. Our experiments, conducted in the BabyAI-Text environment—a textual reimplementation of the BabyAI platform—demonstrated that GLAM enables high sample efficiency and facilitates generalization to both unseen objects and novel tasks. This was achieved without requiring environment-specific architectural modifications. However, we also observed that this form of grounding remains constrained in its capacity to capture higher-level regularities and abstract rules, as learning was driven purely by the agent's task-specific reward, without explicit incentives to induce structured knowledge about the environment.

In Chapter 5, we explored a complementary approach through the WorldLLM framework, in which an LLM and a curiosity-driven RL agent interact in an adversarial setting. The RL agent is incentivized to collect transitions that are least predictable under the current hypotheses generated by the LLM, while the LLM, acting as a Scientist, refines its hypotheses via Bayesian inference to account for newly collected transitions. This iterative process facilitates the discovery of underlying environmental dynamics without the need for hand-crafted theory spaces. We empirically compared hypothesis generation to direct fine-tuning on data and found that while WorldLLM successfully uncovers structural patterns, the generated hypotheses are often surface-level, relying on concrete examples rather than abstract generalizations. However, due to the LLM's limited context window and memory capacity, previously useful hypotheses may be forgotten, raising challenges for long-term accumulation of structured knowledge.

Taken together, these two methods represent key ingredients for equipping an open-ended agent with the ability to build, refine, and internalize models of its environment. While each

method has been presented independently, it is likely that open-ended agents will need to integrate both approaches in a synergistic manner. As seen in Chapter 4, functional grounding through RL is effective but may fall short in capturing high-level abstractions, particularly when such abstractions are not directly incentivized by the reward structure. Conversely, explicit hypothesis generation in WorldLLM offers a powerful way to structure and interpret knowledge, albeit at a higher computational cost and with challenges related to memory and scalability. A promising avenue for integration would be to adopt a two-phase strategy: initially leveraging explicit Bayesian hypothesis generation to scaffold understanding of new phenomena, followed by internalization of these hypotheses through RL-driven grounding, thus reducing computational overhead in the long term. This mirrors the human cognitive process of learning, where explicit rules guide initial understanding, eventually giving way to intuitive expertise.

Another promising extension would involve deploying a GLAM-like agent within the WorldLLM framework. Such an agent, equipped with access to existing hypotheses, could strategically select which hypotheses to test through interaction with the environment. This would emulate the scientific method, wherein theoretical insight informs experimental design. In this autotelic setup, the hypothesis space itself could be framed as a goal space, and successful validation or falsification of a hypothesis would correspond to goal achievement. This would bring agents closer to a form of intrinsic scientific reasoning, capable of autonomously expanding and testing their own theory space.

Nonetheless, navigating a rich and highly structured hypothesis space, potentially comprising thousands of subtly distinct propositions, remains a non-trivial challenge. Likewise, exploration in a general goal space is a difficult endeavor, particularly when such a space is unbounded and evolves over time. Designing mechanisms that allow the agent to both select from existing goals and generate novel, meaningful ones will be the focus of the next part.

Part III

Goal space exploration

Chapter 7

Introduction

In the previous part, we demonstrated how language, instantiated through LLMs operating as RL agents, can be employed to explore and model the structure and dynamics of an environment. However, we argue that for an agent to exhibit truly open-ended learning, it is not sufficient to model the environment alone. An open-ended agent must also explore and construct an internal representation of the *goal space*—a conceptual space encoding the diversity of potential outcomes or trajectories that the agent may aim to achieve.

The goal space can be understood as a compressed manifold of all possible behaviors an agent can execute. Effective exploration of this space is a cornerstone of open-ended learning, as it enables the agent to autonomously identify which goals are most conducive to learning progress at any given moment (Oudeyer & Kaplan, 2007a). Much like humans, we expect open-ended agents to exhibit the capacity to generate and select their own goals, a property referred to as *autotelism*. This ability is essential when dealing with complex, high-dimensional, and evolving environments, in which the set of feasible and meaningful goals cannot be exhaustively predefined. Some goals may only emerge as a consequence of prior actions or environmental changes, and would thus remain absent from the initial goal space.

To address the inherent complexity of goal spaces, numerous approaches inspired by human cognition and behavior have been proposed. In particular, methods grounded in intrinsic motivation—internal signals that drive exploration in the absence of extrinsic rewards—have been shown to be effective in facilitating autonomous goal discovery and learning (**?**Colas et al., 2019; **?**). Among these intrinsic signals, Learning Progress (LP) has received considerable attention as a key mechanism underlying goal selection in both artificial agents and human learners (Ten et al., 2021a; Leonard et al., 2023). LP serves as an indicator of cognitive growth, guiding agents to focus on goals that yield the greatest informational or skill-based gains over time.

Language, with its structured and compositional nature, provides a powerful substrate for representing and manipulating goal spaces. Prior work has demonstrated that linguistic representations can facilitate generalization and learning in RL agents (Luketina et al., 2019a). However, such studies have typically been confined to narrow, highly structured domains with constrained language use. In this section, we present two studies that aim to transcend these limitations by leveraging the expressivity and generality of LLMs.

In Chapter 8, we introduce MAGELLAN, a method for estimating learning progress over a broad set of linguistic goals. Building upon the setup from Chapter 4, we deploy an LLM both as an embodied RL agent and as a meta-cognitive module capable of self-assessing its own LP. We demonstrate that this dual functionality enables the model to generalize to previously unseen goals and supports efficient goal selection.

Yet goal selection alone is not sufficient for open-ended behavior. An agent must also possess the capacity to generate novel, previously unconsidered goals that lie within the frontier of its learning capabilities. To this end, in Chapter 9, we propose a preliminary study on AutotelicLLM a method that leverages LP not only to guide selection among known goals but also to generate new, learnable goals. This mechanism allows the agent to continually expand its goal space in a structured and meaningful manner, pushing the boundaries of its competence through self-motivated discovery.

Together, these contributions offer a step toward constructing language-based agents that not only understand and act within their environments but also autonomously explore the ever-evolving space of goals that define purposeful behavior.

Chapter 8

Navigate the goal space

8.1 Motivations

Humans are open-ended learners, continuously exploring and developing new skills through their lifetime. A key mechanism to enable this remarkable capacity is *curiosity-driven learning* — the intrinsic motivation to explore for the sake of learning and discovery (Berlyne, 1954; Kidd & Hayden, 2015). Crucially, humans are *autotelic learners* intrinsically motivated to represent, invent, select and solve their own goals (Colas et al., 2022a). To navigate in a possibly infinite space of goals, without time to explore it exhaustively, they are equipped with intrinsic motivation signals (Baldassarre & Mirolli, 2012; Gottlieb & Oudeyer, 2018). Research on human curiosity has shown the key role of one such intrinsic signal: Learning Progress (LP), i.e. improvement of one's own ability to solve goals (Kaplan & Oudeyer, 2007).

Computational modeling work showed both how it enables efficient automatic curriculum learning (Lopes & Oudeyer, 2012a; Poli et al., 2024b) and how it generates developmental trajectories that simulate key properties in the development of human infants (Oudeyer & Smith, 2016). Recently, several experimental paradigms where humans were free to explore various learning activities confirmed that humans use metacognitive LP monitoring to explore and prioritize goals (Ten et al., 2021c; Leonard et al., 2023; Sayalı et al., 2023; Poli et al., 2024a).

Inspired by open-ended learning in humans and other natural systems, research in AI and artificial life has studied how to build machines with similar capabilities (Schmidhuber, 2013; Jiang et al., 2023; Sigaud et al., 2023). A promising direction focuses on developing autotelic artificial agents that, like humans, can self-generate their learning curriculum by progressively exploring goals with maximum LP (Baranes & Oudeyer, 2013; **?**). This approach efficiently allocates the agent's learning time by avoiding goals that are either too easy or too difficult (Portelas et al., 2020f; Romac et al., 2021), enabling even physical robots to acquire complex skills like tool use in just a few dozen hours (Forestier et al., 2022). However, while these methods show promise in constrained settings, scaling them to open-ended learning remains challenging. The key difficulty lies in efficiently estimating an agent's current competence and expected LP across potentially infinite, evolving, and high-dimensional goal spaces — a fundamental challenge we address in this study.

Parallel to this, a recent breakthrough has emerged in training LLM agents to learn goal-directed behaviors through online interaction with their environment (Carta et al., 2023; Wen et al., 2024a,c). These LLM agents possess a critical capability for open-ended learning: they can leverage the structure of language to generalize effectively, transferring skills learned from practiced goals to semantically similar goals. Using language-instructed agents also offers extensive expressiveness for specifying goals, as required for the open-ended learning setting. This usually induces huge goal spaces, for which prioritization is crucial. However, current approaches leveraging LP for this prioritization fall short at handling such discrete, high-dimensional and structured goal space. They either work only on small low-dimensional goal spaces (Baranes & Oudeyer, 2013; Portelas et al., 2020b; Kanitscheider et al., 2021; Zhang et al., 2024b) or rely on expert-defined goal groupings to reduce the number of goals (Colas et al., 2019; Akakzia et al., 2021; Kumar et al., 2024). In particular, none of them are able to capture the semantic relationships between goals to efficiently estimate an LLM agent's generalization abilities.

8.2 Contributions

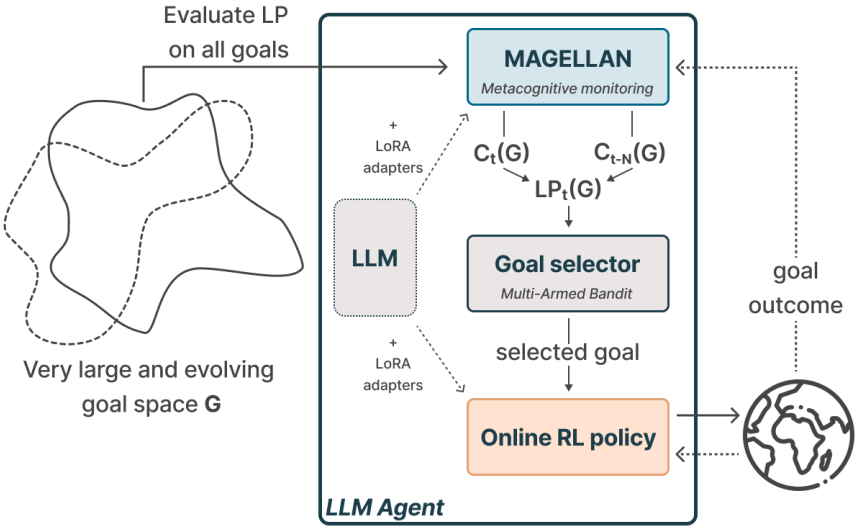


Figure 8.1: **Navigating large goal spaces with MAGELLAN**: During training, our LLM agent uses MAGELLAN to estimate its past and current competence to compute Absolute Learning Progress (ALP) on each goal. Given the per-goal ALP, the LLM agent's goal selector chooses the next goal to practice proportionally to their ALP. The LLM agent then performs a trajectory to achieve this goal and the outcome is used to update both the LLM agent with online RL and MAGELLAN's competence estimation.

In this work, we study how to estimate LP over natural language goals such that an LLM agent learning with online RL in an interactive environment could increase its overall competence as efficiently as possible. For this, we introduce **MAGELLAN**, for **M**et**A**cognitive **GE**neralization of **L**earning progress in **LAN**guage model agents. MAGELLAN leverages the LLM inside the agent to learn an LP estimator that automatically learns semantic relationships and tracks competence transfer between goals in a sample efficient manner (see

Figure 8.1). We evaluate MAGELLAN in the Little-Zoo environment specifically designed as a carefully controlled experimental setup for commonsense-based generalization of agents in a textual environment. In particular, we study the following scientific questions:

• **Q1. Competence estimation** Given an initial set of language goals, how does MAGELLAN's estimation of a learner's competence compare to more classic methods? How does this estimation scale with the size of the goal space?

• **Q2. LLM agent training** Can MAGELLAN be used by an online RL LLM agent to self-organize an efficient learning curriculum over these goals?

• **Q3. Generalization to new goals** How well can MAGELLAN's estimation generalize to predict the agent's competence on unseen goals?

• **Q4. Adaptation to evolving goal sapce** When these new unseen goals are introduced throughout training, can MAGELLAN leverage its generalization abilities to integrate new goals into the curriculum seamlessly?

We show that MAGELLAN 1) accurately and efficiently approximates LP, 2) allows an LLM agent to master all goals from Little-Zoo while prior methods fail when not provided extensive expert knowledge, and 3) generalizes its LP estimation to never-seen goals, enabling faster adaptation to evolving goal spaces. Moreover, we show MAGELLAN learns to cluster goals and achieves results comparable to an LP estimator with expert-defined groups.

8.3 Related work

Goal selection in autotelic agents

Autotelic agents exploring vast goal spaces face a critical challenge: they must prioritize which goals to pursue to efficiently develop general competence (**?**). The Automatic Curriculum Learning (ACL) community has developed various approaches to address this challenge (Portelas et al., 2020f), leveraging different forms of intrinsic motivation: pursuing goals of intermediate difficulty (Held et al., 2017; Racaniere et al., 2020; Castanet et al., 2023), seeking novelty or uncertainty (Warde-Farley et al., 2019; Pong et al., 2019; Pitis et al., 2020), maximizing regret w.r.t. an optimal policy's performance — mostly used in Unsupervised Environment Design (UED) methods (Dennis et al., 2020; Jiang et al., 2021a) — or maximizing LP (Stout & Barto, 2010b; Matiisen et al., 2020; Fournier et al., 2018; Portelas et al., 2020b; Colas et al., 2019; Kanitscheider et al., 2021; Kovač et al., 2023; Zhang et al., 2024b). ACL methods also differ in their use of the goal space: while some methods compute metrics over the goal space and sample from it (e.g. Matiisen et al. (2020); Kanitscheider et al. (2021); Portelas et al. (2020b)), others generate goals (e.g. Held et al. (2017); Castanet et al. (2023); Dennis et al. (2020)). LP-based methods have proven particularly robust, especially upon a limited training budget in which it is not possible to learn all goals (Lopes & Oudeyer, 2012b). They adapt to the agent's capabilities *without requiring environment knowledge* (e.g. which is necessary when using regret) and avoid common pitfalls like getting stuck on goals where progress plateaus or chasing uncontrollable novelty. The key challenge with LP approaches lies in efficiently estimating progress over large goal spaces, which is the focus of our work.

Computing LP over goals

LP measures the expected future improvement in achieving a goal through practice (Oudeyer et al., 2007). Since future progress cannot be directly measured, most approaches use past progress as a proxy, with the recent exception of Kumar et al. (2024)'s Bayesian prediction model. The most direct approach to estimate LP is to regularly reevaluate the agent's competence for each goal (Kanitscheider et al., 2021; Zhang et al., 2024b), which accurately captures competence transfer — the phenomenon where practicing one goal affects performance on other goals. However, this becomes computationally prohibitive for large discrete goal spaces and is just impossible for continuous ones. One way to address this is to only rely on online estimations, where a goal's estimated competence is only updated when this goal is practiced. Online estimations nonetheless fail to capture competence transfer and existing methods addressed this by grouping goals with similar competence together. For continuous spaces, approaches either learn to partition the space directly when dimensionality is low (Oudeyer et al., 2007; Baranes & Oudeyer, 2013; Portelas et al., 2020b), or first embed high-dimensional goals into a lower-dimensional space before partitioning (Laversanne-Finot et al., 2018a; Kovač et al., 2023). For discrete spaces, methods typically rely on expert-defined groupings (Stout & Barto, 2010b; Matiisen et al., 2020). However, these grouping approaches are inherently brittle: they assume no transfer between groups while potentially masking competence variations within groups. This limitation is particularly acute for high-dimensional structured spaces like natural language, where competence transfer naturally occurs between semantically similar goals regardless of predefined groupings. Instead, MAGELLAN leverages an LLM's semantic understanding to dynamically model competence transfer between goals, enabling efficient and adaptive LP estimation without requiring predefined groupings or exhaustive evaluation.

Autonomous LLM agents

Recent work has explored using LLMs to solve complex tasks in interactive environments. Early approaches focused on direct action prediction using LLMs, either incorporating environmental feedback (Huang et al., 2022; Yao et al., 2022b; Hao et al., 2023; Shinn et al., 2023; Wang et al., 2024a) or operating without it (Ahn et al., 2022). However, these methods did not update the LLM's knowledge through environmental interactions. A new direction emerged with GLAM (Carta et al., 2023), followed by (Wen et al., 2024c,a; Zhou et al., 2024) that ground LLMs in interactive environments using online RL. The resulting LLM agents demonstrated remarkable generalization across language tasks; however, they lack the autotelic mechanisms necessary for navigating expansive goal spaces. In this study, we enhance a SAC-GLAM (Gaven et al., 2024) LLM agent with metacognitive abilities, enabling it to estimate its LP and prioritize goals within large language spaces.

8.4 MAGELLAN: Measure and generalize learning progress with a Large Language Model

In this section, we detail how MAGELLAN learns a metacognitive module that estimates and generalizes an agent's LP over language goal spaces. We then explain classic LP baselines

against which MAGELLAN is compared. Finally, we introduce the Little-Zoo environment, specifically designed to study commonsense-based generalization abilities of LLM agents when facing large language goal space.

Problem statement

Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ be an MDP, with $\mathcal{S}$ a set of states, $\mathcal{T}$ the transition function, $\mathcal{A}$ the action space and $\mathcal{R}$ the reward function. Let $\mathcal{G}$ be a goal space and Π the policy space. We define a competence function $C_{\mathcal{M},\pi} : \mathcal{G} \to \mathbb{R}$ that indicates the competence of a policy $\pi \in \Pi$ for a goal in $\mathcal{M}$.[1] The final aim is to find the optimal policy π^* that maximizes

$$J_{\mathcal{M}}(\pi) = \mathbb{E}_{g\sim\mathcal{U}(G)}[C_{\mathcal{M},\pi}(g)],$$

where $\mathcal{U}(X)$ is the uniform distribution over a set X.

In this work, we focus on episodic online goal-conditioned RL with sparse and binary rewards, defined on a goal-augmented MDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{G}, \mathcal{R})$, with $\mathcal{R} : \mathcal{S} \times \mathcal{G} \to \{0; 1\}$ a binary success function indicating whether a state s satisfies a goal g. Here, we define $\mathcal{G} = \{\mathcal{S}_0 \times \mathcal{I} | \mathcal{S}_0 \subseteq \mathcal{S}\}$, with $\mathcal{I}$ an instruction space and $\mathcal{S}_0$ the set of initial states. We consider a textual environment where a prompting function $\phi : \mathcal{S} \times \mathcal{I} \to \mathcal{V}^K$ is given to transform any pair (state, instruction) into a textual prompt of K tokens in a given vocabulary $\mathcal{V}$. Thus, from the agent side, the policy π selects any action $a_h \in \mathcal{A}$ by sampling from a categorical distribution $\pi(.|\phi(s_h, i))$ at any step t of the episode.

For our competence function we use the success probability $\mathbb{P}_\pi(s_0, i)$ defined as the probability for π, starting from s_0, to fulfill i within T steps: $\mathbb{P}_\pi(s_0, i) = \mathbb{E}_{\tau\sim\pi(\tau|\phi(s_0,i))}[r_{\tau,i}]$, with $r_{\tau,i} = \mathbb{1}(\exists s_h \in \tau, R(s_h, i) = 1)$ the goal outcome of episode τ for instruction i, $\mathbb{1}$ the indicator function and $\pi(\tau|\phi(s_0, i))$ the distribution of episodes of T steps induced by π to fulfilling i from s_0. In this setting our objective becomes:

$$J(\pi) = \mathbb{E}_{s_0\sim\mathcal{U}(S_0),i\sim\mathcal{U}(I)} \left[\mathbb{P}_\pi(s_0, i)\right].$$

However, given the possibly huge number of goals (s_0, i), the direct maximization of the problem becomes particularly inefficient. Our aim is to leverage transfer of competence between goals and focus during training on the ones maximizing LP. We denote as π^h the policy obtained after h episodes using an RL algorithm, with Γ^h the set of all trajectories collected during training using $\{\pi^{k-1} | k \in [|1, h|]\}$. The goal of each episode is sampled using a task selector $\eta_{\mathcal{G}}$ that selects a goal based on collected trajectories $\eta_{\mathcal{G}}(\Gamma^h) = g$. Given a budget of H training episodes, we thus consider the problem of approaching the optimal selector $\eta^*_{\mathcal{G}} = \arg\max_{\eta_{\mathcal{G}}} J(\pi^H)$. In particular, we build on prior work to construct η_G on a proxy of the LP at each training episode h. We define the LP for any goal $g = (s_0, i)$ as the improvement of the policy at episode h after k episodes on goal g: $LP^k_{\pi^h}(g) = \mathbb{P}_{\pi^{h+k}}(g) - \mathbb{P}_{\pi^h}(g)$. As highlighted in Section 8.3, since computing the future

[1] In all generality a competence function does not assume the goal to be inside the MDP. For instance, g could ask for a maximum number of steps.

competence on all goals is intractable, prior approaches approximate future progress with past progress ($LP^k_{\pi^h}(g) \approx \mathbb{P}_{\pi^h}(g) - \mathbb{P}_{\pi^{h-k}}(g)$). Nonetheless, accurately estimating past and current competence remains a challenge in large discrete goal spaces.

Metacognitive generalization of learning progress in language model agents

With MAGELLAN, we propose to learn estimators of the current and past policy's competence for any goal. As opposed to prior works, which either consider all goals independently or use goal groupings, we argue that learning goal-conditioned estimators would allow generalization between similar goals without defining any clear group. We propose to leverage the LLM used by our agent to learn the parameters θ_h of a competence estimator $C_{\theta_h}(g)$ for a policy π_h on a goal g. We compute $C_{\theta_h}(g)$ by giving g in the LLM's prompt, which produces a latent representation on top of its final decoder block for the last token. We use a Multi-Layer Perceptron (MLP) to output the estimated competence based on this representation. We train both the LLM and the MLP, leveraging the LLM's ability to project goals into a latent space where semantically similar goals are close. By updating the estimated competence of one goal, this allows MAGELLAN to also update close goals.

In practice, we maintain a buffer $\mathcal{D}_h$ which contains, for the M most recent training episodes at t (i.e., $\tau^{h-M} \dots \tau^h$), their corresponding pair of goal and outcome (i.e. $(g = (s_0, i), r_{\tau,i})$ for each τ). As this work focuses on success probability (i.e., we want $C_{\theta_h}(g) \approx \mathbb{P}_{\pi^h}(g)$), we train C_{θ_h} using stochastic gradient descent to minimize the binary cross-entropy: $\mathcal{L}(\theta_h) = \mathbb{E}_{(g,r)\sim\mathcal{D}_h}\left[BCE(r, C_{\theta_h}(g))\right]$.

We maintain another buffer $\mathcal{B}_h$ storing the last N weights of our competence estimator: $\mathcal{B}_h = [\theta_{h-N}, \theta_{h+1-N}, \dots, \theta_h]$. Weights are added to the buffer every time the competence estimator is updated, enabling access to estimations of the policy's competence from time t to $t - N$. Using this information, we estimate the absolute LP (ALP) (Baranes & Oudeyer, 2013; Kanitscheider et al., 2021), tracking both progress and forgetting, as follows:

$$\hat{ALP}_{\pi_h}(g) = |C_{\theta_h}(g) - C_{\theta_{t-N}}(g)|. \tag{8.1}$$

This ALP estimation can subsequently be used to structure the agent's curriculum. We apply the multi-armed bandit goal selection scheme introduced by Lopes & Oudeyer (2012b) where each arm is a goal, and its utility is MAGELLAN's estimate of the ALP of this goal. Goals are then sampled proportionally to their estimated ALP with an annealing ϵ-greedy scheme (ϵ decreasing from 1 to 0.2). In practice, we train two separate versions of the same initial LLM (using LoRA adapters (Hu et al., 2022)): one for the policy and one for MAGELLAN's current competence estimator. We show in Appendix C.4.1 ablations on architectural choices indicating that 1) keeping the LLM frozen leads to poor results, highlighting the need for a dynamic representation space (see also Figure 8.7), and 2) training separate LoRA adapters for the policy and MAGELLAN leads to more stability.

Classic ALP baselines

Following the literature on ALP in Section 8.3, we implement classic approaches, focusing on two dimensions. First, we consider Online (Baranes & Oudeyer, 2013; Matiisen et al., 2020) vs Evaluation-based ALP (Kanitscheider et al., 2021; Zhang et al., 2024b) estimation. Then, we consider directly using the goal space (Portelas et al., 2020b; Kanitscheider et al., 2021) or using expert-defined groups of goals with assumed competence transfer (Stout & Barto, 2010b; Colas et al., 2019). The latter requires extensive Expert Knowledge (EK) given the absence of automatic approach for discrete goal spaces. As expert-defined groups are created beforehand, no competence transfer is assumed across groups, which is likely to happen in spaces like natural language, where transfer occurs between semantically close goals regardless of groups.

We thus implement four baselines (see all details in Appendix C.3.3):

- **Eval-ALP**: Every N episodes, training stops and the agent is separately evaluated on each goal to obtain a competence estimate. The per-goal ALP is the absolute difference between estimates at t and $t - N$. The same goal selection scheme as in MAGELLAN is used according to the per-goal ALP estimations.
- **EK-Eval-ALP**: Every N episodes, training stops and the agent is evaluated on multiple goals randomly sampled in each expert-defined group to obtain a per-group averaged competence. The per-group ALP is computed using the absolute difference between the competence at t and $t - N$. The goal selection process first selects a group using the same selection scheme as MAGELLAN for goals. Given a selected group, a goal from this group is randomly selected.
- **Online-ALP**: At each goal practiced, the observed policy's competence is added to a buffer of $2M$ past experiences for this goal. The ALP is computed using the absolute difference between the average competence over the last and first M experiences in the buffer. The same goal selection scheme as MAGELLAN and Eval-ALP is used.
- **EK-Online-ALP**: At each goal practiced, the observed policy's competence is added to a buffer of $2M$ past experiences for the goal's expert-defined group. The per-group ALP is computed using the absolute difference between the average competence over the last and first M experiences in the group's buffer. The same goal selection scheme as EK-Eval-ALP is used.

We summarize the four methods above and MAGELLAN in Table 8.1 based on their *Efficiency* (i.e. computational cost introduced by additional evaluations), *Competence Transfer tracking* and *no Expert Knowledge requirement*. We consider a method's efficiency as "high" if it does not require any additional evaluation (i.e. it only uses the performance observed on goals sampled), and as "low" otherwise. We evaluate the competence transfer tracking using the following criteria:

- absence of +: the estimated competence is updated only on sampled goals.
- +: the estimated competence is updated on a predefined group the sampled goal belongs to.
- ++: the estimated competence is updated on a dynamically learned group the sampled goal belongs to.
- +++: the estimated competence is updated on all goals.

We provide in Appendix C.2 the same table for all prior works covered in Section 8.3.

8.5 Experiments

We provide empirical answers to our scientific questions using experiments with 8 different random seeds in the Little-Zoo environment in Section 8.5.1. For our LLM agent, we use SAC-GLAM (Gaven et al., 2024) to finetune Flan-T5 248M (Raffel et al., 2020) as per SAC-GLAM's experiments. We compare MAGELLAN to the classic approaches presented in 8.4. For methods accessing external expert knowledge, we use Little-Zoo's hidden goal families (grasp any object, grow plant, grow herbivore, grow carnivore) but add only possible goals in these groups. Indeed, these baselines are based on groups predefined in advance by human experts with a strong assumption: the goals within a group share the same learning dynamics and therefore the agent's competence is the same over all goals in the group. *If impossible goals were included, these groups would lose their relevance.* Moreover, because of the large number of impossible goals, the average competence within each group will always be very close to 0. There will be no progress niche that the method can use to generate a curriculum, and performance will probably be very close to the random baseline. We thus provide an additional group containing all impossible goals. In all our experiments, we use success rate (i.e. average outcome over multiple trials for a goal) noted SR as the observed competence.

We first study how well MAGELLAN's competence estimation compares to baselines (**Q1**). Then, we study how the different methods (except Eval-ALP and EK-Eval-ALP, which are too costly to run while EK-Online-ALP provides a good estimation of their performance) compare when scaffolding the LLM agent's curriculum (**Q2**). We show how these competence estimators also generalize to goals not seen during training (**Q3**). Finally, we study how all methods adapt as the goal space evolves (**Q4**) by replacing all goals at different points throughout training.

8.5.1 The Little-Zoo environment as a testbed

Evaluating commonsense-based generalization of LLM agents in textual environments requires several key properties. The environment must be fully text-based, with all observations, actions, and goals expressed in natural language. It should feature a diverse set of goals with varying difficulty levels, enabling the assessment of the agent's ability to learn and generalize complex skills. Additionally, these goals should be organized into hidden families based on commonsense knowledge, allowing for targeted evaluation of generalization capabilities.

Existing environments for LLM agents do not have such requirements. Creative environments like Minecraft (Johnson et al., 2016) or Crafter (Hafner, 2022) rely on image-based observations and require an image captioner to use LLM agents. Textual environments like BabyAI-text (Carta et al., 2023) focus on navigation skills without any commonsense-based generalization. Although WordCraft (Jiang et al., 2020) incorporates commonsense-based goals, no relationship between goals exists, limiting the analysis of the agent's generalization

abilities. To address these gaps, we introduce Little-Zoo, a novel environment explicitly designed to meet these criteria.

Built upon the Playground environment (Colas et al., 2020a), Little-Zoo is fully text-based, with observations, goals, and actions expressed in natural language. It features objects that can be combined together and are grouped into the following hidden categories: furniture (which cannot be combined), plants, herbivores, and carnivores. Given the set of all objects and a set of instructions, Little-Zoo's goal space is the combination of all possible instructions and scene initializations. The feasibility of a goal thus depends on the objects available, making most combinations infeasible and not trivial to detect (see Figure 8.2 and Appendix C.1.3). For instance, these are respectively feasible and infeasible goals: *"Goal: Grow deer. You see: baby deer, bookshelf, water, tomato seed."*; *"Goal: Grow deer. You see: baby deer, bookshelf, baby lion, tomato seed."* (water is missing in the last one). Instructions are hierarchically structured, ranging from simple grasping tasks to more complex sequences involving object interactions (e.g. "growing" animals). The complete goal space contains approximately 20 million combinations. In our experiments, we subsample goals with the following proportions: 80% of the goals are impossible 16% involve grasping, 3.2% involve growing plants, 0.7% involve herbivores, and 0.1% involve carnivores. These proportions correspond to proportions in the complete goal space (see Appendix C.1.4).

Little-Zoo is a deterministic, fully-observable and episodic environment: the agent begins an episode by standing on nothing, with full visibility of the whole scene. The action space consists of 8 actions, including movement to objects, grasping, and releasing objects. Observations include the objects in the scene, the ones in the agent's inventory, as well as the object the agent is standing on. See Appendix C.1 for details on Little-Zoo.

8.5.2 Q1. Competence estimation

	Eff.	**Transf.**	**No EK**
EK-Eval-ALP	low	+	×
Eval-ALP	low	+++	✓
EK-Online-ALP	high	+	×
Online-ALP	high		✓
MAGELLAN	high	+++	✓

Table 8.1: Comparison of ALP estimation methods. We use the following dimensions: computational Efficiency, competence Transfer tracking, and required Expert Knowledge.

To assess the ALP methods' ability to efficiently estimate competence, we designed an experimental setup in which our LLM agent was trained for 50k episodes on the Little-Zoo environment with varying goal space sizes (25k, 50k, 100k), while keeping the same repartition between goal types. As computing the expected ALP to train this agent is intractable, one could argue that Eval-ALP is the best approximation. However, it remains

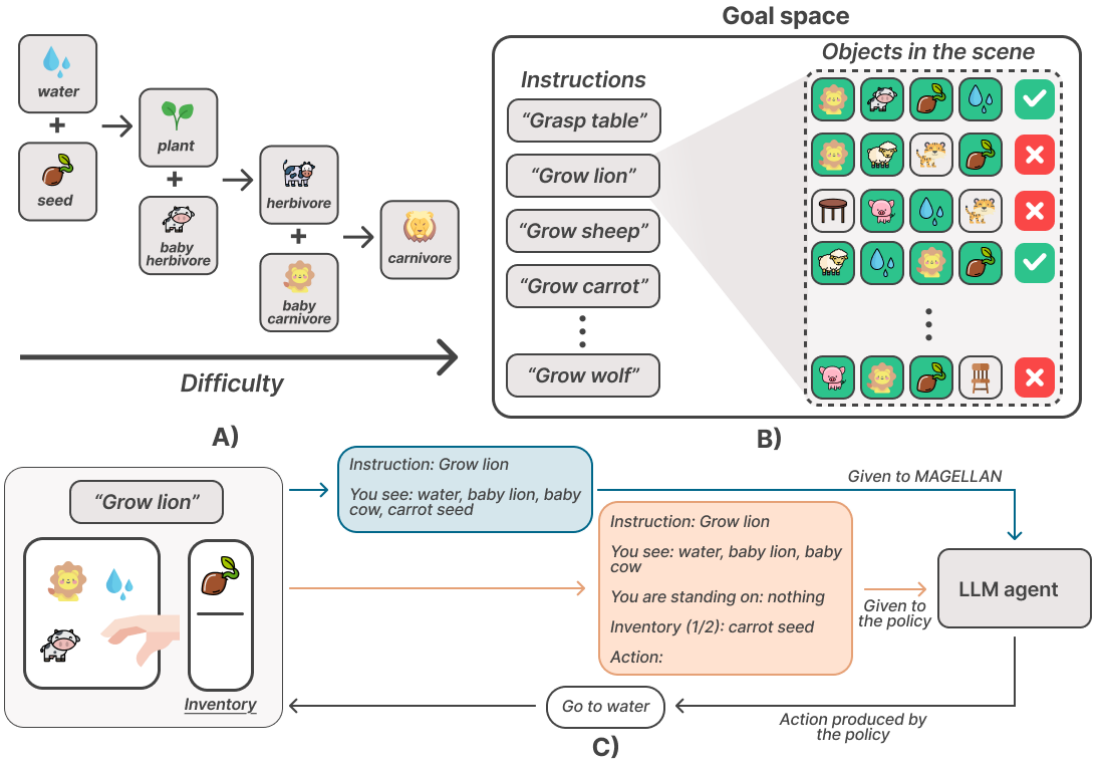


Figure 8.2: A) Little-Zoo's tech tree. B) Little-Zoo's goal space is composed of all the possible combinations between instructions and objects that can be in the scene. Most object configurations make an instruction infeasible (e.g. "grow lion" is impossible with the second configuration, as water, needed to obtain plants, is missing). C) Little-Zoo provides a textual description that is given in our LLM agent's prompt.

too computationally costly to run, even when performing only 50k training episodes with 25k goals. We thus chose to sample goals according to EK-Eval-ALP's estimations. To obtain an accurate estimate, we perform 2048 per-group evaluations every 1000 episodes. The per-group competence evaluated by EK-Eval-ALP is consequently our competence reference, and we compare the other methods against it. For Eval-ALP, we consider it to have zero error, and its computational cost can be estimated without running it. For MAGELLAN and Online-ALP, we average the per-goal competence over groups to compute the error w.r.t. EK-Eval-ALP. Figure 8.3 shows the average error on competence throughout training along and the cost of competence evaluation (i.e. the total number of episodes used only to evaluate competence).

As indicated in Table 8.1, MAGELLAN performs on a par with Eval-ALP, showing that it accurately estimates the transfer of competence while using online estimations. We also observe similar competence errors to methods using expert-defined groups, hinting at MAGELLAN's abilities at learning semantical relationships between goals. We provide a more in-depth analysis of such relationships in Section C.4.4. Finally, MAGELLAN achieves this performance without an estimation cost.

To demonstrate that our method generalizes beyond the Little-Zoo environment, we conducted an additional experiment using goals derived from the OpenR1-Math-220k dataset (Face,

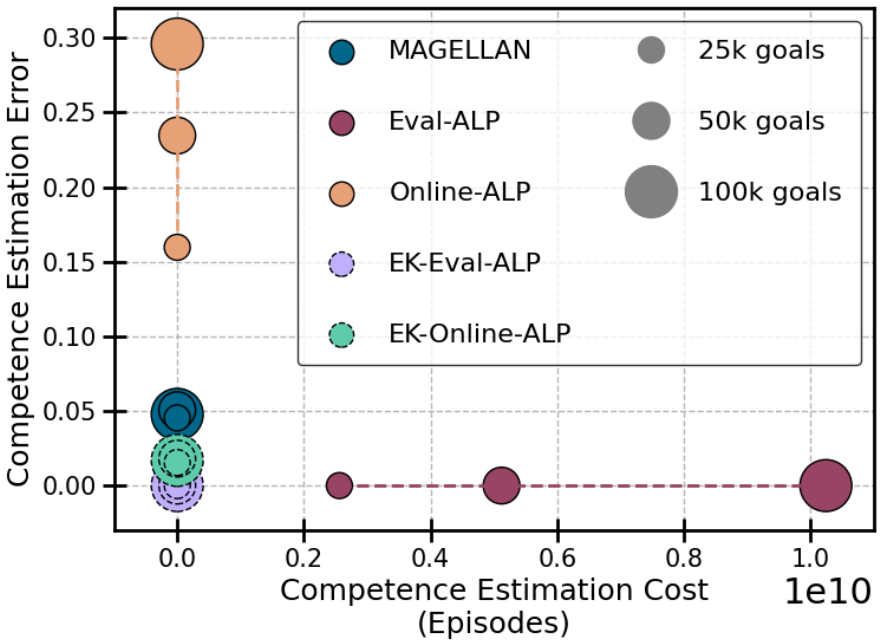


Figure 8.3: Scaling of competence estimation error and competence estimation cost (i.e. total number of additional evaluation episodes) when increasing the goal space size.

2025). In this setup, rather than training a real agent, we simulate the learning of an agent that progressively acquires skills in three categories: *Algebra*, then *Geometry*, and finally *Number Theory*. We compare the competence estimation from MAGELLAN and Online-ALP to the underlying true competence. Figure 8.4 shows that MAGELLAN accurately tracks the agent's competence and distinguishes between the different mathematical categories, clearly outperforming Online-ALP. Additional results, including experiments in the BabyAI-Text environment (Chevalier-Boisvert et al., 2019; Carta et al., 2023) and an ablation study on the impact of LLM size within MAGELLAN, are provided in Appendix C.4.2.

8.5.3 Q2. LLM agent training

Curriculum learning

As demonstrated in 8.5.2, MAGELLAN provides a superior competence estimation than Online-ALP. We further investigate whether this improvement translates into a better curriculum and improved overall goal mastery. We train our LLM agent on the goal space of Little-Zoo with 25k goals for 500k episodes using four methods: MAGELLAN, Online-ALP, EK-Online-ALP and "Uniform", where goals are sampled uniformly. We do not report EK-Eval-ALP, as we report EK-Online-ALP which produces similar competence estimation with no cost. We report the agent's SR every 5000 training episodes by evaluating it on 64 goals uniformly sampled for each category. Figure 8.5 shows the evolution of SR averaged over all categories. Our results show that MAGELLAN is the only method without expert-defined grouping to obtain an SR of at least 90% in all categories. It also masters the

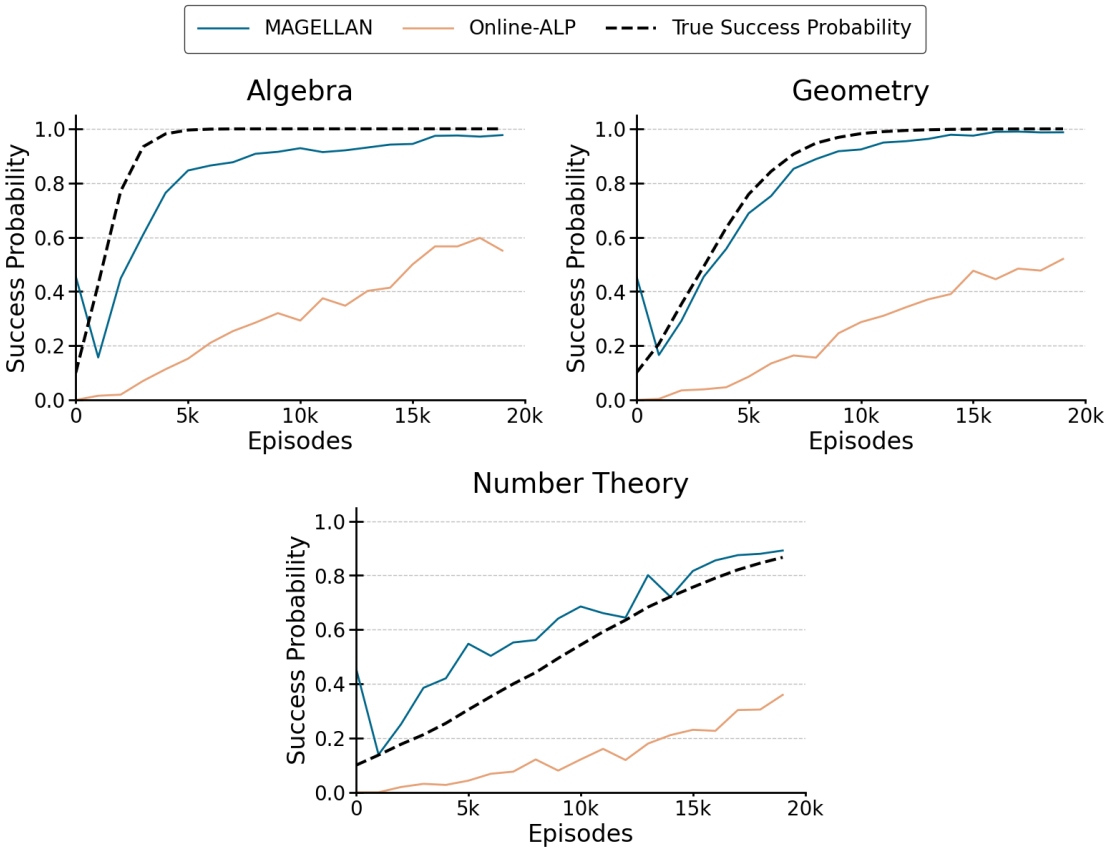


Figure 8.4: Competence estimation on OpenR1-Math-220k. MAGELLAN (blue) accurately tracks competence across Algebra, Geometry, and Number Theory, closely matching true success probabilities and outperforming Online-ALP (orange).

categories significantly faster than baselines. Despite MAGELLAN's similar competence estimation as EK-Online-ALP, the latter learns faster by leveraging the expert-defined groups to better explore. This is because MAGELLAN explores by uniformly sampling goals whereas EK-Online-ALP uniformly samples groups, easily discarding impossible goals.

Goal sampling strategies

We analyze the sampling strategies of the different methods. In Figure 8.6, it appears that MAGELLAN has a sampling strategy close to the one of EK-Online-ALP. The main difference is that EK-Online-ALP does not select impossible goals thanks to the use of expert knowledge. MAGELLAN's sampling strategy quickly begins to sample goals of type "Grow herbivore" after sampling goals of type "Grow plant", which contrasts with the strategy of Online-ALP. Indeed, Online-ALP mostly selects goals already encountered, getting stuck on one goal type before moving to another. The similarity between MAGELLAN and EK-Online-ALP indicates that MAGELLAN is able to cluster the goal space dynamically, using this information to generalize its competence estimation to sample interesting goals not yet discovered. More results on the dynamic clustering of the goal space by MAGELLAN are given in Appendix C.4.4.

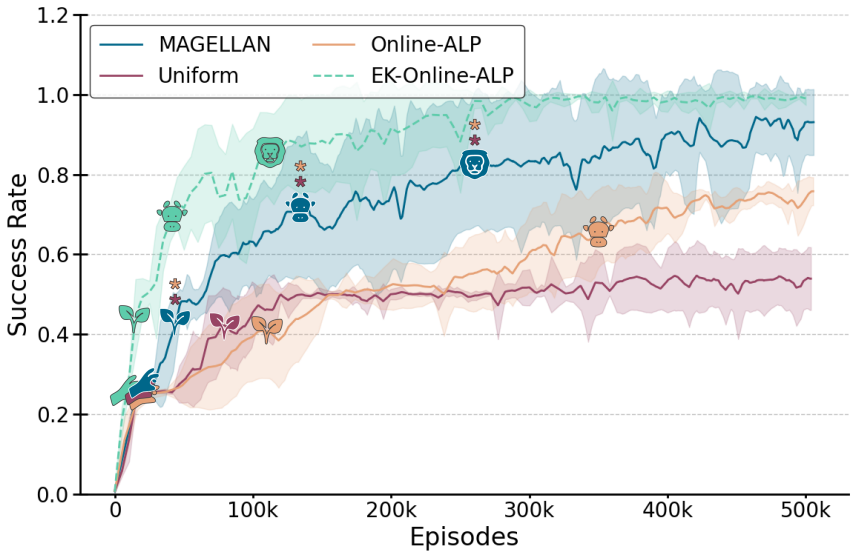
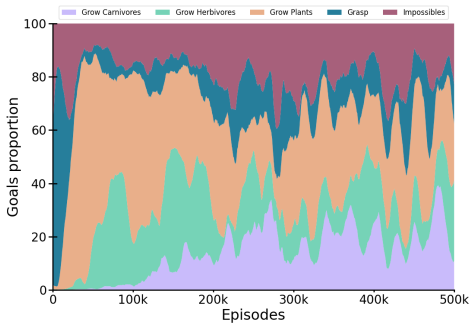
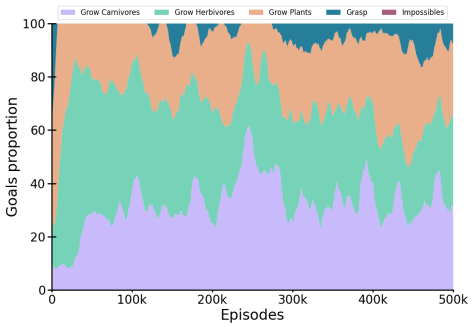


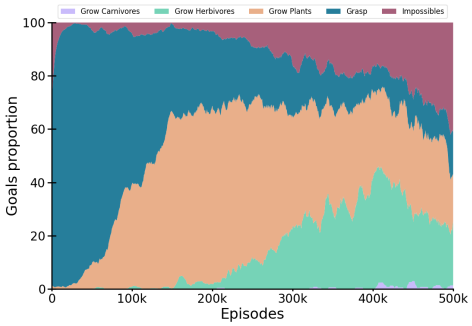
Figure 8.5: Evolution of the observed competence (SR) when evaluating policies on 64 training goals per category every 5000 episodes. We report the average SR over evaluated goals along with standard deviation (8 seeds). Icons indicate the average time step at which a method mastered a goal (i.e. SR $> 90\%$). We add stars to MAGELLAN, denoting significantly earlier mastery of a category compared to the method with the star's color (p-value $< 8 \times 10^{-4}$). The dotted line (EK-Online-ALP) indicates that the method relies on expert knowledge.



(a) MAGELLAN's sampling strategy.



(b) EK-Online-ALP's sampling strategy.



(c) Online-ALP's sampling strategy.

Figure 8.6: Goal sampling strategies of MAGELLAN, EK-Online-ALP, Online-ALP. We do not take into account the 20% uniformly sampled goals from the ε-greedy exploration.

8.5.4 Q3. Generalization to new goals

Generalization results

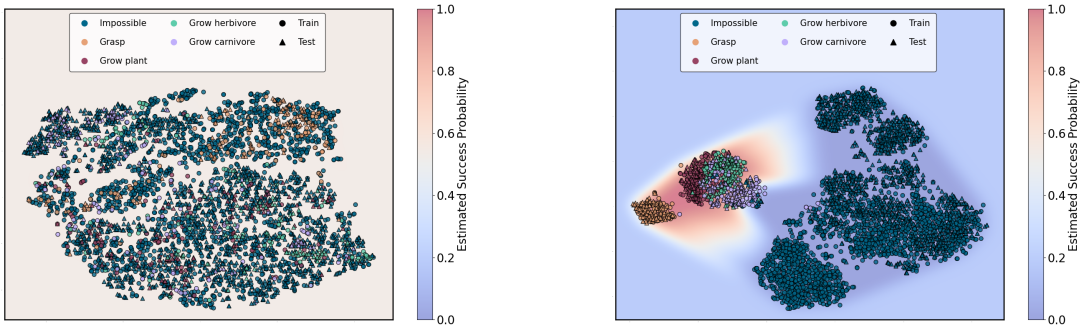
We move further and study the generalization abilities of both our LLM agent and competence estimators. While Section 8.5.3 evaluates each policy on 64 goals per category that belong to the training goal space, this section reports evaluation on a held-out test set composed of unseen goals. As in the previous section, evaluations were performed every 5000 training episodes. However, instead of reporting the policy's observed competence (SR) during evaluation, we report the difference between the observed competence and the competence estimated by the policy's ALP method. We show in Table 8.2 the average error over training.

By only tracking competence on goals practiced by the policy (i.e. the ones from the training goal space), Online-ALP cannot provide any estimation for new unseen goals and uses its default competence of 0. This leads to the largest error among methods except for "Grow carnivore" goals as the policies trained with Online-ALP never mastered these goals (see Figure 8.5). MAGELLAN successfully generalizes its competence estimation and obtains a small error. Finally, EK-Online-ALP produces accurate estimations based on expert knowledge of which group each test goal belongs to. Appendix C.4.4 provides detailed results indicating our LLM agents do not perfectly generalize, which explains MAGELLAN and EK-Eval-ALP estimation error.

Table 8.2: We evaluate the policies trained with each ALP method on a held-out test set. We show the difference between the observed and predicted competence. Online-ALP's performance on "Grow carnivore" is simply explained by the fact that its policies never mastered this goal category.

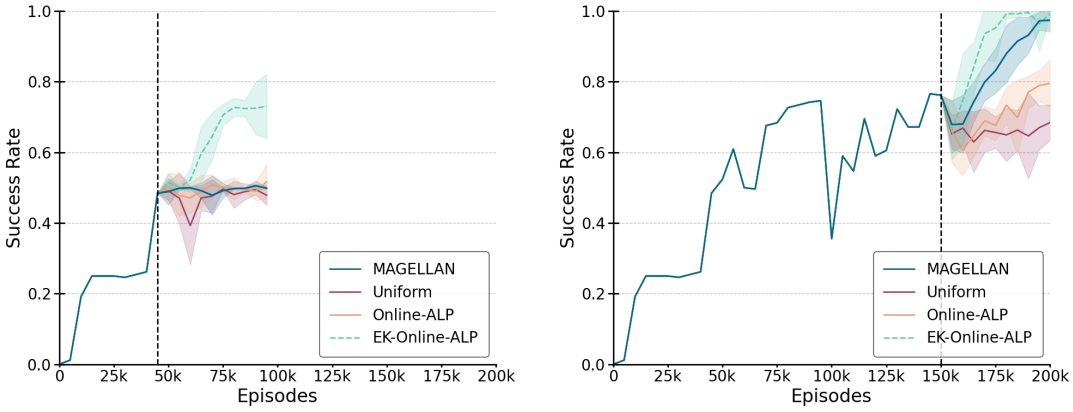
Categories	**MAGELLAN** (Mean ± Std)	**Online-ALP** (Mean ± Std)	**EK-Online-ALP** (Mean ± Std)
Grasp	**0.01 ± 0.00**	0.98 ± 0.00	0.01 ± 0.00
Grow plant	**0.05 ± 0.03**	0.78 ± 0.07	0.03 ± 0.01
Grow herbivore	**0.08 ± 0.05**	0.34 ± 0.18	0.06 ± 0.02
Grow carnivore	0.30 ± 0.16	**0.00 ± 0.00**	0.28 ± 0.08
Mean	**0.11 ± 0.06**	0.53 ± 0.06	0.09 ± 0.03

We further investigate MAGELLAN's generalization abilities by projecting train (Q2) and test (Q3) goals from a single seed into the LLM embedding space MAGELLAN used. The embedding space is plotted both before and after training (Figure 8.7) with projections obtained via t-SNE (van der Maaten & Hinton, 2008). The initial embedding space lacks any discernible structure for goal classification. Post-training, the space exhibits significant restructuring, with similar goals accurately clustered. A small subset of "Grow carnivore" goals are misclassified as impossible, likely due to incomplete mastery of this category by the LLM agent. Furthermore, the spatial arrangement of goals correlates with estimated success rates: newly learned goals tend to lie near the boundary with impossible goals. Additionally, goals from the test set are well clustered, demonstrating strong generalization.



(a) (b)

Figure 8.7: MAGELLAN's LLM embedding space displayed using t-SNE with goals used in Q2 (Train) and Q3 (Test), along with the estimated success probability and linear interpolation between goals. We show the embedding space for a single seed (a) before training and (b) at the end of the 500k training steps. We see that impossible goals have been left aside, and that the other goals with a high estimated success probability are clustered consistently.



(a) (b)

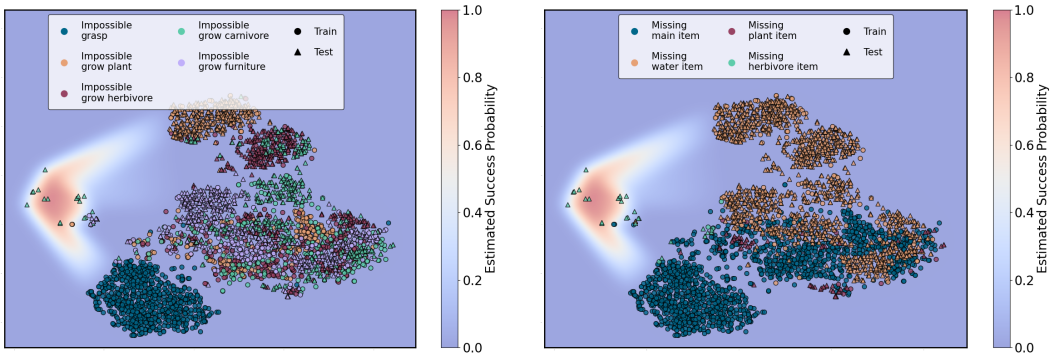
Figure 8.8: **Adaptation tests**: Using a single's seed training of 500k episodes, we stop and replace all goals with unseen ones every 50k episodes. We then resume training and sample goals using each method for 50k training episodes. We show two isolated and representative points of goal replacement: (a) there is no ALP on any goal (after 50k training episodes), and (b) some goals (here, "Grow carnivores" after 150k training episodes) have a high ALP. We report the evolution of SR when evaluating the policies on 64 goals per category from the new training set every 5000 episodes. Results show the average competence over evaluated goals along with standard deviation (8 seeds).

Embedding of impossible goals

As detailed in Section C.1.3, a goal is considered impossible either due to the absence of a required element or when attempting to grow furniture. Accurately identifying impossible goals and generalizing this knowledge is crucial for strong performance in the Little-Zoo environment. In this section, we analyze how MAGELLAN handles impossible goals.

Figure 8.9a illustrates the clustering of different categories of impossible goals. The impossible "Grasp" goals form a compact cluster, while the "Grow" goals—categorized as "Grow plant", "Grow herbivore", "Grow carnivore", and "Grow furniture"—exhibit four less-defined clusters. Additionally, a large, mixed cluster contains various impossible "Grow" goals. However, when examining the same embeddings through the lens of missing elements, as shown in Figure 8.9, a distinct structure emerges: MAGELLAN also organizes goals based on the absent element in the scene.

For instance, the previously observed clusters of impossible "Grow" goals largely align with the "missing water" category, as water is a prerequisite for all "Grow" goals. Moreover, the central mixed cluster from Figure 8.9a is further clustered into sub-clusters based on the missing element. In particular, the red cluster at the bottom of Figure 8.9 represents the absence of a plant, encompassing both "Grow herbivore" and "Grow carnivore" goals. This suggests that MAGELLAN effectively captures underlying environmental dependencies to determine goal feasibility. This ability to infer structural properties of the environment likely contributes to the strong generalization performance observed in Section 8.5.4 and Section 8.5.5.



(a) (b)

Figure 8.9: Plot of the t-SNE-projected embeddings of impossible goals, where each goal is color-coded based (a) on the type of impossible goal (b) on the specific missing element required to make it possible.

8.5.5 Q4. Adaptation to evolving goal space

Finally, we investigate how each ALP method can adapt when the goal space evolves. For this, we isolate the training of one seed using MAGELLAN in Section C.4.3. Every 50k episodes over the 500k training episodes, we stop training, replace the training goals with the ones in our held-out test set, and start four trainings (with 8 seeds each) with this new goal space:

one with each ALP method for 50k steps. We expect MAGELLAN to quickly adapt to new goals leveraging semantical relationships between the new and old goals. Online-ALP starts with a competence estimation of 0 on the new goals as in Section 8.5.4. For EK-Online-ALP, up to the episode where we change the goal space, we make it track the policy's competence in parallel to MAGELLAN. It thus starts with a per-group competence estimation when the goals are replaced along with the information of which expert-defined group each new goal belongs to. We study all adaptation training in Appendix C.4.5 but isolate and study in Figure 8.8 two of them chosen as representative of the scenarios encountered:

- **Scenario zero LP (Figure 8.8a):** The agent has mastered the "Grasp" and "Grow plants" goals and has 0 ALP across all goals. In this scenario, all ALP estimations are equivalent. EK-Online-LP manages to discover new ALP niches faster as all impossible goals are in the same group.
- **Scenario high LP (Figure 8.8b):** The agent is getting a high ALP as it is learning some "Grow carnivores" goals. Here, MAGELLAN outperforms baselines by generalizing its ALP estimation and continuing training on these goals. MAGELLAN even gets on par performance with EK-Online-LP.

8.6 Discussion

In this work, we introduced MAGELLAN, a metacognitive module designed to efficiently estimate an LLM agent's LP across large, language-defined goal spaces. By leveraging the capabilities of the LLM embedded within the agent, MAGELLAN learns to capture semantic relationships between goals, thereby enabling accurate and scalable LP estimation. This contrasts with previous methods, which often rely on expert-defined goal taxonomies or fail to generalize across extensive goal spaces. By guiding goal selection based on LP estimations, MAGELLAN empowers the agent to construct effective curricula and progressively master an entire goal space, including novel goals, without external supervision. Furthermore, MAGELLAN's semantic reasoning enables competence estimation even for previously unseen goals and ensures adaptability as the goal space evolves. While our experiments have focused on a textual environment, the general nature of the method suggests applicability to a broader class of problems, particularly those involving code-based or structured symbolic goals (Wang et al., 2024a; Pourcel et al., 2024b). Additionally, the capacity of MAGELLAN to model relationships between discrete goals opens the door to applications beyond artificial agents, notably in educational domains where learners must navigate large sets of language-defined tasks (Doroudi et al., 2019; Clement et al., 2015).

Nonetheless, several limitations of MAGELLAN remain. Chief among them is the requirement to estimate competence across the entire goal space, which may become intractable for open-ended agents facing thousands of goals. A promising direction to address this issue lies in the automatic partitioning of the goal space, enabling the estimation of global competence within subspaces rather than over all individual goals. Hierarchical decompositions could alleviate the computational burden while preserving goal coverage. Investigating such hierarchical strategies remains an open avenue for future research. Another, more subtle, limitation concerns the contextual variability of goal execution. Our current implementation assumes that the goal itself encodes the relevant environmental state (as is the case in our

textual environment). However, in more complex or embodied settings—such as Crafter (Hafner, 2022)—the agent's ability to achieve a goal like `Collect stone` depends on its current state and available tools. This variation transforms the problem from a classical multi-armed bandit to a contextual bandit formulation. This extension has been explored more rigorously in our HERAKLES method, detailed in Chapter 13 and further elaborated in Appendix F.7. Finally, to broaden the range of applicable environments, a natural extension of MAGELLAN would involve integrating Visual Language Models (VLMs), enabling goal management in multimodal contexts where visual understanding is required.

Finally, while MAGELLAN generalizes effectively to existing and unseen goals, it remains constrained to selecting goals from a predefined or previously encountered set. However, a truly open-ended agent must be capable not only of selecting but also of autonomously generating new goals during exploration. In the following chapter, we investigate a complementary mechanism for generating novel and diverse goals that maximize current learning progress, thus extending MAGELLAN's capabilities into the creative frontier of open-ended learning.

Chapter 9

Generate new learnable goals

9.1 Motivation

Humans possess a remarkable capacity for continuously generating new goals throughout their lives. This phenomenon, referred to as *open-endedness*, enables individuals to structure their exploration of the world by acquiring new skills that can be recombined and repurposed to accomplish increasingly complex behaviours. For artificial agents to be considered genuinely open-ended, they must exhibit the ability to autonomously generate and pursue novel, learnable artefacts (Sigaud et al., 2024; Edward et al., 2024). However, exploring a vast goal space is inherently challenging. Even in a constrained domain such as chess, the estimated number of possible games exceeds 10^{120} (Shannon, 1988). In richer environments with more complex dynamics—such as Minecraft—the enumeration of all possible goals becomes utterly intractable. Current artificial agents are limited in their capacity to explore such environments and typically rely on significant human intervention to structure their learning. A promising alternative, rooted in developmental robotics and cognitive science, posits that agents should actively explore their environment by autonomously setting and striving to achieve self-generated goals—an approach formalised by autotelic, curiosity-driven agents (Oudeyer & Kaplan, 2007a; Colas et al., 2022c).

Foundational work in psychology, notably by Vygotsky (1934), and more recently by Scott Weng Fai (2011) and Lidayan et al. (2025), has suggested that private speech—i.e., internal dialogue—plays a critical role in children's creative problem-solving. These findings support a broader understanding of language as a cognitive tool that shapes learning and exploration, particularly in creative processes (Guentulle et al., 2024). Drawing inspiration from these insights, recent advances in artificial intelligence have increasingly incorporated language as a scaffold for reasoning and goal-setting. Within the context of RL, language has demonstrated substantial utility, thanks to its compositionality and generalisation capabilities (Luketina et al., 2019b), often through the use of simple linguistic abstractions (Colas et al., 2019).

The emergence of LLMs has further expanded the role of language in open-ended learning. These models, capable of sophisticated linguistic reasoning, have been used to emulate complex human behaviours, such as selecting interesting goals to pursue (Zhang et al., 2024b; Wang et al., 2024a). Beyond goal selection, LLMs have been harnessed to drive continuous innovation by generating new variations of existing environments, as illustrated in works like

Mario-GPT (Sudhakaran et al., 2023) and LLM-POET (Aki et al., 2024). Building on this line of research, OMNI-EPIC (Faldor et al., 2025) integrates a Vision Language Model as a reward function, thereby enhancing the diversity and relevance of generated tasks. This approach also incorporates learning progress signals (Kaplan & Oudeyer, 2003) to ensure that the tasks remain both novel and learnable, thus fostering sustained skill acquisition in open-ended settings.

9.2 Contributions

In this preliminary work, we present AutotelicLLM an autotelic agent (see Figure 9.1), that autonomously explores a 2D minecraft-like environment Craftax (Matthews et al., 2024a). AutotelicLLM has two modules: a goal generator, and a goal-conditioned learner that select goals based on learning progress and execute them with goal-conditioned RL-agent. The two modules communicate through a shared goal archive. Then, the difficulty (D) and learnability (L) of those goals are estimated. Inside the goal generator, goals are group into hand crafted categories and sampled based on their learnability (L). Those goals, are used as in-context examples for the LLM-based goal generation alongside a numerical estimate of their difficulty and learnability estimates. We compare the learnability of goals generated by our method to that generated by a OMNI-EPIC-like baseline (an adaptation of OMNI-EPIC to a goal-conditioned agent). Our preliminary experiments imply that our method is able to leverage the learnability estimates to generate a higher percentage of learnable goals.

The main contributions of this work are: 1) a first study of LLM-goal generation for a goal-conditioned (autotelic) RL agent 2) a method leveraging learnability and difficulty to improve goal generation

We explore the following two questions:

- **Q1.** Does AutotelicLLM generate diverse goals adapted to the agent abilities?
- **Q2.** Do the handcrafted categories necessary for generating diverse goals?

9.3 Related work

Goals generation

Autonomously discovering a diverse set of achievable goals has proven to be a powerful mechanism for embodied agents. This process plays a fundamental role in the construction of internal world models and the development of reusable behavioural repertoires (Baranes & Oudeyer, 2013), and is particularly advantageous for improving exploration in deep reinforcement learning tasks characterized by sparse or deceptive reward structures (Conti et al., 2018; Colas et al., 2018). A substantial body of work has focused on *autotelic agents*—agents that are intrinsically motivated to generate and pursue novel goals, especially those not previously encountered in the environment. Within the framework of goal-conditioned reinforcement learning, MUGL (Laversanne-Finot et al., 2018b) proposes sampling goals from a disentangled latent space learned via a variational autoencoder, while IMAGINE

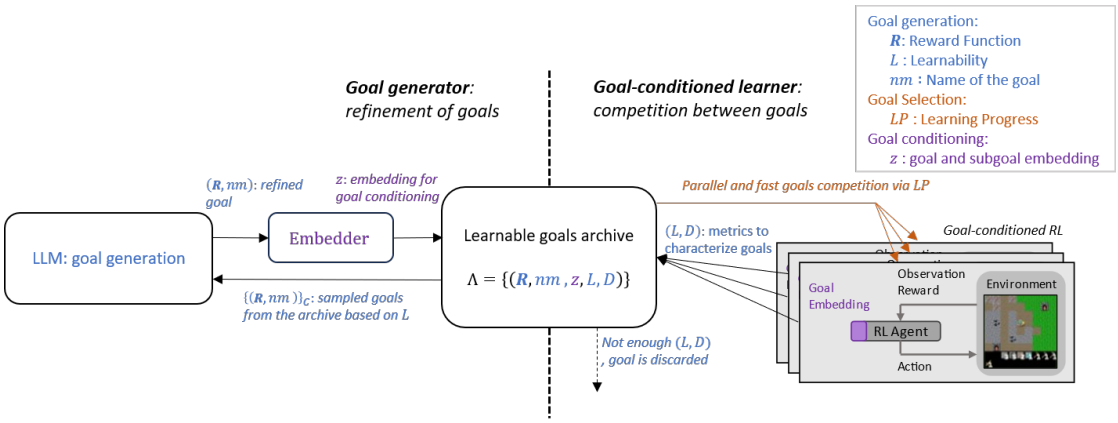


Figure 9.1: The autotelic agent architecture consists of two main modules: the goal generator and the goal-conditioned learner, along with a goal archive. In the goal generator, goals are sampled based on their learnability. These goals, their learnability (L) and difficulty (D) estimates, are given to LLM to generate new goals: reward functions (R) and names (nm). Names are embedded for goal conditioning of the RL agent. In the goal-conditioned learner, goals for training the RL agent are sampled based on Learning Progress (LP), and their learnability (L) and difficulty (D) are estimated.

(Colas et al., 2020a) exploits the compositional properties of language to construct new goal descriptions. More recently, LLMs have been deployed to autonomously generate novel goals in text-based environments (Colas et al., 2023) and interactive 3D settings such as Minecraft (Wang et al., 2023a), where LLMs act both as planners and generators of new objectives. In contrast, AutotelicLLM leverages the generative capabilities of LLMs not only to produce goal descriptions, but crucially integrates a learning progress signal over the existing set of generated goals to guide exploration. This mechanism enables the agent to strategically focus on regions of the goal space that are both novel and learnable, providing a more structured and efficient open-ended learning process.

Environment generation

A parallel line of research has focused on generating curricula not merely in the form of tasks or goals, but as entire environments, with the objective of presenting RL agents with a structured sequence of challenges that facilitate progressive learning (Dennis et al., 2020; Jiang et al., 2021c; Portelas et al., 2020e). In this context, ALP-GMM(Portelas et al., 2020a) applies Absolute Learning Progress (ALP) to tailor a curriculum for a bipedal walker by dynamically adjusting environmental parameters such as obstacle height and separation. POET(Wang et al., 2019, 2020) advances this idea by co-evolving a population of agents and their respective environments, allowing each agent-environment pair to adapt and increase in complexity over time. Other approaches have investigated morphology-aware curriculum generation, such as TeachMyAgent(Portelas et al., 2020c), which selects environment types—e.g., aquatic vs. terrestrial—based on the agent's physical embodiment. More recently, LLMs have been employed to automate environment design. LLM-POET(Aki et al., 2024) integrates LLMs to generate environments for bipedal locomotion tasks, effectively expanding the diversity of learning scenarios. However, these methods often restrict the agent's exploration to a predefined set of environmental conditions. In contrast, OMNI-EPIC (Faldor et al.,

2025) combines an LLM with a vision-language model to generate both environments and reward functions directly as executable Python code, offering a more open-ended and flexible learning setup. AutotelicLLM similarly leverages LLMs to generate goal-reward pairs, but differs from OMNI-EPIC by focusing on single-agent learning rather than co-evolution across populations. Furthermore, AutotelicLLM integrates a learning progress signal to adaptively guide goal generation, ensuring that the agent is continuously challenged in a way that fosters both novelty and learnability.

9.4 AutotelicLLM: Generating creative and learnable goal with a Large Language Model

Problem statement

We conduct our study in a goal-augmented MDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{G}, \mathcal{R}, \gamma)$, with $\mathcal{S}$ a set of states, $\mathcal{T}$ the transition function, $\mathcal{A}$ the action space, G the goal space, R the space of binary reward functions indicating whether a state s satisfies a goal $g \in \mathcal{G}$ and γ the discount factor. In our setting, a goal g is a tuple $g = (nm_g, \mathcal{R}_g)$, where $nm_g \in \mathcal{V}^{L_{nm_g}}$ is the name of the goal, $\mathcal{V}$ the vocabulary and $\mathcal{R}_g \in \mathcal{R} \subset \mathcal{V}^{L_\mathcal{R}}$ is the associated reward function in the form of a code. Here $(L_{nm_g}, L_\mathcal{R}) \in \mathbb{N}^2$ are respectively the maximum length of the goal name and maximum length of the code for the reward function corresponding to the goal. The agent is model by a goal condition policy $\pi : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$ and we define the history of the agent as

$$\mathcal{H}_{ist} = \{(g, SR(g)), \forall g \in \mathcal{G}_{att}\},$$

where $\mathcal{G}_{att} \subseteq \mathcal{G}$ the set of goals attempted by the agent.

We define the learnability L of a goal $g \in \mathcal{G}$ between a time step k_{init} and a time step k_{end} as:

$$L(g, k_{init}, k_{end}) = \max_{k \in [|k_{init}, k_{end}|]} (SR(k, g) - SR(k_{init}, g)),$$

where k_{init} is the step at which the agent has trained on g for the first time and $SR(k, g)$ is the success rate for goal g at step k.

Be $\mathcal{E} : \mathcal{V}^{L_{nm_g}} \to \mathbb{R}^{n_e}$ an embedding function that map the semantic description nm_g of a goal g to a vector in $\mathbb{R}^{n_e}$. We define the distance between two goals $(g_i, g_j) \in \mathcal{G}^2$ as:

$$d_{\text{emb}}(g_i, g_j) = ||\mathcal{E}(n_{g_i}) - \mathcal{E}(n_{g_j})||_2,$$

where $||\cdot||_2$ the L^2 norm.

The objective is to design a goal generator function $\mathfrak{G}$ that takes $\mathcal{H}_{ist}$ and generates a new goal $g \in \mathcal{G}$ such that g maximize both the learnability (a goal on which the agent can do a lot of progress, neither to easy or too difficult) and the diversity by maximising $d_{\text{emb}}(g, g')$ for all $g' \in \mathcal{G}_{gen}$, where $\mathcal{G}_{gen} \subseteq \mathcal{G}$ the set of all goals already generated by $\mathfrak{G}$.

Method overview

The agent operates through two distinct modules (Fig.9.1) to facilitate the open-ended generation and learning of goals. In the first module, an LLM model by adaptively generating goals based on the agent's current skill level. All tasks are stored in an archive Λ that approximates $\mathcal{H}_{ist}$. In the second module, a multi-goal deep RL agent learns the tasks from the archive.

The process is initialized by adding a set of hand-crafted goals to the archive. The manually created goals are very simple and allow the LLM to learn how to generate syntactically correct reward functions. The multi-goal agent is then trained on these goals for a few updates, involving data collection steps and weight updates. Then, based on the success of the agent in learning the different tasks, the archive is updated: tasks that are too difficult or too easy to learn are removed from the archive. This iterative process allows the model to continuously generate and adapt goals that align with the agent's evolving capabilities, fostering an open-ended learning environment. In the next two sections, we first present how the goal generator is modeled by an LLM, then how goal condition policy is trained on the generated goals The pseudo-algorithm for the process described above is given in Algorithm 2.

The goal generator

$\mathcal{H}_{ist}$ can become too big to be used in practice, thus we approximate it with Λ. This set contains at most n_Λ tuples of the form $(nm_g, \mathcal{E}(nm_g), \mathcal{R}_g, L(g), D(g), c)$ where c is a handcrafted category such as "collecting objects" and $D(g) = 1 - SR(g)$ is the difficulty of a goal. It is important to note that even if the difficulty and the learnability of a goal shared similarities, they measure two different properties. If we consider a goal that has a probability of 0.5 of being reachable, at best the policy would have a success rate of 0.5, resulting in a difficulty of 0.5 but a future learnability of 0.

We model the goal generator using an LLM, which is prompted with in-context examples of goals and their associated learnability $L(g, k_{init})$ and difficulty $D(g)$, where k_{init} correspond to the time g has been generated. These examples inform the LLM about the capacities of the goal-conditioned agent and are selected in Λ using a filtering function $\mathcal{F}_\Lambda$ that filters examples based on their learnability, difficulty, $\mathcal{F}_\Lambda$ only returns examples from the same category. $\mathcal{F}_\Lambda$ is fully described in Appendix D.1.1.The full prompt is given in Appendix D.4. The goal generator proposes new goals or modifies existing ones to maximize the estimated L and D (the tasks, although difficult, must be learnable). The L metric helps selecting learnable tasks from a larger pool of non-learnable ones, while the difficulty metric filters out tasks that are too simple. All generated goals are assigned the category used for the examples that have served to generate them.

After generation, a procedure (see Appendix D.1.3) removes goals g whose reward functions $\mathcal{R}$ are not syntactically correct (that do not compile). $\mathcal{R}$ can use privileged information such as the current state of the game engine, the agent's current action, and a reward state where memory can be stored. This allows the agent to target challenging goals such as

time-extended goals (e.g., "move up three times") and goals involving optimization under selected constraints (e.g., "build a shelter while maintaining your health above 5"). Examples of such reward code are provided in Appendix D.3. The generated goals are then added to Λ.

The goal conditioned learner

While any RL algorithm could be used, we train the goal-conditioned agent on goals in Λ using PPO. Appendix D.2.1 details the hyperparameters used for this training. The agent is a causal transformer (Dai et al., 2019), conditioned on both textual and visual information. The textual information is the embedding of the nm_g, $\mathcal{E}(nm_g) = z \in \mathbb{R}^{n_e}$, with n_e being the dimensionality of the embedding space. The visual information consists of the last n_{obs} images returned by the environment. The goals are sampled in the archive Λ following a distribution $\mathcal{D}_\Lambda$ based on the LP (i.e. improvement of agent's own ability to solve goals (Oudeyer & Kaplan, 2007b)) on each goal. At the end of each trajectory the learning progress of a goal is updated using the formula given in OMNI (Zhang et al., 2024b).

After each training session (a data collection and weight update), all the goals in the archive are ranked, and the n_{best} are kept in Λ. To class the goals we use a fitness function $f(g) = L_g \times D_g$, where L_g is the learnability of g and D_g its difficulty.

Algorithm 2 Autotelic open-ended goal generation

Input: Goal archive $\Lambda = \{(nm, z, \mathcal{R}, L_{ini}, D_{ini}, c)\}$ with hand-made goals, $\mathcal{E}$ an embedding function, $\mathfrak{G}$ a goal generation function
Initialize: Initialize policy π
for *generation* $= 1, 2, \ldots, G$ **do**
  for *update* : $k = 1, 2, \ldots, K$ **do**
    Sample a batch B from the goal selection distribution, $\{(z, \mathcal{R})\}_B \sim \mathcal{D}_\Lambda$
    Collect p steps of policy rollout $\{\tau\}_B$ per sampled goals $\{(z, \mathcal{R})\}_B$
    Evaluate the goal-specific success rate $\{SR_k\}$
    Update the goal-specific Learning Progress $\{LP_k\}$ using the formula in Zhang et al. (2023a)
    Update the goal-specific Learnability and Difficulty $\{(L_k, D_k)\}_B$
    Update the goal-condition policy π with $\{\mathcal{R}(\tau)\}_B$ and $\{z\}_B$ using PPO
  Update the Archive Λ with the goal learnability and difficulty $\{L_K, D_K\}$
  Update the archive by keeping the N-fittest goals $\Lambda \leftarrow \{\lambda \in \Lambda | rank(\lambda, L_\lambda \times D_\lambda) \leq N\}$
  for $i = 1, 2, \ldots, I$ **do**
    Sample a goals from the archive $((nm, \mathcal{R}))_C \sim \mathcal{F}_\Lambda$
    Generate a new goal $(nm, \mathcal{R}) \sim \mathfrak{G}((nm, \mathcal{R})_C, G_{prompt})$
    Create Embedding $z_g = \mathcal{E}(nm_g)$
    Test validity of the goal generated (syntax+compilation)
    if valid goal **then**
      Add the goal to the archive $\Lambda \leftarrow \Lambda \cup (nm, z, \mathcal{R}, L_{ini}, D_{ini})$

9.5 Experiments

Environment

We conduct our preliminary experiments on a 2D Minecraft-like environment, Crafter (Hafner, 2022) with a fast JAX implementation called Craftax (Matthews et al., 2024a). The environment contains various materials (e.g. wood, stone) and mobs (creatures such as zombie, skeleton, cow, etc.). Each observation is a 8x8 semantic grid and the action space contains actions for movement, attacking, and crafting. Crafting enables the building of additional objects (e.g. table pickaxe).

Baseline

The most similar related work is OMNI-EPIC (Faldor et al., 2025), with a key conceptual distinction. In OMNI-EPIC, for each new environment, the RL agent corresponding to the closest environment is cloned and fine-tuned, i.e. they create a lookup table of specialized agents as opposed to one generalist goal-conditioned agent. Moreover, they only use the difficulty D to rank and choose the examples used for goal generation. We adapt there set up to ours by using OMNI-EPIC on our goal conditioned agent.

Metrics

To better understand AutotelicLLM we want to know: How does the difficulty of the generated goals evolve? Are the generated goals learnable by the agent? How does the diversity of the generated goals evolve? To this end we define three metrics, each one providing information on a different aspect of the method:

Average Difficulty: The average difficulty of the goals generated during the update i is defined as:

$$\bar{D} = \frac{1}{|\mathcal{G}_i|} \sum_{g \in \mathcal{G}_i} (1 - SR_{\text{scratch}}(g)),$$

where $\mathcal{G}_i$ is the set of goals generated during the update i and SR_{scratch} the success rate of the agent trained from scratch.

Cumulative Learnability: The cumulative learnability CL measures how much learning has occurred in the tasks since the beginning of training. It is expressed as the sum over the generated goals of the maximum progress, where the progress for a goal g at step k is the difference between the success rate of the agent at step k, $SR(k, g)$, and the one when the goal have been generated $SR(k_{init}, g)$:

$$CL = \sum_{g \in \cup_{j=0}^{i} \mathcal{G}_j} \max_{k \in [k_{init}, i]} (SR(k, g) - SR(k_{init}, g)).$$

This metric increases monotonically with time and if at some point the goals are not learnable anymore it remains constant.

Diversity: This metric expresses how the goals contained in the archive at step k are semantically diverse. It is defined as the average pairwise distance between embedding of the goals.

$$Div = \frac{1}{|\Lambda_k|^2} \sum_{(g_i, g_j) \in \Lambda_k^2} d_{emb}(g_i, g_j,)$$

where Λ_k, at step k and d_{emb} is the L^2 norm between two embeddings of the goals.

Set up

To study Q1 and Q2, we train an agent with PPO on goals generated by AutotelicLLM. The agent is trained for $5,000$ data collection turns. After each data collection the agent is updated for 4 epochs. For each data collection, we collect of 128 steps from 1024 parallelized environments, so 131072 steps in total. We use a JAX implementation of PPO with the transformerXL architecture (Hamon, 2024). Every 500 PPO data collection the archive Λ that contains the goals is updated by AutotelicLLM, the archive can at most contain 200 goals and is initialised with 33 goals divided into 6 categories: *Crafting tools*, *Combat*, *Collecting/Finding Things*, *Construction (shelter, bridges)*, *Spatial Navigation/Movement*, *Sequence of goals* (multiple goals that are not preconditions for each other). To measure SR_{scratch} in the average difficulty metric, we train from scratch a policy with PPO for 500 iterations of data collection-weight update. The goals are sampled uniformly in the archive. All hyperparameters are given in Appendix D.2.1. For the experiments we use DeepSeek v2.5, for practical reasons since it is a preliminary work we limit the number of seed to 1.

9.5.1 Q1.Does AutotelicLLM generate diverse goals adapted to the agent abilities?

Evolution of the metrics

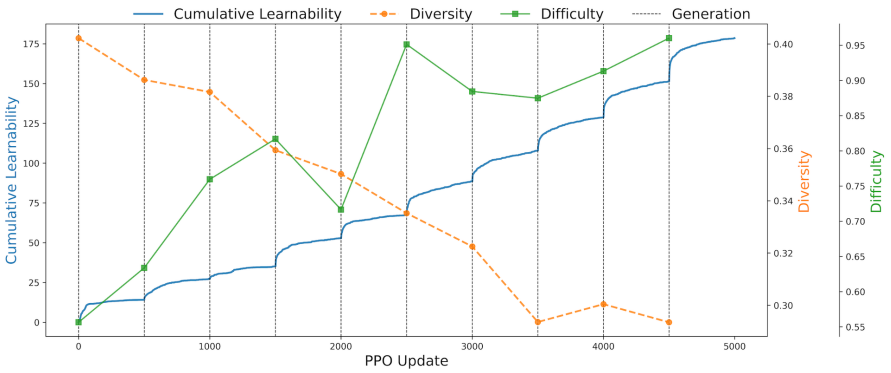


Figure 9.2: Evolution of cumulative learnability, diversity and difficulty with the number of data collection turns.

Looking at the evolution of the three metrics in Figure 9.2, we observe that both difficulty and cumulative learnability increase as training goes on. This shows that during training, as agent progresses, AutoteliCLLM proposes goals adapted to its level (hence the fact that cumulative learnability does not stagnate) and that they are not just repetition of the simple goals from the start (hence the fact that difficulty increases).

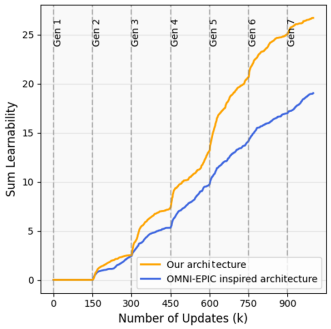


Figure 9.3: Cumulative learnability of all goals generated by our method compared to the OMNI-EPIC-like baseline.

Furthermore, when comparing AutoteliCLLM with the OMNI-EPIC-like baseline in Figure 9.3, we observe that cumulative learnability increases more sharply for AutoteliCLLM. As learnability is null or low for goals that are trivially easy, too hard, or even impossible, it serves as a good estimate of the adaptability of the goal generator to the current capabilities of the learner. Thus, since AutoteliCLLM appears to be better at generating more learnable goals, that implies that it adapts better to the learner compared to OMNI-EPIC.

Finally, we observe that the diversity in the archive decrease through time. We have by design created a limited number of categories where AutoteliCLLM can generate new goals. We interpret this result as the filling of these categories by AutoteliCLLM. At the beginning, 33 goals of the archive are divided into 6 categories, as training progresses, the archive fills up to 200 goals. Once the archive is filled around $3,500$ data collection turns, the diversity remains constant around 0.3. Thus, each categories get filled-up decreasing the average distance between the goal embeddings. Figure 9.4 tends to confirm this interpretation: with each new generation of the archive, we see the embedding of goals within each category becoming closer.

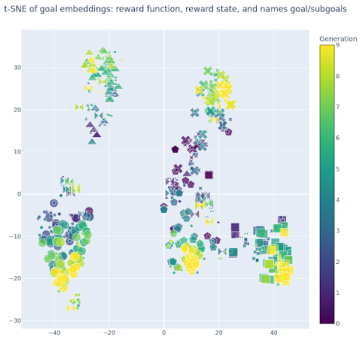


Figure 9.4: t-SNE of the goals embeddings contained in the archive at each generation.

Case study of generated examples

In this section, in order to build intuition of how the exploration evolves, and the limitations of our approach we looked at hand picked examples and present examples of reward functions (names and implementations) discovered during our preliminary experiments. We first observe that simple tasks that align well with the original game code are generated easily with correct semantics. For instance in Figure 9.5-d AutoteliCLLM proposes the goal of *Craft and use wood-swords*, when looking at the code we see that it requires the agent to first *collect wood*, then *make wood sword* then *defeat a mobster*. Such subgoals are intuitively well aligned with the final goals and are composition of the simple original goals found in the archive such as *collect wood*.

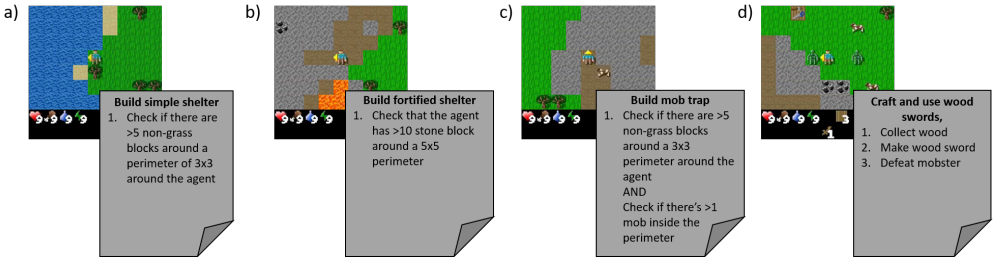
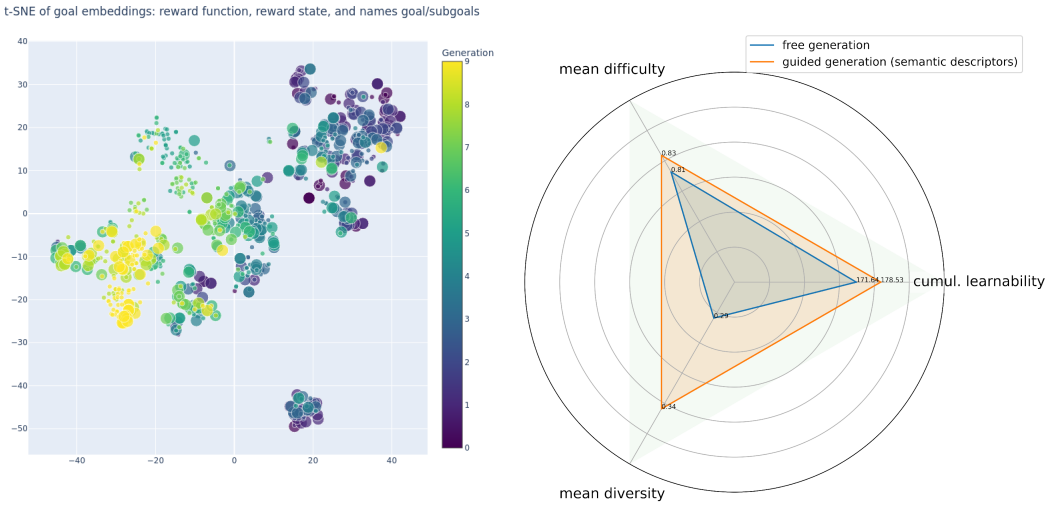


Figure 9.5: Examples of generated reward functions, pseudocode, and visualizations of the observations at the task completion state. Complexity increases from a)-c), but implementations semantically limited (shelters are not enclosed). In d), a more standard (less abstract) minecraft-like goal is implemented well, despite being multi-step.

Furthermore, we observe implications that our architecture can generate increasingly complex goals. In the Figure 9.5 the goals a-c have been sampled in the archive at 3 different time steps during training. The agent first generates a simple shelter reward function based solely on checking whether there are non-grass blocks in its perimeter (Fig.9.5a). Then, another reward function is created checking for the presence of stones in the perimeter (Fig.9.5b), and finally, a further reward function checks for a "mob trap", which should correspond to a shelter around another character - a mob (Fig.9.5c). However, we note that the LLM fails to adequately match the semantics of those more abstract goals: the shelters created are not enclosed, allowing a mob to always attack the agent, and for a "mob trap" it merely checks if there is a creature close to the agent and more than 5 non-grass objects. The full codes of the generated reward functions are given in Appendix D.3. We believe that these limitations in the LLM's ability to generate reward functions that match the textual description of these goals stem from two factors: first, the code is implemented in JAX, which is a library that, in our experience, seems less well mastered than Python; second, the LLM lacks grounding for tasks that often require taking into account the space around the agent. This lack of spatial grounding in pretrained LLMs was discussed in Chapter 4.

9.5.2 Q2. Do the handcrafted categories necessary for generating diverse goals?

In the previous experiment, we let the LLM generate new goals but gave it examples from the same category. However, we saw that this led to a reduction in diversity within each category (Figure 9.4) and ultimately in overall diversity. In this ablation, we remove any notion of categories in the example given to the LLM for the generation of a new goal. Figure 9.6a shows that over generation the goals freely generated (with example not constrained to be in the same categories) tend to regroup in one or two clusters. This is despite the fact that the examples from generation 0 stored in the archive are diverse and spread across many small clusters. The Figure 9.6b supports this first analysis by showing that in the free generation case the metric of diversity is much more affected compared to the two other metrics. We can hypothesis that some clusters of goals present in the original archive had in average less learning progress at the beginning, the goals manually crafted where either too easy or too hard. Thus, as generation of archive went on they where progressively removed from it, as they where less numerous they also represent less examples for the LLM to get inspired from



(a) t-SNE of the embeddings of the goals generated whitout using goal categories.

(b) Radar plot for the three metrics denied in Section 9.5.

Figure 9.6: Ablation between guided generation (using examples from similar predefined categories) and free generation (where the examples in the prompt are not group by categories).

to build new goals. IN reverse the clusters of goals which have from the beginning a good learning progress where selectively keep and where used to generate new goals who where biased to be part of the same categories as their "parents" examples. Thus the imposed hand crafted categories allow for less successful categories to have a chance to develop suitable goals for the agent akin to a speciation bias (Stanley & Miikkulainen, 2002).

9.6 Discussion

In this chapter, we presented AutotelicLLM, a novel method in which an autotelic agent leverages a LLM to autonomously generate new goals. These goals are articulated as short textual descriptions, each accompanied by a corresponding reward function. Central to our approach is the use of learnability estimates to select in-context examples for the LLM, ensuring that the generation process is guided by previously evaluated goal difficulty and learnability scores. This allows the agent to iteratively produce goals that are both novel and adapted to its evolving capabilities.

To evaluate how well the generated goals align with the agent's learning progress, we introduced a cumulative learnability metric. Our preliminary results, benchmarked against a baseline inspired by OMNI-EPIC, suggest that AutotelicLLM generates goals that are more learnable and better adapted to the agent's current skill level. Notably, we observed a progressive increase in goal difficulty—as measured by an untrained agent—while maintaining high instantaneous learnability, indicating that the goals remain appropriately challenging as the agent continues to learn.

Nevertheless, several limitations emerged in this initial implementation of AutotelicLLM. First, our method currently relies on expert-defined goal categories to preserve goal diversity

throughout training. In the absence of such categories, we observed a collapse in diversity over time, with only a few original goal types persisting. A promising direction to address this issue involves preserving a minimal number of goals from each class to prevent irreversible loss of diversity, as rediscovering entire goal classes is a non-trivial challenge for the system. Additionally, we identified discrepancies between the generated reward functions and their corresponding textual goal descriptions—a form of specification gaming or "hacking." This undermines the coherence between goal intent and execution. A potential solution lies in introducing retroactive validation mechanisms, where generated reward functions are evaluated for alignment with their textual descriptions, and misaligned examples are filtered out. Finally, to strengthen the conclusions of our experiments, we should test our method for several seeds and conduct further experiments with the OMNI-EPIC-like baselines.

Despite these limitations, our study demonstrates the feasibility and promise of using LLMs to generate goals that are both novel and well-suited for autotelic agents. AutotelicLLM is a step toward more autonomous and creative forms of goal space exploration, where agents continuously invent and pursue new goals in an open-ended manner.

Chapter 10

Discussion

In this part, we have presented two language-based methods to explore the goal space. These methods harness the linguistic capabilities of LLMs and their internet-scale common-sense knowledge to enable efficient automatic curriculum learning strategies (MAGELLAN) and to discover learnable and creative new goals (AutotelicLLM).

In Chapter 8, we introduced MAGELLAN, a method equipping LLM agents with a metacognitive mechanism that provides them with awareness of their own competence and learning progress. Just as humans reflect on what they know and what they should learn next, MAGELLAN allows LLM agents to monitor their own learning trajectories and autonomously select which goals to pursue. This endows agents with intrinsic curiosity and promotes self-driven growth through informed goal selection. We conducted a comprehensive study of MAGELLAN along four axes. First, we showed that it allows accurate competence prediction at a significantly lower computational cost compared to classical baselines. Second, we demonstrated that this estimation facilitates an efficient curriculum over the goal set, rivaling expert-informed baselines. Third, MAGELLAN generalizes robustly to unseen goals and produces a goal embedding space structured by both semantic similarity and competence. Lastly, the method quickly adapts to changes in the goal space—an essential property for open-ended agents continuously generating new goals during exploration. However, despite its efficiency, MAGELLAN still requires periodic competence estimation across all discovered goals, which may become computationally prohibitive as the goal space grows unbounded. A potential solution lies in automatically clustering similar goals and computing a shared competence estimate, thereby avoiding redundant computations on individually irrelevant goals.

Chapter 9 presents a preliminary study on the AutotelicLLM method, in which an autotelic agent leverages an LLM to generate novel goals, expressed as short textual descriptions paired with executable reward functions. AutotelicLLM filters goals based on estimated learnability, using this to construct informative in-context examples for subsequent goal generation. Compared to a baseline inspired by OMNI-EPIC (Faldor et al., 2025), AutotelicLLM produces goals that are more learnable and gradually more complex. However, it currently relies on expert-defined categories for goal generation; in their absence, we observe a collapse in diversity, with only a few categories persisting. Thus, the creativity of AutotelicLLM remains constrained by the breadth of expert input. Future research may focus on automatic category discovery to unlock the full generative potential of this approach.

Together, these two chapters present complementary strategies for exploring and managing goal spaces. AutotelicLLM focuses on expanding the space with creative and learnable new goals, while MAGELLAN provides a means for agents to assess their competence and organize their learning within that space. These methods address two fundamental challenges in open-ended learning: generating new goals in a meaningful and structured way, and evaluating an agent's competence across an ever-growing and complex goal space while maintaining computational feasibility. Over time, an open-ended agent should generate increasingly difficult and complex goals, as we observed in Chapter 9. However, if the time required to master each goal scales similarly, the learning process may decelerate to the point where meaningful progress stalls. In the following part, we turn our attention to two language-based methods designed to accelerate the learning process of reinforcement learning agents.

Part IV

Efficient learning of new goals

Chapter 11

Introduction

In the preceding two parts, we examined two essential capabilities for the development of open-ended agents. First, we investigated the importance of modelling and internalising environmental dynamics, which enables agents to interpret and exploit their surroundings effectively. Second, we studied the notion of goal space exploration, wherein agents autonomously select or generate goals that maximise their learning progress. These two dimensions—understanding the environment and identifying suitable objectives—constitute foundational pillars of open-ended learning.

However, a third and equally critical competency must be addressed: the ability to learn increasingly complex goals efficiently. As an open-ended agent matures, the goals it encounters are expected to become progressively more sophisticated. If the time required to learn each new goal increases disproportionately, the overall pace of learning will inevitably stagnate. This trajectory is contrary to human development, where individuals routinely acquire more intricate skills without a corresponding slowdown in learning progress. This suggests the presence of mechanisms that support efficient goal acquisition even in the face of rising complexity.

Language, as previously discussed, offers valuable properties—such as generalisation and compositionality—that are highly beneficial in reinforcement learning contexts (Gopnik & Meltzoff, 1987; Luketina et al., 2019a). In language-conditioned reinforcement learning, where goals are expressed in natural language, these properties could, in principle, support the efficient learning of complex tasks. Yet, in practice, the use of language in RL environments such as BabyAI, LittleZoo, and Crafter often involves sparse rewards, which hamper effective credit assignment and render traditional RL algorithms less effective. As tasks grow in complexity—typically involving longer trajectories and delayed feedback—the challenge of learning from sparse supervision becomes more pronounced, ultimately impeding learning efficiency.

In this final part, we address this issue by investigating two complementary, language-based strategies designed to improve learning efficiency in the face of increasing task complexity. In Chapter 12, we propose EAGER a method of reward shaping that leverages the structure of language. Specifically, we use a system of fill-in-the-blank questions and their corresponding answers to provide intermediate supervision, thereby guiding the agent toward its ultimate objective with denser feedback signals.

Chapter 13 turns to the paradigm of hierarchical reinforcement learning. We propose an unpublished work where we introduce HERAKLES, a method in which an LLM serves as a high-level policy, responsible for selecting skills from a library mastered by a low-level policy. As new tasks are learned, they are abstracted and incorporated into the low-level skill set, allowing the LLM to reuse them as building blocks for solving more complex objectives. This progressive accumulation of reusable skills enables the agent to maintain high learning efficiency even as the complexity of its goals increases.

These contributions aim to address a critical bottleneck in open-ended learning: sustaining the agent's capacity to acquire increasingly difficult goals without diminishing returns. Through the integration of language-based reward shaping and hierarchical skill composition, we take further steps toward the realization of agents capable of lifelong, open-ended learning.

Chapter 12

Using pure linguistic abstraction to generate a reward signal

12.1 Motivation

One of the main challenges of RL research is to train agents able of abstraction, generalisation and communication. Languages, be they natural or formal, afford these desirable properties (Gopnik & Meltzoff, 1987). Based on this insight, many papers have tried to leverage the abilities of language in RL to enable communication and improve generalisation and sample efficiency (Andreas et al., 2017; Mei et al., 2016a; Goyal et al., 2019; Xu et al., 2022). The domain can be subdivided into language-conditioned RL (LC-RL), in which language conditions the formulation of the problem (Anderson et al., 2018; Goyal et al., 2019), and language-assisted RL, where language helps the agent to learn (Hu et al., 2019; Colas et al., 2020a; Akakzia et al., 2021; Colas et al., 2022b). In the present chapter, we focus on the LC-RL framework where the agent initially receives a language instruction and must act to optimise the corresponding reward function. Unfortunately, the corresponding RL algorithms are sample inefficient, especially due to the fact that the reward function is sparse when it is restricted to completing the goal.

To tackle the reward sparsity issue, one idea is to densify the reward by decomposing the general goal into sub-goals and rewarding them individually. This idea is based on a decomposition principle which hypothesises that a general goal can be decomposed into a set of easier ones. Previous works assume a strict decomposition of the general goal into sub-goals (Andreas et al., 2017; Jiang et al., 2019). However, strict decomposition requires that that each high-level goal can be reached through an exact series of low-level policies, an assumption which fails when all the primitive actions are not known in advance. More recently, the ELLA method was proposed as a language reward shaping technique that is sample efficient and does not require a strict decomposition: it rewards interesting auxiliary objectives, without requiring rigid ordering and correspondence with sub-goals (Mirchandani et al., 2021).

However, all these methods suffer from the need for expert input. Indeed, they require task-specific engineering for determining at least one of the following elements: the set of sub-goals or auxiliary objectives, the relevant ones and when they are achieved, and

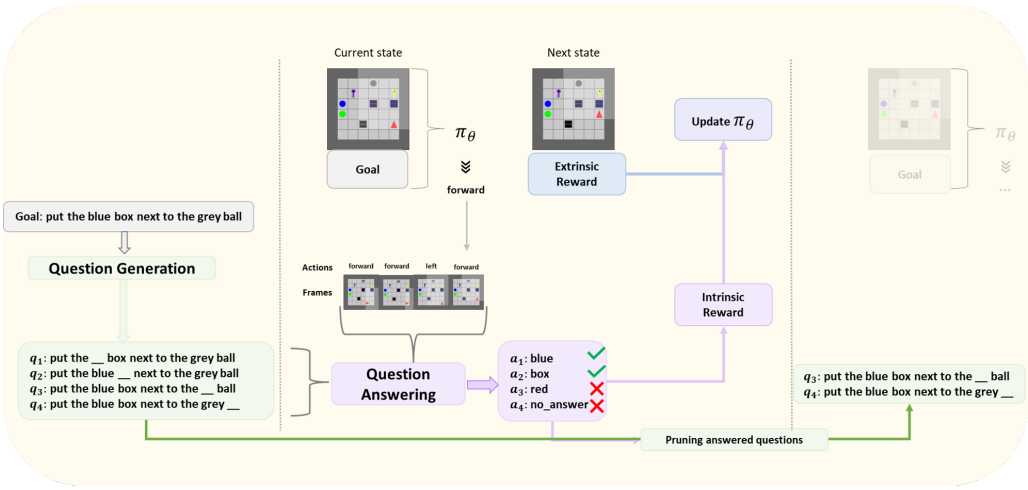


Figure 12.1: During training, the agent uses the goal to generate relevant questions using its question-generation module **QG**. Then, it attempts at answering them from current trajectories at each step with its question-answering module **QA**, by looking at the trajectory. When it succeeds, it obtains an intrinsic reward proportional to its confidence in its answer. Then it removes the answered questions from the list of questions. This incentivizes the agent to produce trajectories that enable to reconstruct unambiguously partial information about the general language goal, enabling to shape rewards and guide learning.

the appropriate reward. While all these points can be addressed more or less easily in simple game-like environments, they get demanding when the environment is more complex. Therefore, it seems desirable to find an alternative reward shaping approach that minimises expert involvement. We would like the agent to generate its own auxiliary objectives and create intrinsic rewards, just like a human would do.

The Natural Language Processing (NLP) field often suffers from the same need of expert input, e.g. for the evaluation of automatic summary tasks. In that context, some successful approaches have developed reference-less metrics (Scialom et al., 2021), (Rebuffel et al., 2021), based on question generation (QG) and question answering (QA). These techniques assess the quality of a generated text by measuring the quantity of information from the source conserved in the generated text (broadly, how well one can answer questions about the original text using the generated text). These metrics are said reference-less because they do not require a comparison with a man-made example to evaluate the quality of a text.

12.2 Contributions

In our work, we build on these reference-less metrics to circumvent the need of expert input for generating auxiliary objectives. We adapt it and propose a novel QG/QA framework for RL called EAGER.[1] In EAGER, an agent reuses the initial language goal sentence to generate a set of questions (QG): each of these self-generated questions defines an auxiliary objective. Here, generating a question consists in masking a word of the initial language goal.

[1]Exploit question-Answering Grounding for effective Exploration in language-conditioned Reinforcement learning, see https://github.com/flowersteam/EAGER for access to the code.

Then the agent tries to answer these questions (guess the missing word) only by observing its trajectory so far. When it manages to answer a question correctly (QA) it obtains an intrinsic reward proportional to its confidence in the answer. The QA module is trained using a set of successful example trajectories. If the agent follows a path too different from correct ones at some point in its trajectory, the QA module will not answer the question correctly, resulting in zero intrinsic reward. The sum of all the intrinsic rewards measures the quality of a trajectory in relation to the given goal. In other words, maximizing this intrinsic reward incentivizes the agent to produce behaviour that unambiguously explains various aspects of the given goal.

To the best of our knowledge, EAGER is the only framework that can automatically 1) generate relevant auxiliary objectives, 2) determine their completion and 3) return the appropriate intrinsic reward. This approach only assumes the agent has access to a dataset of demonstrated behaviours associated to global language commands, which enables it to pre-train its question answering module.

Thus our work brings the following contributions:

• We create a QG/QA metric providing to an agent an information-rich measure of the quality of its trajectory given a goal.

• We propose the EAGER framework that lets the agent guide its own learning process by generating auxiliary objectives and producing intrinsic rewards without requiring any expert intervention.

• We show that EAGER retains the good properties of ELLA without requiring task-specific expert knowledge, by leveraging properties of language.

• We experiment EAGER with the BabyAI platform (Chevalier-Boisvert et al., 2019): we compare our approach against ELLA (SOTA on BabyAI) and RIDE (a non-language based reward shaping approach using intrinsic motivation), showing its robustness and sample efficiency. Furthermore, although we use example trajectories to train the QA, their use is much more parsimonious than training an agent using behavioural cloning (Chevalier-Boisvert et al., 2019), as we show in Appendix E.3.

12.3 Related work

Language-conditioned RL.

We place our work in the LC-RL setting, where an agent learns a policy to execute language commands (Misra et al., 2017; Chevalier-Boisvert et al., 2019; Luketina et al., 2019a; Lynch & Sermanet, 2020). We reuse the BabyAI platform (Chevalier-Boisvert et al., 2019), widely used in this domain as it enables to decouple exploration challenges from perception challenges. It uses a synthetic language exhibiting interesting combinatorial properties with possible conjunction of properties, and procedural generation to avoid overfitting (Cobbe et al., 2020). Here, we consider *instruction following* agents which receive external instructions and rewards (Hermann et al., 2017; **?**; Jiang et al., 2019).

Language as an abstraction in hierarchical RL.

Several approaches leverage language for abstraction in hierarchical RL. One approach uses language for training a low-level instruction-following policy, then learns a high-level policy that generates the sequence of low-level goals (Jiang et al., 2019). Another one explicitly decomposes high-level tasks into low-level ones as in policy sketches (Andreas et al., 2017). ELLA (Mirchandani et al., 2021) also uses language to decompose high-level tasks but relaxes the strict decomposition constraint by replacing sub-goals with auxiliary objectives. We adopt the same flexible framework in our approach but automatise the decomposition.

Language for exploration and reward shaping.

Reward shaping is a form of guidance that supplies additional rewards to the agent to direct its learning process. Among approaches studying how language can shape rewards and exploration, LEARN (Goyal et al., 2019) proposes to map intermediate natural language instruction to intermediate rewards. Similarly, Waytowich et al. (2019) enable reward shaping using natural language through a narration-guided method. The high-level tasks are decomposed into low-level tasks and rewarded using narration. ELLA (Mirchandani et al., 2021) is positioned in the same paradigm but with fewer assumptions about the environment or the structure of the task.

Other approaches assume that an oracle provides language descriptions of environment states which are used as state abstraction to generate novelty intrinsic rewards and guide exploration (Mu et al., 2022; Tam et al., 2022). This extends classical approaches to intrinsically motivated exploration (Aubret et al., 2019), such as count-based methods (Bellemare et al., 2016) or RIDE (Raileanu & Rocktäschel, 2020). Here, we also use language to generate intrinsic rewards, but we do not need language descriptions of states. Besides, other forms of intrinsic motivation systems, such as IMAGINE (Colas et al., 2020a), first learn language-parameterized reward functions through interaction with a social peer, then autonomously use language to generate diverse and novel goals, using the learned reward function for self-supervision.

Asking questions in RL.

Beyond reward shaping, some methods consider agents that use language to ask questions to external knowledge sources. In QWA (Xu et al., 2022), questions are used to identify sub-tasks and prune irrelevant actions. In AFK (Liu et al., 2022), questions are used to obtain world knowledge that helps completing tasks.

Natural Language Processing

One of the sources of QG/QA methods is the thriving field of question generation from natural language processing and information retrieval (Jain et al., 2018). Our approach is inspired from text generation methods where QG and QA are used to measure the quality

of a generated text without using a human reference (Scialom et al., 2021; Rebuffel et al., 2021).

12.4 EAGER: Asking and Answering Questions for Automatic Reward Shaping in Language-guided RL

12.4.1 Problem statement

We set ourselves in the standard framework of LC-RL with an augmented Partial Observation Markov Decision Process (Sutton & Barto, 2018a) $\mathcal{M}$ defined by the tuple ($\mathcal{S}$, $\mathcal{A}$, Ω, $\mathcal{T}$, O, $\mathcal{G}$, $\mathcal{V}$, $\mathcal{I}$, $\mathcal{R}$, γ), with $\mathcal{S}$ the state space, Ω the observation space, $\mathcal{A}$ the actions space, $\mathcal{G}$ the goal space, $\mathcal{V}$ the vocabulary of goal instructions. O stands as the observation function $O : \mathcal{S} \to \Omega$, which maps states to the observations space. $\mathcal{I}$ is the instruction function $\mathcal{I} : \mathcal{G} \to \mathcal{V}^{isize}$, which maps any goal in $\mathcal{G}$ to the set of language instructions, which correspond to sequences of $isize$ symbols (the empty symbol ϵ belongs to $\mathcal{V}$ to allow variable instruction sizes). $\mathcal{R}$ is a goal-conditioned state action reward function, with $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to [0, 1]$ the extrinsic reward received for some goal $g \in \mathcal{G}$ and γ the discount factor. For simplicity, we note in the following $r_t^g = \mathcal{R}(s_t, a_t, g)$ as the reward obtained at step t of any episode with goal g.

At each time step t, the agent receives an observation $o_t \in \Omega$ following the observation function $O : \mathcal{S} \to \Omega$ and selects an action $a_t \in \mathcal{A}$ to reach a goal $g \in \mathcal{G}$, expressed by $\omega^g = \mathcal{I}(g)$. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathbb{P}(\mathcal{S})$ is the transition function. Using RL, we search for an optimal goal-conditioned policy π^*, such that $\pi^* : \mathcal{S} \times \mathcal{V}^{isize} \to \mathcal{A}$ maximises the discounted expected return $G_t^{\pi_g} = \mathbb{E}_\pi[\sum_{k=0}^{T} \gamma^k r_{t+k+1}^g]$. We consider in this work sparse reward problems, where $r(s, a, g)$ returns 1 for any state s such that $d(s, g) \leq \epsilon$, for a given distance function d and a specified threshold ϵ, and 0 otherwise. Moreover, we assume a limited number of steps at most T steps. These two conditions result in a hard exploration problem.

To deal with those problems, many methods aim at densifying rewards, by focusing on auxiliary objectives during training, whose accomplishment can help the agent to reach the goal g at hand. In previous work, such as ELLA (Mirchandani et al., 2021), the selection of relevant objectives required the intervention of an expert (in the form of expert annotations and example trajectories), which can be problematic because new expert trade-offs have to be established for each new environment. Thus, an automated way must be found to recover the relevant auxiliary objectives, measure their completion, and associate the appropriate reward.

Rather than relying on expert knowledge for defining auxiliary objectives, we assume that we have access to a set of trajectories of successful examples coupled to their respective instructions $\{(\tau_0, \omega_0^g), ..., (\tau_n, \omega_n^g)\}$, where $\tau_n = (o_i, a_i)_{i \in [|0,k|]}$ with k the number of steps. For any goal instruction ω^g, we consider a function f that aims at generating a set of auxiliary goals of g, such that all $g' \in f(\omega^g)$ belong to $\mathcal{G}^g$, with $\mathcal{G}^g \subset \mathcal{G}$ the set of goals that help training the agent towards g. Then, for a trajectory τ and any $g' \in f(\omega^g)$, a function h_c determines the probability $h_c(\tau, g') \in [0, 1]$ that the auxiliary objective g' has been achieved. We train h_c using the example demonstrations such that we can leverage its generalisation abilities to use h_c for unseen trajectories.

Please note that Behavioural Cloning methods also rely on the exploitation of expert trajectories, by learning the policy π_{BC} that maximises the log-likelihood $\mathcal{L}_{BC}$:

$$\mathcal{L}_{BC} = \sum_{i=0}^{n} \log \mathbb{P}(\tau_i) \text{ where } \log \mathbb{P}(\tau) = \sum_{j=0}^{k} \log(\pi_{BC}(a_k|o_k)\mathcal{T}(s_{k+1}|s_k, a_k)).$$

However, the latter uses demonstrations to train an agent to copy an expert based on a data set of example trajectories. This technique is only effective when the agent is close enough to the demonstrated examples. A large number of demonstrations are therefore required to cover the space of states that the agent may encounter. Methods such as GAIL (Jonathan Ho, 2016) can partially circumvent this problem by forcing the agent to stay on known trajectories, but this impairs generalisation. Our method is much more parsimonious in the use of such examples: because it generalises well, the agent can receive intrinsic rewards even in areas not encountered in the expert demonstrations (see Appendix E.3).

12.4.2 EAGER

Method overview

We need an automatic evaluation method that is fine-grained enough to rank various trajectories depending on a language instruction. But various successful policies can generate a set of valid trajectories for the same goal. It could be deceiving to rank them based only on their final results, as an overly complex trajectory seems intuitively worse than a simpler one even if the result is the same.

To address this issue, the EAGER framework consists of an agent learning module, a question generation module QG (automatic, but not learned) and a learned question answering module QA. These two modules fulfil the role of the functions f and h_c defined in Section 13.4. EAGER is inspired from works like QuestEval (Scialom et al., 2021) and Data-QuestEval (Rebuffel et al., 2021), developed for natural language generation. For instance, for abstractive summarization, by generating questions from the original text (QG) and trying to answer them using the summary QA, this method measures the quantity of information shared between both texts.

In our work, we draw the analogy with the NLP task with the goal replacing the original text and the trajectory replacing the summary. We use the QG/QA system to verify that a trajectory contains the same level of information as the language instruction, meaning that the goal is contained in the trajectory. As the goal can be contained in a lot of different trajectories, we also favour simple trajectories. If one can easily answer the question, that means the trajectory is simple.

The QG module

We assume that the goal linguistic instruction ω^g of $g \in \mathcal{G}$ is a highly expressive language instruction (e.g *Put the red ball next to the blue box*, *open the red door*, ...), containing by themselves enough world knowledge to generate questions by masking words. Thus, the QG

module returns a list of k questions $QG_k(\omega^g)$ that can be seen as auxiliary objectives. Each question is formed by masking one word in the linguistic instruction. Crucially, the choice of words to mask can be done automatically without any expert knowledge of the task, or the environment, for instance masking all nouns and adjectives. These questions can be seen as auxiliary objectives guiding the agent during training. Besides, being formulated in natural language, these auxiliary objectives are easily interpreted.

The QA module

Let $\tilde{A}$ be the set of possible answers generated automatically from the list of tokens masked by the QG. Thus, $\tilde{A}$ contains (q, a^*) pairs of questions and expected answers. The QA module returns the probability for all $\tilde{a} \in \tilde{A}$, that $\tilde{a}$ answers question q, given a trajectory τ_t, where $\tau_t = (o_i, a_i)_{i \in [|0,t|]}$ are state-action pairs and t the time step. We note $\tilde{a}^* = QA(q, \tau_t)$ the answer greedily generated from the module and $QA(\tilde{a}^* \,|\, q, \tau_t)$ the associated probability. The auxiliary objectives from the QG are considered achieved when the QA answers them successfully. In our work, the QA module is pre-trained using full example trajectories generated by a bot, see Section 12.5.1, without any type of annotations to guide it. Besides, at time step t, there is no guarantee that the trajectories contain enough information to correctly answer a question. Thus we include a *«no_answer»* token in $\tilde{A}$ to prevent the QA from answering correctly by chance. Moreover, since the QA module takes the whole trajectory, once it has answered a question, it can also answer it at the next step. To avoid giving a reward that does not have direct link to the current step, every time a question q is answered, it is removed from the set of questions: We note $\mathcal{Q}_t$ the active set of questions, the initial set is $\mathcal{Q}_0 = QG_k(\omega^g)$ and once a question is answered, we apply $\mathcal{Q}_t \leftarrow \mathcal{Q}_{t-1} \setminus \{q\}$.

Architecture of the QA

The QA is used to compositionally chain low-level tasks. To do this, it relies on the episodic transformer (Pashevich et al., 2021). This architecture uses multimodal transformers (over language, visual observation, and a list of actions) that have access to the full episode. For any time step t and any question, a $(x_{1:L}, v_{1:t}, a_{1:t})$ tuple is given as input to the QA module. The language input $x_{1:L}$ is the question, it is a sequence of L tokens with $x_i \in \mathbb{N}$. The visual input $v_{1:t}$ is the list of observations $v_t \in \mathbb{R}^{W \times H \times 3}$. Finally, the action input $a_{1:t}$ is a list of discrete actions. As output, the network returns the probability distribution over the set of possible answers $\tilde{A}$.

Intrinsic reward

The main difficulty in ranking trajectories is that many trajectories can share the same description and thus can answer questions correctly. However, a human assessing several trajectories, with the same extrinsic reward, relatively to the same goal would prioritise the simplest one. To account for this, we make the reward proportional to the confidence in the answer at time step t: $QA(\tilde{a}^* \,|\, q, \tau_t)$. An overly complex trajectory being harder to understand for the QA, the answer should be given with less confidence, so it should obtain less reward than a direct and clear trajectory, even with the same number of correct answers.

In this chapter, we keep the same desired properties for the shaped reward function as the ones given in ELLA: the reward shaping should not change the optimal policy that prevails before the reward shaping (policy invariance), and the reward should encourage sample efficient exploration based on auxiliary-objectives. Using intrinsic rewards, we modify the global reward from r to r' through a policy invariant transformation (Ng et al., 1999), which ensure that the new policy π^* is optimal for both $\mathcal{M}$ and $\mathcal{M}' = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{G}', \gamma)$.To do so, we ensure that only successful trajectories get the same return with or without the reward shaping by substracting the shaped reward at the final time step N of a successful trajectory: $r'_N = r_N - \sum_{t \in T_\mathbb{S}} \gamma^{t-N} r'_t$ with $r'_t = \lambda \sum_{q \in \mathcal{Q}_t} \sum_{\tilde{a}^* \textit{is correct answer}} QA(\tilde{a}^* \,|\, q, \tau_t)$, where $T_\mathbb{S}$ is the set of time steps where a bonus is applied. r_t is the reward given at time step t, in the case of sparse reward studied here $r_t \neq 0$ only if t corresponds to the last step of a successful trajectory. In ELLA (Mirchandani et al., 2021) the authors prove that this transformation is policy invariant. Thus, as long as the policy produces unsuccessful trajectories, the agent is guided by the shaping reward. Then once it has learn to successfully complete an instruction, the shaping reward is substracted at the last step and the agent improves using only the extrinsic reward. Further details are provided in the appendix.

The QA system is pre-trained with successful trajectories, which prevents reward hacking. Indeed, if the agent individually completes the auxiliary objectives without a meaningful trajectory, the QA does not consider the trajectory meaningful and answers: "no_answer", preventing the agent to get rewarded.

Using the above notations and concepts, we can define a metric to measure the adequacy of a trajectory to a goal, that corresponds to our cumulative intrinsic reward over a trajectory τ of length N, up to a λ factor:

$$m_{QG/QA}(g, \tau) = \sum_{t=0}^{N} \sum_{q \in \mathcal{Q}_t} QA(\tilde{a}^* \,|\, q, \tau_t) \; \mathbb{I}[\tilde{a}^* \text{correct answer to q}]. \tag{12.1}$$

Algorithm

Our algorithmic procedure is given in 3. At the beginning of an episode, the QG takes the goal and returns a set of questions related to it. Then the QA module is applied at each step over the active set of questions $\mathcal{Q}_t$. When a question is answered correctly, the shaped reward function returns a bonus $\lambda\, QA(\tilde{a}^* \,|\, q, \tau_t)$, where λ is a scaling factor, to the agent and the answered question is removed from the set of active questions.

Then we tune λ to ensure that no unsuccessful trajectories can get more reward than a successful one from the optimal policy π^*. The higher bound for the shaped reward of unsuccessful trajectories is $\lambda\, k$ and the lower bound for the reward of a successful trajectory is $\gamma^T\, r_T$, where k is the number of questions generated by the QG and T is the maximum length of an episode. Thus we obtain

$$\lambda < \frac{\gamma^T\, r_T}{k}. \tag{12.2}$$

Algorithm 3 Automatic auxiliary goal generation and reward shaping using EAGER

Input: θ_0 initial policy parameters, λ bonus scaling factor, ENV the environment and OPTIMISE an RL optimiser

for k=0,..., n_{step} **do**
  $\omega^g, o_0, \text{done}_0 \leftarrow$ENV.$reset()$
  $\mathcal{Q}_0 = \{q_1, ..., q_k\} \leftarrow QG_k(\omega^g)$
  $t \leftarrow 0$
  while done_t not True **do**
    $a_t \leftarrow \pi^{\theta_0}(o_t)$
    $o_{t+1}, r_t, \text{done}_{t+1} \leftarrow$ENV$(a_t)$
    $r'_t, \mathcal{Q}_{t+1} \leftarrow$ QA_SHAPE$(\mathcal{Q}_t, \tau_t, r_t)$
    if done_{t+1} is True **then**
      $N \leftarrow t$
      $r'_N \leftarrow$ NEUTRALISE$(r'_{1:N})$
  Update $\theta_{k+1} \leftarrow$ OPTIMISE$(r'_{1:N})$

function QA_SHAPE$(\mathcal{Q}_t, \tau_t, r_t)$
  for q in $\mathcal{Q}_t$ **do**
    $\tilde{a}^* \leftarrow QA(q, \tau_t)$
    if $\tilde{a}^*$ is correct answer to q **then**
      $r'_t = r_t + \lambda QA(\tilde{a}^* \mid q, \tau_t)$
      $\mathcal{Q} = \mathcal{Q} \setminus \{q\}$
  return $r'_t, \mathcal{Q}$

function NEUTRALISE$(r'_{1:N})$
  $r'_N \leftarrow r'_N - \sum_{t \in T_S} \gamma^{t-N} r'_t$
  return r'_N

Note that by making a less conservative hypothesis, i.e. assuming in the worst case the successful trajectory takes $N < T$ steps, we could obtain a higher λ leading to faster learning (Mirchandani et al., 2021).

12.5 Experiments

12.5.1 Environment

We use the BabyAI (Chevalier-Boisvert et al., 2019) platform to run our experiments. 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 several entities: the agent, boxes, balls, doors, and keys of 6 different colors. These entities are placed in rooms of 8×8 tiles that are connected by doors that could be locked or closed. The agent can do 6 primitive navigation actions such as `forward`, `toggle`, `pick up` to solve the language instruction (for instance `Pick up the red box`). It only has access to partial observations of its environment inside which irrelevant objects are randomly added. Moreover, the observations are in a symbolic space using a compact encoding, with 3 input values per grid cell,[2] $8 \times 8 \times 3$ values in total. When the agent completes the task after N steps, it receives the reward $r_N = 1 - 0.9\frac{N}{T}$, where T is the maximum number of steps. During training, all rewards are scaled up by a factor of 20 to ensure a good propagation of the rewards. If the agent fails, the reward is 0. We focus our tests on tasks of varying complexity: PutNextTo, Unlock and Sequence. The task can also take place in one room `local` or two rooms `Medium`.

[2] 3 integers: one representing the shape of the object, one its color, and one its state. For instance, (4, 1, 1) represents a closed green door

We now explain how to adapt the EAGER method to train an RL agent in BabyAI, a language-conditioned environment where the agent has a limited number of steps to complete a language goal. In this environment the agent receives a reward if and only if it finishes the task successfully.

The BabyAI benchmark contains tasks with highly expressive language instructions e.g *Put the red box next to the green key, ...*). Thus they are rich enough to generate questions by masking words. In practice, we mask nouns and adjectives: this form of QG is very simple and can be automated using standard NLP techniques, thus it does not require expert knowledge. For instance, for the goal *Put the red ball next to the blue box*, using the token *«question»* as a mask we generate 4 questions among which *Put the «question» ball next to the blue box.*

The environments in our experiments are partially observable. Thus, our agent takes sequences of observations $(o_1, o_2, ..., o_t)$ as inputs of a recurrent network (Hausknecht & Stone, 2015).

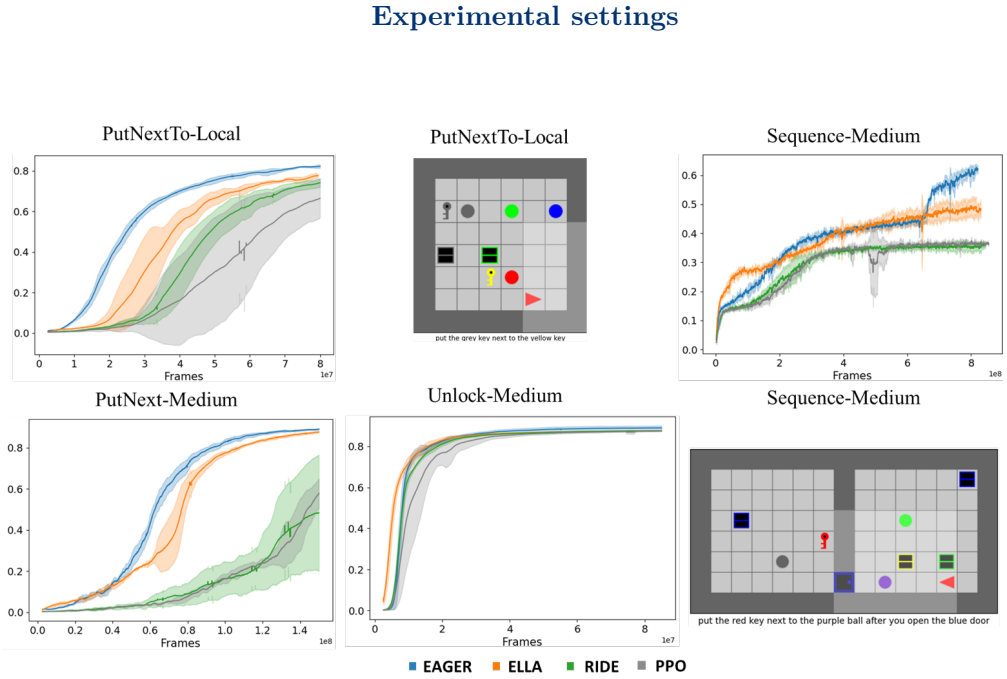


Figure 12.2: Average reward for EAGER and baselines for four tasks, with error regions to indicate standard deviation over four random seeds. For the PutNext-Local and Sequence-Medium tasks, we give an example of possible tasks with environment at time step 0.

To train the QA module through supervised learning, we build a dataset of example trajectories associated to language goals using a bot provided in BabyAI, then we generate related questions and answers. To obtain a QA that can operate on various tasks, we use a mix of the PutNextTo, PickUp, Open, and Sequence tasks for generating training trajectories. This dataset is only used to train the QA, not to bootstrap policy learning. During training, we give as input of the QA the full trajectory — the list of observations and actions — and

all the questions generated by the QG and we use the cross-entropy loss over its output distribution to update it. To train the QA to answer: "no_answer" and prevent it from guessing the answer by chance, we randomly associate some trajectories and questions from unrelated goals. For instance, we associate the trajectory from the goal: *take the red box next to the blue ball* and associate it with the question: *take the red «question» next to the blue key* generated from the goal *take the red box next to the blue key*. The QA must learn to pay attention to all the details of the question. It has to see that the trajectory describes an agent taking the *red box* but placing it next to an object other than the one in the question. Thus the good answer to the question is *no_answer* and not *box*.

Moreover, we empirically show in Section 12.5.3 that the QA is more efficient when it learns from a broad distribution of trajectories for similar tasks. The intuition behind such behaviour is that a QA trained on a narrow distribution of successful bot-generated trajectories will not recognise the noisy trajectories of the agent when it starts training. Thus the QA will too often answer "no_answer", resulting in no intrinsic reward and hurting the reward shaping efficiency. To produce a wider distribution using the given procedural bot, we force the bot to take with a certain probability a random action at each time step and only keep successful noisy trajectories as training examples. More details on the QA pre-training are given in Appendix E.2.1. All the tasks we use for training the QA and the agent are summarised with examples in Appendix E.1.

Table 12.1: All the assumptions and expert knowledge required for RIDE, EAGER, and ELLA

Method	**Number of expert demonstration per task**	**Human Expert Intervention**	**Automated parts done by the agent**
RIDE	0	None	Determining if a new state is impacting
EAGER	7.500 noisy bot trajectories (see supplementary *Wide distribution of trajectories*)	Determining what words are nouns or adjectives	Determining relevant auxiliary objectives Determining auxiliary objectives completion Determining auxiliary objectives associated reward
ELLA	15.000 bot trajectories	Determining the class of relevant auxiliary objectives Determining auxiliary objectives associated reward	Determining relevant auxiliary objectives among the predetermine class Determining auxiliary objectives completion

To evaluate our reward shaping framework, we use the PPO algorithm (Schulman et al., 2017a), but our reward shaping method is algorithm agnostic. We compare our framework to PPO without reward shaping, ELLA and RIDE. RIDE (Raileanu & Rocktäschel, 2020) is an exploration method that does not use language and addresses sparse reward problem by rewarding impactful change in state. We use Nvidia Tesla V100 with 10 cores to train our model and we use 4 seeds in each experiment.

12.5.2 Q1. Sample efficiency

Main results

Figure 12.2 presents learning curves for EAGER, ELLA and RIDE across 4 environments. Table 12.1 describes the assumptions and the type of expert knowledge required by the three reward shaping methods. It clearly appears that EAGER requires less expert human intervention than ELLA.

Effect of sparsity

In the PutNextTo and Unlock tasks, EAGER obtains results better than ELLA (SOTA in BabyAI) without using expert knowledge. It also performs significantly better than RIDE for the tasks PutNext and Sequence and slightly better for Unlock. The better performance of EAGER with respect to RIDE is not surprising as the EAGER agent receives some indications based on example trajectories through the QA module.

For Unlock-Medium, EAGER overcomes a bottleneck. The general goal being *Open the «colour» door*, the agent has to first pick up the key of the corresponding *colour* before reaching the door to open it. ELLA rewards picking up keys, via the PICK low-level instruction chosen via expert knowledge. EAGER reaches better performance (see the statistical test in Appendix E.3) without the need for expert knowledge. Moreover, although this is not its main purpose, EAGER gets a similar or better sample efficiency for most tasks.

Temporal constrain

The Sequence task adds a temporal constraint by chaining two tasks using 'before' or 'after' together with a high number of instructions (over 1.5M instructions in comparison with PutNext-Medium with 1440 instructions). Moreover both EAGER and ELLA decompose the goal into auxiliary objectives. This decomposition does not retain the temporal constraint, there is no notion of doing one auxiliary objective before another.

Our tests show that EAGER retains strong performance, doing better than RIDE and ELLA. The slow progress of EAGER at the beginning can be attributed to the time when the agent is not good enough to efficiently trigger an intrinsic reward signal from the QA module. Indeed, at the beginning, trajectories are noisy and it is more difficult for the QA to exploit a trajectory with more rooms. As a result, it over-responds "no_answer" leading to a lesser intrinsic reward.

Empirical observations on EAGER

To understand why EAGER achieves better results than ELLA or RIDE, we propose an answer based on qualitative observations. We believe that EAGER guides the agent more finely. The generated questions break down the trajectory into more precise key points than the auxiliary objectives generated in ELLA. To verify our point, we plot the average trajectory over 4 seeds for agents trained with EAGER or ELLA, see Figure 12.3. Looking

at the average trajectories, it appears that EAGER generates trajectories that are spatially more compact and that return to the same place fewer times.

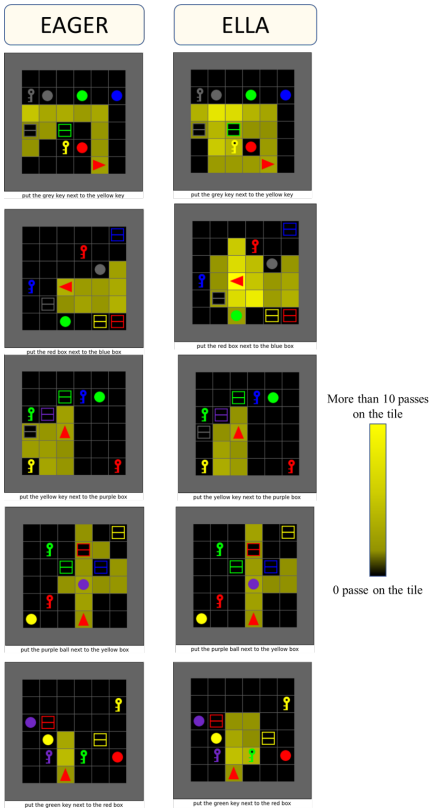


Figure 12.3: Average trajectory for PutNextTo-Local tasks for EAGER and ELLA. The colour of the tiles depends on the average number of passes of the agent on the tile. The 5 environments presented here have been randomly selected. The position of the agent is its position at the beginning of the episode.

12.5.3 Q2. Robustness

Robustness of EAGER to QA performance

At first glance, the reliability of the QA looks crucial to our method. However, the QA could be difficult to train in some environments e.g. if you want the QA to learn to answer in a large set of answers from a small number of example trajectories. This is why we tested the robustness of our method using the PutNextTo-Local task with two metrics: the success rate SR of the QA after pre-training and the distribution of example trajectories. For the former, we take the same QA at different training epochs and we determine its SR over a test dataset, then we train the agent using the reward shaping provided by this QA. Figure 12.4(left) shows the robustness of our method with agents that display similar training curves as soon as $SR > 0.56$.

In Figure 12.4(right), we plot the SR of the QA when training the agent. Initially, the QA with a $SR \leq 0.56$ at pre-training time tends to have a higher SR. Indeed, the distribution of

answers is less peaked and answers are often correct by chance. On the opposite, the QA with a $SR > 0.56$ answers "no_answer" and obtains no reward. However, in this case, the agent learns faster because it only receives a reward for meaningful answers. For the QA with a $SR \leq 0.56$, the SR along training first grows then decreases. First, the agent is biased by the intrinsic reward to follow a path that improves the SR, but once the agent learns to complete the trajectory leading to the extrinsic reward, the SR converges to pre-training SR.

As explained in Section 12.5.1, we added noise to trajectories generated by the bot to compensate for a too narrow trajectory distribution. Figure 12.5 shows training curves for two environments for QA trained on Wide trajectory Distribution (WD) and Narrow Distribution (ND). The reward shaping method trained on (WD) learns faster because they efficiently reward the agent early in training.

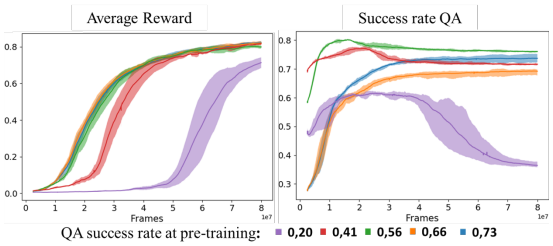


Figure 12.4: Average reward (left) and success rate of the QA (right) for the PutNextTo-Local tasks. The agent are trained with QA having different success rate after pre-training.

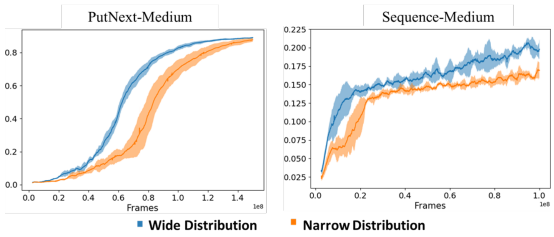


Figure 12.5: Average reward in tasks PutNextTo and Sequence for two distributions of bot trajectories used to train QA. A narrow and a wide distribution where noise is added to bot trajectories

Robustness of EAGER to QA design

In Section 12.4.2 we made two choices for the QA module and the associated intrinsic reward: first we added a "no_answer" response, second we rewarded each answer by the confidence the agent had in its own answer. To verify the influence of these choices over EAGER's performance, we use the PutNextTo task to compare EAGER against "EAGER \no_answer", "EAGER Simple", and "EAGER Simple \no_answer". "Simple" means that the agent received a binary intrinsic reward (1 for a good answer, 0 otherwise) and "\no_answer" means we suppressed the "no_answer" solution. Figure 12.6 gives the results of these ablations. We can see that both the use of "no_answer" and the non-binary reward independently boost sample efficiency.

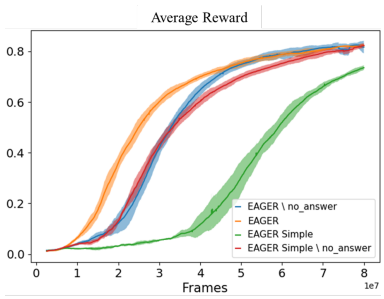


Figure 12.6: Average reward in PutNextTo with various ablations of the QA module.

Verifying that the QA does not guess

With the QA, the agent can self-check the followed instructions. Due to the format of the generated questions (i.e. masking a word) and proceeding by elimination depending on the object present in the environment, a simple QA could guess the answer, breaking the EAGER method. Thus, to avoid such issues we added so-called “no_answer questions” to the QA training data set. These questions are examples where the trajectory does not correspond to the question, e.g. a trajectory corresponding to the goal "pick up the red box then pick up the red ball” is associated to the question "pick up the «question» key then pick up the red ball". The QA must learn to associate each element to see that the path is not associated to the correct question and appropriately answer “no_answer”. In practice, this prevents the QA from guessing the answer, it needs to wait for the agent’s path to match before answering. Thus for a question such as "pick up the «question» box then pick up the red ball”, even if the only box in the room is red, the QA must answer “no_answer” as long as the agent has not completed the corresponding auxiliary objective.

We verified that the QA correctly acts by checking a hundred trajectories with the possibility to guess an answer based on the linguistic input, such as with the goal “pick up the red ball then pick up the red ball”. We give 4 examples among the tasks we have checked in Figure 12.7. We did not find any instance where the QA guessed the answer by chance right from the beginning of a trajectory. Indeed, even if it is possible to just use linguistic elements to answer a question such as “pick up the «question» ball then pick up the red ball”, the QA cannot associate the answer “red” to any element of the trajectory and thus returns “no_answer”.

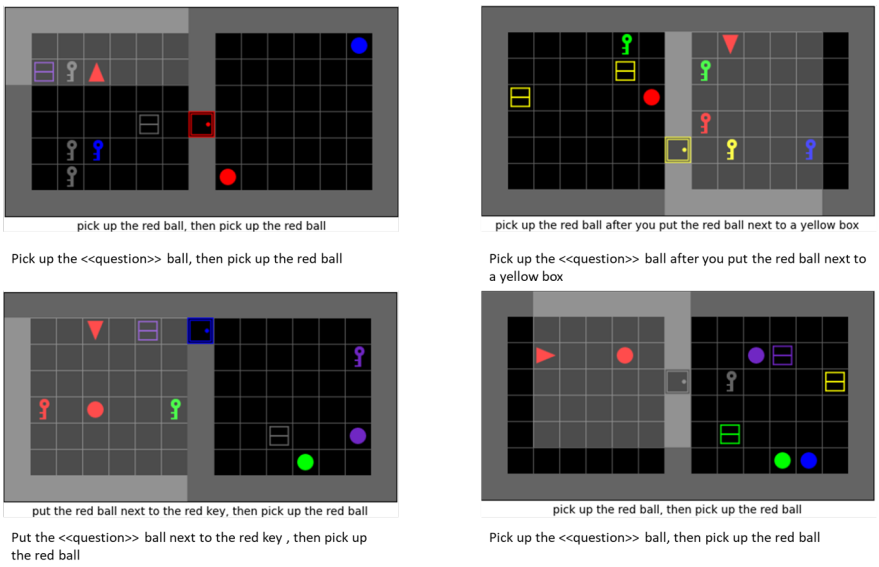


Figure 12.7: Four examples and possible question at time step $t = 0$ where the QA could guess the answer either linguistically or by elimination process. In all of these examples the QA returns no_answer because the trajectory does not correspond to a possible answer

12.6 Discussion

In this work, we have proposed to leverage the abstraction and generalisation properties of language to construct an automatic reward shaping method tailored to long-horizon, sparse-reward tasks. Our learning agent generates its own questions based on the target goal and rewards itself for answering them correctly, thereby creating an efficient and adaptive curriculum of auxiliary objectives. This stands in contrast to prior work such as ELLA (Mirchandani et al., 2021), which relies on expert knowledge to manually select suitable auxiliary tasks. Moreover, unlike the method proposed by (Mu et al., 2022), our approach does not require access to an oracle capable of providing linguistic descriptions of environmental states.

The current implementation of EAGER relies on a pre-trained QA system built using a fixed set of example trajectories. A natural next step would be to investigate how to remove this constraint by learning QG and QA mechanisms online, potentially following autotelic strategies. Furthermore, while our experiments were conducted in the BabyAI environment, which features synthetic language and a controlled 2D setting, future work should consider more linguistically and visually complex environments. For instance, environments such as ALFRED (Shridhar et al., 2019), which involve real human instructions and 3D embodied interaction, would allow us to test the robustness and scalability of our method in more realistic scenarios.

Since the publication of this work, several studies have advanced the idea of leveraging the semantic richness of language—particularly through the use of LLMs—to derive reward signals. Kwon et al. (2023) employ LLM outputs as reward functions in conversational environments, demonstrating that such rewards better align agent behaviour with human preferences. Pignatelli et al. (2024) provide a systematic investigation into the use of LLMs for automating credit assignment via reward shaping and options discovery, achieving promising results in challenging environments such as MiniHack (Samvelyan et al., 2021).

In EAGER, we improve the agent's learning efficiency by exploiting the compositional and hierarchical structure of language for reward shaping. By rewarding partial achievements along a trajectory, the agent implicitly learns subgoals, facilitating temporal abstraction. As also highlighted by Jiang et al. (2019), language is particularly well-suited to expressing decomposable goals, making it a natural fit for the options framework (Sutton et al., 1999; Precup & Sutton, 2000). Motivated by this insight, in the following chapter we introduce a hierarchical reinforcement learning approach in which a large language model operates as a high-level policy. This high-level controller learns to select among an expanding set of low-level behaviours, dynamically constructing increasingly sophisticated strategies over time.

Chapter 13

Efficient learning of skills with hierarchical learning

13.1 Motivation

Recent breakthroughs in AI have produced agents with human-level performance in vision and language goals, largely due to foundation models trained on large-scale internet data (Yu et al., 2022; Achiam et al., 2024; DeepSeek-AI et al., 2025). However, the static and limited nature of these datasets raises concerns about their sufficiency for achieving truly general intelligence (Villalobos et al., 2024).

In contrast to these dataset-dependent approaches, humans acquire a remarkably diverse repertoire of skills continuously throughout life without relying on static datasets. One major objective of artificial intelligence is to endow agents with similar capabilities: learning without a dataset through interactions with the environment and autonomously discovering and expanding a diverse skill set. The former is covered by RL methods while the latter, referred to as open-endedness, is considered fundamental for the development of AGI (Sigaud et al., 2024) and even ASI (Edward et al., 2024). According to (Colas et al., 2022a), such agents should be autotelic, meaning they are capable of generating, selecting and training on their own goals for continuous and open-ended self-improvement.

Recent approaches have begun leveraging foundation and generative models to construct open-ended and autotelic agents, as exemplified by MineDojo (Fan et al., 2022), Voyager (Wang et al., 2023b), OMNI (Zhang et al., 2023b) or ACES (Pourcel et al., 2024a). These systems utilize large models to explore the goal space, either by autonomously generating novel goals and associated rewards or by selecting from a pre-generated goal set and structuring learning via a curriculum through prioritizing goals with high learning progress (Gaven et al., 2025). However, as the goal space expands, increasingly complex goals emerge that require the composition of multiple elementary actions. This combinatorial explosion in goal complexity, observed in (Valmeekam et al., 2023; Bauer et al., 2023; Poesia et al., 2024), tends to impede the open-ended learning process. When each new goal demands an increasingly longer time to master, agent progress stalls.

Humans, as quintessential lifelong learners, face similar challenges in acquiring complex skills, such as mastering a new sport. Nevertheless, they do not experience diminishing

returns in learning. Building upon (Fitts & Posner, 1967), Tsay et al. (Tsay et al., 2024) demonstrate that humans employ hierarchical learning to overcome the complexity barrier: cognitive systems decompose new skills into simpler, previously internalized subskills handled by sensorimotor systems. As new goals are mastered, they are recursively encoded at lower levels, enabling rapid reuse in future learning.

This biological principle has inspired artificial intelligence research. Hierarchical structures have been adapted to enable open-ended object recognition (Kasaei et al., 2016) and HRL (Sutton et al., 1999; Precup & Sutton, 2000). Several recent works, including (Hu et al., 2019; Duminy et al., 2021), (Ahuja et al., 2023), and (Jiang et al., 2019), have further incorporated natural language structures to facilitate goal decomposition. However, most of these methods assume predefined skills, often requiring pre-trained policies on these skills. This static setup is inherently incompatible with open-ended agents, which must continuously face novel and increasingly complex goals.

13.2 Contributions

In this work, we propose HERAKLES for **H**i**ER**archic**A**l s**K**ill compi**L**ation for open **E**nded llm agent**S**. In open-ended settings, the efficiency with which new goals are learned becomes critical: if each new goal must be learned from scratch, the agent may eventually fail to achieve its own proposed goals, stalling its progress. Our work targets this challenge directly—not by focusing on the creative generation of new goals (which is a distinct and valuable research direction), but by addressing efficient generalization and reuse of previously mastered skills to accelerate learning in a growing goal space. HERAKLES extends prior HRL and language-based approaches to support continual adaptation without requiring pre-specified skills or extensive pre-training. Specifically, we train a high-level (HL) policy to invoke a low-Level (LL) policy on goals the latter has already mastered. The HL policy is instantiated as a LLM, leveraging its capacity to constrain exploration, select relevant skills, and operate effectively in a shifting skill landscape. The HL policy is trained online over an automatic curriculum of increasingly difficult goals, while its learned skills are progressively distilled into the LL policy: a lightweight, computationally efficient model specialized in executing primitive actions.

We evaluate our method in the Crafter environment (Hafner, 2022), designed to assess a wide spectrum of agent capabilities within a unified open-ended framework.

Our key research questions include:

- **Q1. Implementation** How can we implement high- and low-level interaction and concurrent learning? How can we compile skills acquired by the hierarchical agent as a whole into the LL policy to foster efficient learning?

- **Q2. Scaling** How does HERAKLES scale with the difficulty of the goal?

- **Q3. Sample efficiency** Is HERAKLES more sample-efficient than prior HRL approaches or LLM-only strategies?

•**Q4. Generalization** Can HERAKLES generalize to new goals without additional training?

To answer these scientific questions, we describe how HERAKLES works in Section 13.4 and test it in the Crafter environment (Hafner, 2022) in Section 13.5.

13.3 Related work

Neuroscience

The strong abilities of humans to learn new skills as open-ended agents have long been studied in psychology and neuroscience. From learning to crawl on the ground as infants to performing complex sensorimotor operations playing an instrument or cooking, humans continuously acquire increasingly complex sensorimotor skills during their lifetime. Gottlieb & Oudeyer (2018) explain that this lifelong process is nurtured by curiosity and supported by effective learning of new skills. Fitts & Posner (1967) proposed a phenomenological model in which a transition occurs from cognitively demanding tasks to more automatic actions. Tsay et al. (2024) expand on this model and proposes three processes: reasoning, when the novice reasons to understand action–outcome relationships; refinement, for optimizing the sensorimotor part to achieve motor goals; and retrieval, when the agent infers the best sensorimotor skills to recall to reach a new goal. Riesenhuber & Poggio (1999) have also found a hierarchical structure in the brain for life-long object classification and recognition. This capacity is also crucial for open-ended agents which are confronted to new artifacts (created or encountered) during their existence. Cangelosi et al. (2010) advocate in the context of developmental robotics, in favor of hierarchical learning with language, since the latter has combinatorial properties and generalization properties that would help the former. Although it does not directly try to model the processes described in these previous papers, our work shares strong conceptual similarities with them. In fact, it uses an LLM as a high-level policy guiding a smaller neural network which acts as a low-level policy using only elementary actions. Our method proposed a way to continuously compile the skills learned by the hierarchic agent as a whole into the low-level policy. Thus, the action space on which the high-level policy can act expands, facilitating the exploration and learning process of more complex goals.

Hierarchical Reinforcement Learning and language

The design or learning of meaningful abstractions at different resolutions in space or time in order to learn tasks more efficiently (Dayan & Hinton, 1992) has long been studied in RL. Since Dayan et al. Dayan & Hinton (1992) proposed to generate a hierarchy of abstractions to solve MDPs, the *option* framework was formalized in order to learn temporal abstractions (Sutton et al., 1999; Precup & Sutton, 2000; Bacon et al., 2017). In recent years, several implementations have been proposed such as h-DQN (Kulkarni et al., 2016), FuN (Vezhnevets et al., 2017) or HIRO (Nachum et al., 2018c). However, these approaches rely purely on the final reward and decompose the goal into subgoals that are points in an abstract embedding space to guide the low-level policy. The embedding space is learned during training.

Nachum et al. (2018b) have shown that the mapping of the observation space to the goal space has a strong influence on the quality of the hierarchical agent. The linguistic space already has generalization and abstraction properties, which are key to exploiting the compositional, relational, and hierarchical structure of an environment (Kaplan et al., 2017; Luketina et al., 2019b; Jiang et al., 2019). In recent years, the abilities of LLMs to model natural language have been used in HRL setups. They model HL policies decomposing goals into skills treated by specialized LL policies (Ahn et al., 2022; Ahuja et al., 2023). Finally, to avoid the bottleneck of the linguistic description of the environment, used by LLMs, several articles use VLMs to accept both visual and linguistic inputs (Shi et al., 2025; Intelligence et al., 2025). All these methods based on LLMs and VLMs pre-train the LL policy while the HL policy is trained to use it (Ahuja et al., 2023; Shi et al., 2025; Intelligence et al., 2025) or kept frozen (Ahn et al., 2022). To be efficient, such settings suppose that any goal in the goal space can be decomposed into a few skills that are given by the experimenters. Our method also relies on language to obtain a structured space of skills. We do not rely on an expert-defined skill space as we reuse previously mastered goals as skills for reaching new ones. In addition, our method does not pre-train the LL policy on a fix set of skills.

Some methods use programmatic or symbolic policies as low-level components (Wang et al., 2023b; Liu et al., 2023). These methods offer strong interpretability and generalization capabilities. However, a key limitation remains: it is not evident that executable programmatic policies can be feasibly or scalably constructed across a wide variety of environments, particularly those with continuous, stochastic, or partially observable dynamics. Our work instead focuses on neural low-level policies, which are more broadly applicable and better suited for the types of tasks and state/action representations encountered in the Crafter environment.

Eventually, BOSS (Zhang et al., 2023c) similarly to HERAKLES also involves the composition of learned low level skills via a hierarchical structure using an LLM for the high level policy. However, HERAKLES does not rely on a pre-trained library of skills to start the learning process . In contrast, HERAKLES begins with a minimal and universal set of basic atomic actions (e.g., "move forward", "pick", etc.), which require no pre-training or demonstration data. All complex skills in HERAKLES emerge over the course of interaction with the environment through reinforcement learning and policy distillation, allowing for skill acquisition and composition from scratch. Moreover, BOSS employs a high-level policy that proposes the next skill based solely on past skill execution history, and operates in a goal-agnostic, bottom-up manner, aiming to discover goals through interaction. Thus, BOSS focuses on autonomous goal discovery, whereas HERAKLES emphasizes goal-directed skill reuse and efficient learning within a top-down curriculum. These distinct motivations lead to distinct architectures and learning dynamics. Finally, at test time, BOSS evaluates the generalization capability of the low-level policy in isolation, requiring it to achieve a novel goal based on previously observed goal distributions. HERAKLES, by contrast, retains its full hierarchical structure during evaluation, with the high-level LLM-based policy actively reusing and composing previously learned skills.

Open-ended autotelic agent and language

During their lifetime, humans continuously face, or even invent new goals and learn how to solve them (Gottlieb & Oudeyer, 2018). This open-ended ability has interested the AI researchers since the birth of the field (Reader, 1969; Turing, 1950; Harvey, 1991). Recently, Sigaud et al. (2024) and Edward et al. (2024) have proposed formal definitions that require a system to produce a sequence of novel and learnable artifacts for an observer to be qualified as open-ended. The abstraction and generalization properties of language make it an excellent structure for creating new artifacts, and several papers have tackled the open-ended text generation issue (Li et al., 2013; Huang et al., 2016; Wang et al., 2016a; Santhanam et al., 2020; Zhu et al., 2023).

The capacity of an artificial agent to autonomously select, generate or discover goals is called autotelic (Colas et al., 2022a). Sigaud et al. (2024) argue that this property plays a key role in the open-ended agent framework. Recently, several papers have used LLMs to generate new goals by relabelling trajectories (Zhang et al., 2023c) or creating new goals (Wang et al., 2023b) and their corresponding reward functions (Faldor et al., 2025; Pourcel et al., 2024a). In our work, we assume that a mechanism has already generated new goals and their corresponding reward functions. We study how to learn to achieve goals as efficiently as possible in a vast goal space with goals of heterogeneous difficulty.

13.4 HERAKLES: Hierarchical Skill Compilation for Open-ended LLM Agents

In this section, we detail how HERAKLES learns new goals. We first formalize in Section 13.4 the RL problem we tackle. Then we detail how the inner working of our hierarchical policy 13.4. We explain how a skill space is generated at each step of the high level policy in Section 13.4 and how the policy of the hierarchical agent is compiled in the low level agent in Section 13.4.

Problem statement

Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{G}, \Omega, O, \mathcal{R}, \gamma)$ be a goal-augmented Partially Observable Markov Decision Process, with $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, $\mathcal{T}$ the transition function, $\mathcal{G}$ the goal space, Ω the observation space, $O : S \to \Omega$ the observation function that maps states to observations, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \{0; 1\}$ the sparse reward function and $\gamma \in [0, 1]$ the discount factor. Here, $\mathcal{G} \in \mathcal{V}^{L_i}$ is a textual space of at most $L_i \in \mathbb{N}$ tokens in a given vocabulary $\mathcal{V}$. In our setting, we consider that $\mathcal{G}$ always includes instructions corresponding to each action. We also consider a prompting function $\phi : \Omega \times \mathcal{G} \to \mathcal{V}^{L_p}$ that transforms any (observation, goal) pair into a textual prompt of $L_p \in \mathbb{N}$ tokens.

Let the goal space $\mathcal{G}$ be compositional, which means that there exists pairs of goals $(g, g') \in \mathcal{G}^2$, such that $g' \prec g$, meaning that the completion of g' is a prerequisite for the completion of g. We assume goal spaces whose the set of precedence relations $g' \prec g$ defines

a hierarchy. For pairs of goals (g', g) that respect $g' \prec g$, one can factor any trajectory $\tau_g = \{s_0, a_0, s_1, a_1, ..., s_n\}$ that reaches g with a trajectory that reaches g' such that $\tau_g = \tau_{g'} \cup \{s_k, a_k, s_{k+1}, a_{k+1}, ..., s_n\}$, with $k > 0$. This structure is common for goal spaces generated by open-ended agents, as they tend to reuse previously reached goals to generate new ones (Wang et al., 2023b; Faldor et al., 2025; Pourcel et al., 2024a).

The hierarchical policy

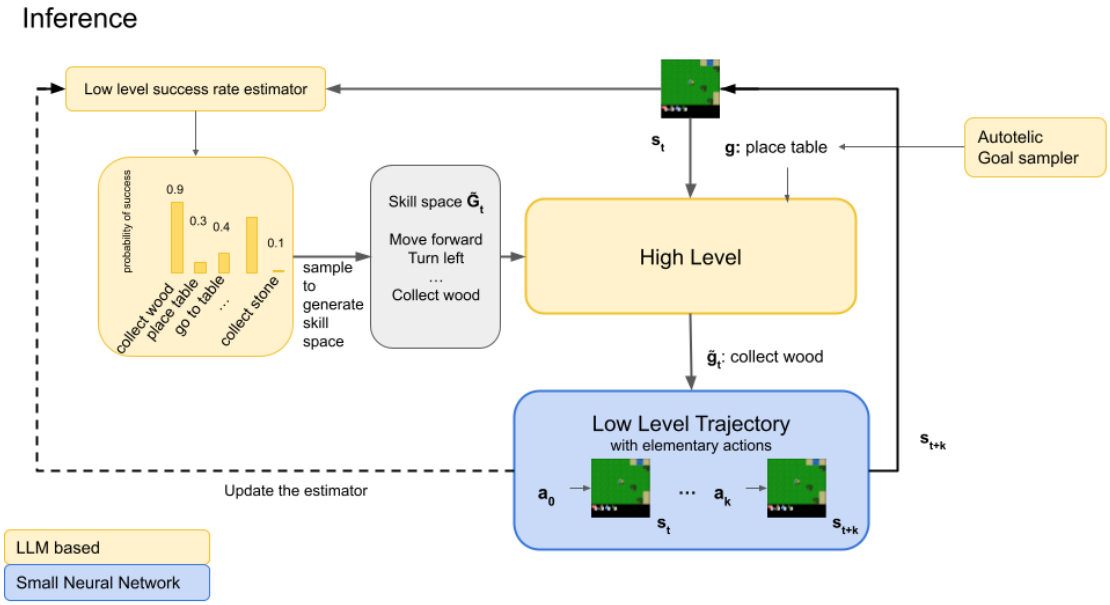


Figure 13.1: **HERAKLES** is an autotelic hierarchical RL agent which relies on two policies. An LLM-based high-level policy iteratively selects goals for a low-level policy which interacts with the environment through elementary actions to solve them. In particular, the high-level policy only selects goals among the ones with high success probability by learning to estimate the probability of success of the low-level policy on each goal. As HERAKLES masters a goal, the full trajectory (i.e. all elementary actions used to accomplish the sequence of goals produced by the high-level policy) is compiled and learned as a new single skill by the low-level policy. This new skill can then be directly used by the high-level policy.

To exploit the structure of $\mathcal{G}$, we consider an HRL agent that can learn and operate at different levels of temporal abstraction. Our agent is modelled as a two-level policy, composed of an HL policy $\pi^{HL} : \phi(\Omega, \mathcal{G}) \to \tilde{\mathcal{G}}$ that samples skills in a skill space $\tilde{\mathcal{G}} \subseteq \mathcal{G}$ and an LL policy $\pi^{LL} : \Omega \times \mathcal{G} \to A$ that samples actions. π^{HL} operates on a long-term scale with goals $g \in \mathcal{G}$ that can be achieved by π^{LL} using several elementary actions $a \in \mathcal{A}$.

The skill $\tilde{g} \in \tilde{\mathcal{G}}$ sampled by π^{HL} is passed to π^{LL} that executes it (see Figure 13.1). π^{LL} operates on a finer time scale compared to π^{HL}, allowing for a finer control of the agent trajectory using elementary actions $a \in \mathcal{A}$. As $\mathcal{A} \subset \mathcal{G}$, π^{HL} can directly guide π^{LL} using elementary actions $a \in \mathcal{A}$: $\forall (a, s) \in \mathcal{A} \times \mathcal{S}$, $\pi^{LL}(a|s, a) = 1$. This property is especially useful when π^{LL} is learning the goal.

π^{HL} must reach the goal within N^{HL} sampled skills and π_{LL} must execute the skill provided by π^{HL} within N^{LL} steps in the environment. The objective is to find the (π^{HL*}, π^{LL*}) pair that maximizes $\mathbb{E}_{g\sim\mathcal{G}}[\sum_k^T \gamma^k r^g_{t+k+1}]$, where $r^g_t = \mathcal{R}(s_t, a_t, g)$, the reward obtained at step t when the hierarchical agent try to complete the goal g.

Skill space generation

A crucial point of our method is how to construct $\tilde{\mathcal{G}}_k$ at each step k of the policy π^{HL}. Indeed, at the beginning of training, $\tilde{\mathcal{G}}$ is initialized with the set of elementary actions as basic skills. At that point, π^{LL} can only complete these elementary actions $\mathcal{A}$. Adding skills where π^{LL} fails would only encourage π^{HL} to completely stop sampling such skills, which could be an issue later in the training. After π^{LL} masters a new goal g, it can be added to the skill space $\tilde{\mathcal{G}}$. Thus, the optimal $\tilde{\mathcal{G}}$ evolves during training and depends on the state s_k of the environment. At each step k of π^{HL}, we construct $\tilde{\mathcal{G}}_k$ starting from an empty set and adding to it each goal $g \in \mathcal{G}$ with a probability:

$$p_g = \max(\mathbb{E}_{\pi^{LL}(.|g), s_k\sim\mathcal{D}_{\pi^{LL}}}[r^g|O(s_k)], \varepsilon_k),$$

with $\mathcal{D}_{\pi^{LL}}$ the distribution of states generated by π^{LL} and ε_k an exploration term. Thus, if π^{LL} has a high probability of success for a goal for a given state s_k, the goal is added to $\tilde{\mathcal{G}}_k$. As π^{LL} always successfully executes elementary actions $a \in \mathcal{A}$, we have $\forall k, \phi(\mathcal{A}) \subseteq \tilde{\mathcal{G}}_k$.

To measure the probability of success $\mathbb{E}_{\pi^{LL}(.|g), s_k\sim\mathcal{D}_{\pi^{LL}}}[r^g|O(s_k)]$, we leverage the LLM used by π_{HL} by learning the parameters θ_k of a competence estimator $C_{\theta_k}(s_k, g) \approx \mathbb{E}_{\pi^{LL}(.|g), s_k\sim\mathcal{D}_{\pi^{LL}}}[r^g|O(s_k)]$, following a strategy similar to that of (Gaven et al., 2025). We pass in the LLM a prompt describing the pair $(s, g) \in \mathcal{S} \times \mathcal{G}$ which produces a latent representation on top of its final decoder block for the last token. The representation is passed to a multilayer perceptron (MLP) that outputs the estimated success probability $C_{\theta_k}(s, g)$. We train both the LLM and the MLP by minimizing the binary cross-entropy: $\mathcal{L}(\theta_t) = \mathbb{E}_{((s,g),r)\sim\mathcal{D}_t}[BCE(r, C_{\theta_t}(s, g))]$, with $\mathcal{D}_t$ a buffer containing the most recent $((s, g), r)$ pairs. In practice, we train two separate versions of the same initial LLM using LoRA adapters (Hu et al., 2022): one for π_{HL} and one for our competence estimator.

RL training of the HL and LL policies

HERAKLES simultaneously learns π^{HL} and π^{LL}, respectively the HL and LL policies (see Figure 13.2). π^{HL} is a pre-trained LLM, fine-tuned using RL. It samples skills from the set of skill $\tilde{\mathcal{G}}_k$, the skill space constructed at each step (see Section 13.4), using constrained decoding. π^{LL} is a small, not pre-trained, neural network also trained using RL.

π^{HL} is trained with on-policy RL using trajectories $\{\tau^{HL}_g = \{(s_0, \tilde{g}_0, r^g_0), .., (s_K, \tilde{g}_K, r^g_K)\} | g \in \mathcal{G}, K \in [|0, N^{HL}|]\}$ with $\tilde{g}_K \in \tilde{\mathcal{G}}_k$ the skill space.

As depicted in Figure 13.2, π^{LL} is trained with off-policy RL using two types of trajectories:

- Those from when it acts to execute the skill $\tilde{g} \in \tilde{\mathcal{G}}$ given by π^{HL}, $\{\tau^{LL}_{(s_0,\tilde{g})} = \{(s_0, a_0, r_0^{(s_0,\tilde{g})}), .., (s_T, a_T, r_T^{(s_0,\tilde{g})})\} | \tilde{g} \in \tilde{\mathcal{G}}, T \in [|0, N^{LL}|]\}$.

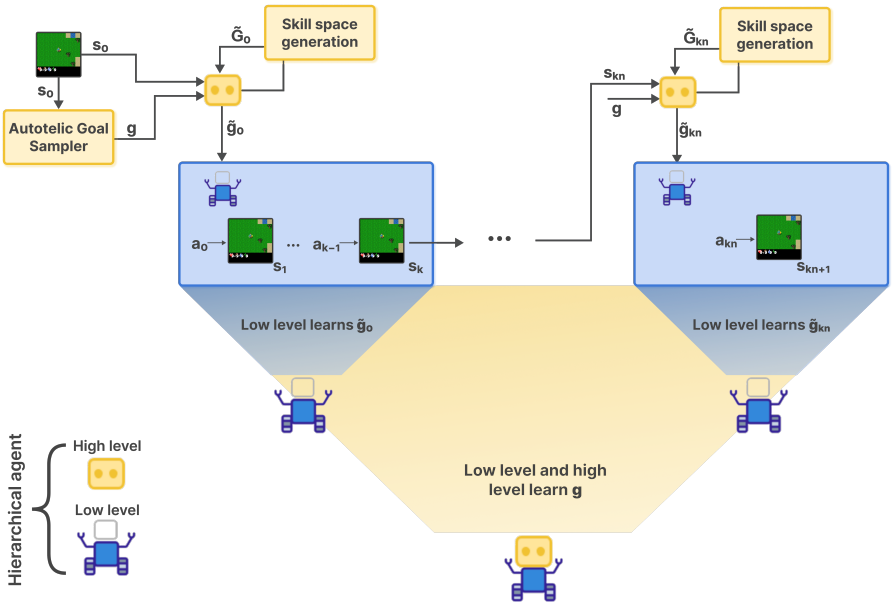


Figure 13.2: **Skill learning and compilation in HERAKLES**: Given a goal g and a state s_0, the high-level policy generates its skill space $\tilde{\mathcal{G}}_0$ and samples one skill $\tilde{g}_0$. This skill is then sent to the low-level policy which performs k elementary actions in the environment to achieve $\tilde{g}_0$, leading to state s_k. Given s_k and g, the high-level policy then generates a new skill space, samples a new skill g_k and sends it to the low-level policy. This process repeats until g has been achieved or the limited number of high-level steps have been reached. This process produces both multiple low-level trajectories (i.e. one per skill sampled by the high-level) and a single high-level trajectory (i.e. the sequence of skills sampled). The high-level policy is trained on the high-level trajectory to improve skill selection. The low-level policy is trained on all the low-level trajectories, conditioned to reach the associated skill. Crucially, we perform **skill compilation** by also training the low-level policy on a trajectory that is the concatenation of all the low-level trajectories and that is conditioned to reach the high-level policy's goal g. Doing so, our low-level policy progressively internalizes the sequence of elementary actions required to directly solve g.

• Those which achieve the goals $g \in \mathcal{G}$ obtained by concatenating the low-level transitions for each skill selected by the HL policy $\{\tau_g^{LL} = \{(s_0, a_0, r_0^g), .., (s_M, a_M, r_M^g\} | g \in \mathcal{G}\}$.

As π^{HL} masters a goal g, it is distilled into π^{LL}. π^{HL} can then use π^{LL} to reach g, for example inside a trajectory to achieve a more complex goal g'.

The HERAKLES algorithm

In this section, we provide the pseudocode of the HERAKLES algorithm (see Algorithm 4) to facilitate a clearer understanding of the core training loop and its hierarchical decision-making structure. For clarity and brevity, we omit the update mechanisms for the MAGELLAN goal-sampling module (Gaven et al., 2025) as well as for the low-level success rate estimator. These components, while essential to the full implementation, are orthogonal to the main hierarchical control logic and are described in detail in Sections F.7 and F.6, respectively.

Algorithm 4 HERAKLES

Require: π_{HL} the high-level policy, π_{LL} the low level policy, M, the MAGELLAN module (Gaven et al., 2025), $\mathcal{G}$ the goal space to be learned, $\mathcal{A}$ set of low level sensory motor commands, $\mathcal{B}_{\text{LL}}$ the buffer for the transitions of the low-level policy, and *env* the environment.

$(obs^{visual}, obs^{textual}) \leftarrow env.reset()$ ▷ see Appendix F.1.4
$g \leftarrow M(obs^{textual})$ ▷ Choose a goal g in an autotelic way
$done_g, r^g \leftarrow env.verifier(g)$ ▷ Verifier checks if g is done, r^g reward for g
while $done_g$ not `True` **do**
  $\tilde{\mathcal{G}} \leftarrow \{\}$ ▷ Instantiate the skill set
  for $g' \in \mathcal{G}$ **do** ▷ Adding a skill with probability $p_{g'}$
    $\tilde{\mathcal{G}} \leftarrow \tilde{\mathcal{G}} \cup \{g'\} \sim p_{g'} = \max(\mathbb{E}_{\pi^{LL}(.|g), obs^{visual} \sim \mathcal{D}_{\pi^{LL}}}[r^g | obs^{visual}], \varepsilon)$
  $\tilde{g} \leftarrow \pi_{HL}(g, obs^{textual}, \tilde{\mathcal{G}})$
  $done_g, r^g \leftarrow env.verifier(g)$
  $done_{\tilde{g}}, r^{\tilde{g}} \leftarrow env.verifier(\tilde{g})$
  while $done_{\tilde{g}}$ not `True` or $done_g$ not `True` **do** ▷ π_{LL} executes the skill in the environment
    $a \leftarrow \pi_{LL}(\tilde{g}, obs^{visual})$ ▷ $a \in \mathcal{A}$
    $(obs^{visual}_{next}, obs^{textual}_{next}) \leftarrow env.step(a)$
    $done_g, r^g \leftarrow env.verifier(g)$
    $done_{\tilde{g}}, r^{\tilde{g}} \leftarrow env.verifier(\tilde{g})$
    $\mathcal{B}_{\text{LL}} \leftarrow (obs^{visual}, a, r^{\tilde{g}})$ ▷ Transition seen as targeting $\tilde{g}$
    $\mathcal{B}_{\text{LL}} \leftarrow (obs^{visual}, a, r^{g})$ ▷ Transition seen as targeting g
    $obs^{visual} \leftarrow obs^{visual}_{next}$
    if enough new transitions in $\mathcal{B}_{\text{LL}}$ **then**
      Update π_{LL} on $\mathcal{B}_{\text{LL}}$ ▷ Using AWR (Peng et al., 2019)
  Update π_{HL} on the high level transition $(obs^{textual}, \tilde{g}, r^g)$ condition by g ▷ Using POAD (Wen et al., 2024b)
  $obs^{textual} \leftarrow obs^{textual}_{next}$

13.5 Experiments

13.5.1 Experimental setup

Environment description

We evaluate our method in Crafter (Hafner, 2022), a procedurally generated and partially observable Minecraft-like 2D environment. It provides a rich and structured goal space, organized hierarchically within an achievement tree (see Appendix F.1.1). At its foundation, we leverage environment variables (i.e., entities such as objects and tools) to define atomic goals such as “go to X” or “collect/place X” which serve as building blocks for more complex achievements. Achieving higher-level goals often require agents to reuse or recombine previously acquired artifacts and skills—for example, "make wood pickaxe" first requires constructing a crafting table. This hierarchical design captures a locally open-ended setting, where agents progressively unlock new learnable goals through interaction, despite the environment’s finite scope. Such a setup makes Crafter an effective proxy for studying open-ended agents (Edward et al., 2024), as it mirrors the continual discovery and reuse

of artifacts central to open-endedness. Thus, Crafter offers a controlled but still expressive testbed to evaluate how efficiently agents can scaffold knowledge, generalize to semantically or structurally novel tasks, and exploit compositional skill reuse as goal complexity increases.

Similarly to Du et al. (2023) we modify the Crafter game in three ways. First, we convert the *'do'* action that originally represents several types of actions (meaning *'attack'* in front of a zombie but *'eat'* in front of a plant) into distinct actions (such as *'eat plant'* or *'attack zombie'*). This disambiguates these elementary actions for LLMs. Second, we remove enemies such as zombies or skeletons. More details are given in Appendix F.1.2. Finally, we make the environment generate textual description of the scene. Thus, an observation $o \in \Omega$ is a tuple made from visual and textual observations $o = (o_v, o_{text})$ (see Appendix F.1.4 for more details). π^{HL} receives the linguistic description o_{text} and π^{LL} uses the visual observation o_v.

Goal Sampler

In Crafter, the agent is not assigned a single main goal but is instead encouraged to explore and master as many goals as possible. These goals are structured within an achievement tree (Appendix F.1.1). The difficulty of a goal $g \in \mathcal{G}$ is not fixed, but conditioned on the current state of the environment s_k. For example, the goal *'place table'* becomes substantially easier if the agent already possesses wood in its inventory.

To enable the agent to select a goal g conditionally to a state s_T at a time step T that maximizes the hierarchical policy competence across the large goal space $\mathcal{G}$ of Crafter, we draw inspiration from methods based on LP. These approaches prioritize goals that are expected to yield the greatest future improvement through practice (Oudeyer et al., 2007). Specifically, we implement the MAGELLAN algorithm (Gaven et al., 2025), which equips LLM agents with the ability to predict their current competence and learning progress online. Moreover, MAGELLAN enhances LP estimation by leveraging semantic relationships between goals, thus facilitating generalization across structurally related skills. Implementation details of MAGELLAN within our framework are provided in Appendix F.7.

Training HL and LL policies

We model π^{HL} using Mistral 7B and train it with the POAD on-policy RL algorithm (Wen et al., 2024b). At each step of the high-level we sample a skill inside the current skill space using constrain decoding. For π^{LL}, we adapt the ResNet-based architecture given by (Moon et al., 2023) ($2M$ parameters). We train π^{LL} using the AWR off-policy algorithm (Peng et al., 2019) and fill the buffer with both types of trajectories described in Section 13.4. Appendices F.4 and F.5 give more details on the architectures and training of π^{HL} and π^{LL}.

Baselines

In order to measure the impact of HRL in HERAKLES, we compare our method with POAD (Wen et al., 2024b) which corresponds to only considering the high-level policy of HERAKLES and restricting its action space to the set of elementary actions A. A detailed

description of POAD's setup is given in Appendix F.8.2. To evaluate the interest of using a skill space stuctured by language, we evaluate HERAKLES against FuN (Vezhnevets et al., 2017) a classical feudal HRL method where the high-level policy learns to generate subgoals in an embedding space. These subgoals are passed to guide the low-level policy in direction of the goal g. To have the same automatic curriculum learning than with HERAKLES, we modify the initial setup of FuN making its high-level policy goal-conditioned. To do so, we pass to the high-level an embedding of the goal in addition to the current observation. The embeddings of the goals are generated with the same LLM as the one used by HERAKLES, more details about FuN' training are available in Appendix F.8.1. In addition to the experiments with the baselines presented here, we present the results of several ablation experiments in Appendix F.3.2.

13.5.2 Heterogeneous goal space

A characteristic of Crafter's goal space is its heterogeneous level of difficulty and compositionality. Such goal spaces are typically difficult to learn, as the number of elementary actions to chain for solving goals explodes as the difficulty of goals grows.

To investigate how HERAKLES' performance scales with the difficulty of goals, we measure the number of high-level steps necessary for an agent to master a goal depending on its difficulty. In this experiment, a goal is mastered when the agent obtains a success rate superior to 0.8. To assess the difficulty of a goal, we measure how many *collect* `materials`, *place* `{table, furnace}` and *craft* `object` are required to reach it. For instance, the goal *'place table'* that requires 2 *collect* `wood` and 1 *place* `table` is considered easier than *'make wood pickaxe'* which requires 3 *collect* `wood`, 1 *place* `table`, and 1 *craft* `wood pickaxe`. We expand on our measure of the difficulty of goals in Appendix F.1.3.

Figure 13.3 shows the results for 4 goals of increasing difficulty. Only HERAKLES succeeds in learning the goals within less than $30,000$ high level steps. The number of steps grows quasi-linearly for HERAKLES, whereas POAD and FuN have a much higher growth rate, even at low difficulty. Such a behaviour is expected from non-hierarchical methods such as POAD which cannot rely on abstractions to explore the environment and must accumulate elementary actions. Comparing HERAKLES and FuN, we see that using a skill space structured by language helps the hierarchical agent to capture the relation between skills and reuse them to foster efficient learning.

13.5.3 Sample efficiency

In order to study the sample efficiency we train our agents for $30,000$ high level steps in the Crafter environment. We re-use the Crafter score defined by Hafner (2022) as:

$$S_c = \exp(\frac{1}{N}\sum_{i=1}^{N}\ln(1+sr_i)) - 1,$$

where $sr_i \in [0;100]$ is the agent's success rate for the goal i and $N = 10$ is the number of goals in the environment. It summarizes the agent's performance by aggregating goal

success rates using a geometric mean, which emphasizes rare and difficult achievements over frequently attained ones, providing a difficulty-aware metric without requiring prior knowledge of goal complexity. We measure the success rate of the agent on a goal by testing it in 40 held-out environments at regular intervals.

Figure 13.4 shows that HERAKLES quickly masters a large number of goals within the Crafter heterogeneous goal space. When a goal is mastered by the hierarchical agent and compiled in the form of a skill in the low-level policy, this skill can be re-exploited to quickly discover and master new more complex goals. On the other hand, POAD quickly plateaus at a low score of $S_c = 5$, remaining below the performance of a random policy over elementary actions.

The increasing difficulty of exploration required to solve more complex goals explains this plateau. Notably, POAD's success rate improves on few easier goals (such as *'go to tree'* with $sr_{\text{'go to tree'}} = 1.0 \pm 0.0$ at the end of training) without making much progress on more difficult ones (such as *'make wood pickaxe'* with $sr_{\text{'make wood pickaxe'}} = 3.9 \pm 6.8\,10^{-3}$ at the end of training). The hierarchical FuN agent makes very slow progress, obtaining results slightly above the random policy. This can be explained by the fact that, even though FuN can make temporal abstractions to learn and master a goal, it is not capable of reusing a previously mastered goal to reach another goal. More details about the evolution of sr for each goal and each method are in Appendix F.2.1.

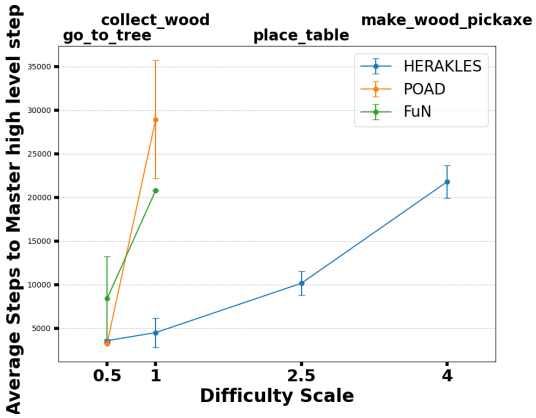


Figure 13.3: Average number of low level steps to master a goal as a function of the difficulty of that goal. Error bars account for standard deviation over 4 seeds. HERAKLES achieves harder goals much faster.

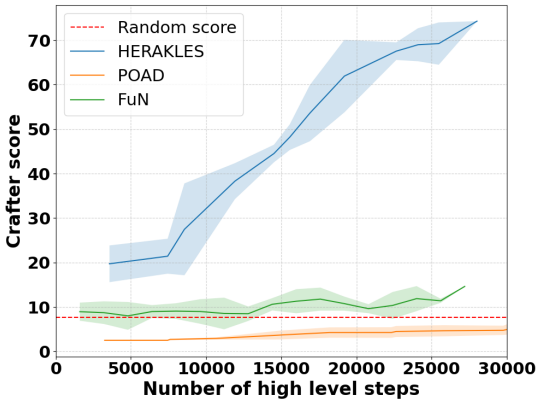


Figure 13.4: Number of goals reached as a function of the number of high level steps. Shaded area standard deviation based on 4 seeds. HERAKLES is the only method whose number of goals grows linearly with time.

13.5.4 Generalization to new goals

As HERAKLES is a language-based method using an LLM as a high-level policy, this section examines its ability to generalize to new goals. In our experiments, we are interested in 3 types of generalization:

• Compositional generation, where the goal that tests the generalization asks for repeating the same goal from the goal space k times (in this case *'collect k woods'*, where k is the number of repetitions).

• Synonymous goals, the verbs in the goal were replaced by a synonym.

• Goals that have a similar trajectory. In this experiment, we looked at the never-seen-before *'make wood sword'* goal, which is as difficult as *'make wood pickaxe'* seen during training. Both goals require the agent to make a table beforehand and face it with wood in its inventory. The only difference between the two goals is the last elementary action necessary to complete them, one being *craft* `wood sword` and the other being *craft* `wood pickaxe`.

Synonym generalization We assess the generalization performance of HERAKLES, POAD, and FuN on a set of synonym-based goals. For each original goal, such as *"collect wood"*, we define a synonym set by selecting five alternative formulations (e.g., *"gather wood"*, *"harvest wood"*, *"procure wood"*, *"acquire wood"*, and *"amass wood"*), and compute the average Crafter score across these variants. In Appendix F.2.2, Table F.2 lists the complete set of synonym categories used in our evaluation. Of note, certain words such as *"acquire"* appear across multiple categories (e.g., for both *"collect"* and *"make"*), requiring the agent to rely on contextual cues to disambiguate the intended task—a key competency in linguistically grounded goal spaces.

We quantify generalization performance by extending the Crafter score for synonym reformulations as:

$$S_c^{syn}(t) = \exp\left(\frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}}\frac{1}{|Syn(g)|}\sum_{g'\in Syn(g)}\ln(1+sr_{g'}(t))\right) - 1,$$

where $Syn(g)$ stands for the set of possible reformulations of goal $g \in \mathcal{G}$ given the list of synonyms given in Table F.2, and $sr_{g'}(t)$ is the average success rate over a reformulated goal $g' \in \bigcup_{g\in\mathcal{G}} Syn(g)$ at step t of the training. The rate $sr_{g'}(t)$ is estimated by performing 8 runs using the model at training step t, with the high-level policy conditioned on the goal g'. As shown in Table 13.2 HERAKLES experiences only a 16% drop in average score relative to the original goal space, POAD and FuN exhibit more substantial decreases of 24% and 27%, respectively. These results highlight the superiority of HERAKLES in handling semantic variability in goal specification.

n-CompositionalityA second form of generalization we investigate is *n-compositionality*, which evaluates an agent's ability to scale learned behaviours to repeated instances of the same subgoal. In this setting, we construct new environments where each original goal must be achieved n times consecutively, with $n \in 2, 3, 4$. For example, the original goal *"collect wood"* is reformulated as *"collect 2 woods"* in the 2-compositional environment. This setup tests the compositional depth of the agent's policy and its capacity to generalize beyond

one-shot achievement. We exclude *"go to X"* goals from the n-compositional environment, as repeating these goals lacks meaningful interpretation—one cannot meaningfully go to the same object n times in succession. The Crafter scores of all the methods are given in Table 13.1.

We observe that HERAKLES exhibits a significantly smaller relative performance drop compared to the baselines. For instance, at $n = 4$, HERAKLES incurs a 54% reduction in Crafter score, whereas POAD suffers a 92% drop and FuN a 84%. These results indicate that HERAKLES is better equipped to generalize through compositional repetition, a key property for agents operating in scalable, open-ended settings.

Goals with shared trajectory We analyze the results on how each method generalises from *'make woof pickaxe'* to *'make wood sword'*. The two goals share the same trajectory ('collect 3 woods', 'place table') excepted for the last action (either 'craft wood pickaxe' or 'craft wood sword'). We see that the Crafter score of HERAKLES goes from 0.93 ± 0.25 for *'make wood pickaxe'* to 0.69 ± 0.46 *'make wood sword'* retaining most of its previous performances. Conversely FuN goes from 0.1 ± 0.25 to 0.00 ± 0.00 and POAD is not considered for this generalization test as it fails to complete *'make wood pickaxe'*.

When analyzing performance across all generalization settings—synonym-based, n-compositional and shared trajectory—we observe complementary strengths in the two baseline methods. Although POAD exhibits a lower overall average Crafter score, it often outperforms FuN on synonym generalization tasks. This is likely attributable to the linguistic capabilities embedded in its LLM backbone, in contrast to FuN, which relies solely on a static embedding for goal representation. Conversely, FuN's hierarchical architecture enables stronger performance in the n-compositional setting, where the reuse and chaining of subskills are crucial. Notably, HERAKLES, which integrates the strengths of both approaches—leveraging both a linguistically grounded high-level policy and a hierarchical skill structure—consistently outperforms both baselines across all generalization regimes.

Table 13.1: Generalization for n-compositional goals.

	HERAKLES	POAD	FuN
Original goals	35.4	1.25	27.3
2 Compositional	29.6 (**-16%**)	0.396 (**-68%**)	11.6 (**-58%**)
3 Compositional	22.3 (**-37%**)	0.231 (**-82%**)	7.09 (**-74%**)
4 Compositional	16.0 (**-54%**)	0.102 (**-92%**)	4.24 (**-84%**)

	HERAKLES	POAD	FuN
Original goals	49.4	3.3	27.3
Synonyms	41.5 (**-16%**)	2.5 (**-24%**)	19.9 (**-27%**)

Table 13.2: Generalization for synonyms goals.

13.6 Discussion

In this chapter, we introduced HERAKLES, a hierarchical framework designed for open-ended autotelic agents, in which high-level and low-level policies are trained concurrently. As the agent progressively masters more complex goals, learned behaviours are incrementally compiled into the low-level policy as reusable skills. This continual expansion of the agent's behavioural repertoire enables the high-level policy to operate effectively in dynamic, heterogeneous goal spaces. Through extensive experiments in the Crafter environment, we showed that HERAKLES (i) scales nearly linearly with goal difficulty, (ii) exploits previously compiled skills to accelerate the acquisition of new goals, and (iii) demonstrates strong generalisation to novel and unseen tasks.

Despite these promising outcomes, several limitations merit further attention. Currently, the high-level policy in HERAKLES relies exclusively on textual goal descriptions. Extending this capability to visual or multimodal goals—using VLMs (Liu et al., 2024)—represents a natural and impactful avenue for future research. Additionally, while our framework incrementally compiles new skills into the low-level policy, retraining it on the entire set of previously acquired behaviours introduces a computational overhead that may hinder long-term scalability. Efficient strategies for continual skill consolidation, potentially inspired by lifelong learning and memory-based methods, could help mitigate this cost.

Furthermore, the process of compiling increasingly complex hierarchical behaviours into robust and general low-level skills remains an open challenge. Such a process requires ensuring both the stability and transferability of learned behaviours as the agent encounters more sophisticated tasks. Lastly, our current setting presupposes the existence of a goal space generated by an external open-ended agent. A promising direction for future work is the integration of HERAKLES with automatic goal generation methods (Faldor et al., 2025; Pourcel et al., 2024a), as introduced in Chapter 9. Coupling goal discovery with skill acquisition and hierarchical abstraction would bring us closer to truly autonomous open-ended agents capable of lifelong learning in complex and evolving environments.

Chapter 14

Discussion

The two methods presented in this part address the challenge of efficiently learning goals in complex environments. Both approaches leverage the structural and generalisation properties of language, through linguistic abstractions in the case of EAGER, and via the use of a LLM in the case of HERAKLES.

Chapter 12 introduces a reward shaping technique, EAGER, that exploits linguistic cues to decompose high-level goals into intermediate sub-objectives. Specifically, it generates fill-in-the-blank questions derived from the natural language formulation of a goal, and rewards the agent when it is able to answer these questions correctly as it progresses along a trajectory. A series of experiments in sparse reward environments demonstrate that EAGER significantly improves sample efficiency when compared to existing baselines such as ELLA (Mirchandani et al., 2021) and RIDE (Raileanu & Rocktäschel, 2020). The method also proves effective for temporally constrained tasks, where specific sub-goals must be completed in a prescribed order (e.g., "before" or "after" relationships). However, EAGER relies on relatively simple linguistic templates and does not yet fully exploit the expressive and inferential power of language.

Chapter 13 presents HERAKLES, a hierarchical reinforcement learning framework grounded in language. The agent is composed of two interacting components: a high-level policy instantiated by an LLM, and a low-level policy implemented as a smaller neural network. Initially, the low-level agent is limited to executing primitive actions. As training progresses, policies learned for individual goals are compiled into reusable skills, expanding the behavioural repertoire of the low-level agent. The LLM learns to orchestrate this growing skill set, enabling the agent to solve increasingly complex tasks by composing previously acquired behaviours. This skill compilation mechanism is particularly advantageous, as it mitigates computational overhead by reducing the frequency of calls to the high-level LLM.

These two approaches represent distinct yet complementary strategies for enhancing the learning efficiency of RL agents through language. Nevertheless, both currently rely on expert input: EAGER presupposes knowledge of appropriate linguistic templates for question generation, while HERAKLES assumes access to a sufficiently structured and dense goal space. Yet, the promise of open-ended agents lies precisely in their ability to explore domains where human expertise is limited or unavailable, and to autonomously discover novel goals and learning strategies.

For such agents to emerge, the proposed methods must evolve to operate with minimal supervision and maximal reuse of acquired knowledge. In particular, the techniques enabling efficient task acquisition should integrate insights from both the modelling of environment dynamics (Part II) and the exploration of the goal space (Part III). In this regard, HERAKLES already exemplifies a promising direction: its high-level policy, implemented via a LLM fine-tuned through reinforcement learning, can ground the agent's understanding of its environment—as demonstrated in GLAM (Chapter 4)—while also benefiting from goal selection strategies derived from MAGELLAN (Chapter 8). This convergence between language, skill composition, and autotelic behaviour defines a coherent framework for building open agents capable of lifelong learning in complex and dynamic domains.

Part V

Conclusion

Chapter 15

Conclusion

As we come to the end of this thesis, we first briefly summarize the contributions that have been presented. We then present recent developments in the fields of agents and languages and look at what is currently lacking in order to have open-ended agents. Finally, we present some thoughts on what a world with efficient open-ended agents and their interactions with humans might look like.

15.1 Summary of contributions

This thesis has explored a suite of complementary, language-based methods for enabling open-ended agents to **model their environments**, **explore unbounded goal spaces**, and **efficiently acquire new skills**. Across the three parts, we have progressively built towards a unified framework for autonomous agents capable of continual learning and adaptation in complex, dynamic domains.

Modelling and Understanding the Environment.

In Part II, we introduced two complementary approaches to endow agents with structured models of their surroundings. First, GLAM functionally grounds LLM-based agents through reinforcement learning, aligning internal symbols with the relational structure of the environment and enabling generalization without environment-specific architectures. Second, WorldLLM fosters scientific model-building via an adversarial loop between an RL agent and a hypothesis-generating LLM, uncovering structural patterns without handcrafted theory spaces. Together, these methods address both low-level grounding and high-level hypothesis formation, suggesting a two-phase integration where explicit hypotheses scaffold early learning before being internalized through grounded interaction.

Exploring and Structuring the Goal Space.

Part III addressed the challenge of navigating an unbounded, evolving set of goals. MAGELLAN equips agents with metacognitive competence estimation, enabling autonomous curriculum construction and adaptive learning trajectories. AutotelicLLM extends this by generating creative, learnable goals expressed in natural language and executable reward

functions. These methods jointly tackle the dual problem of *expanding* the goal space in a meaningful way and *organising* it for efficient learning.

Accelerating Skill Acquisition.

In Part IV, we proposed two strategies to improve learning efficiency in complex environments. EAGER leverages linguistic cues to derive intermediate sub-objectives from high-level goals, boosting sample efficiency in sparse-reward and temporally constrained settings. HERAKLES adopts a hierarchical architecture where an LLM-driven high-level policy composes and reuses skills learned by a low-level agent, enabling increasingly complex behaviours while reducing computational overhead. Both methods highlight the role of language in structuring learning signals and orchestrating skill composition.

15.2 Limitations

In this section, we acknowledge the limitations of the approaches developed in this work. In particular, we discuss constraints related to the action space, limitations in handling observations, and the reliance on expert knowledge.

In GLAM and MAGELLAN, we focus on agents with finite action spaces. For instance, we allow *go forward* as an action but exclude synonymous formulations such as *move forward* or *go forward one step*. In HERAKLES, we also work with a fixed action space, although its size may vary during the experiment as new sub-goals are introduced. The aim of these choices was to simplify the configuration in order to study the properties of our mechanisms independently of efficiency constraints or the mapping between the agent's unconstrained action space and that of the environment. However, if we aim to apply these methods to realistic open-ended agents, such constraints must be lifted. For example, in problems requiring open-ended code generation, constraining the action space is simply infeasible. One possible solution, as proposed by Wen et al. (2024b), is to consider each token as an action. If necessary, we could enforce template adherence (e.g., through reward shaping) to ensure that generated actions can be reliably mapped to executable behaviors in the environment. Another limitation lies in agent decision-making: actions are taken without using language to deliberate on which action should be chosen. By restricting responses to 0-shot completions, we limit the ability of LLM-based agents to identify successful trajectories. Allowing reflection before selecting an action, as in Guo et al. (2025), could improve action selection—a capability that becomes especially critical in environments with very large action spaces.

A second limitation of approaches such as GLAM, MAGELLAN, and HERAKLES is that they rely on a purely linguistic observation space. Many environments, including BabyAI (Chevalier-Boisvert et al., 2019) and Crafter (Hafner, 2022), provide rich visual information. To make them usable with an LLM, we manually created captioners and embedded them in the environments, yielding, for example, BabyAI-Text. However, real-world environments can produce diverse observation modalities, and effectively integrating these into the core of an LLM remains a challenge. One possible solution is to train captioners specialized in producing LLM-tailored descriptions (Aissi et al., 2025a). Another direction is to extend our

methods to vision-language models (VLMs). More generally, we can imagine realistic settings requiring the integration of multiple modalities (visual, auditory, spatial, etc.). Furthermore, in GLAM and HERAKLES, we restrict the size of the observation history for computational reasons. This is feasible in simple environments that do not require long-term memory to select actions, but in more complex settings such as NetHack (Küttler et al., 2020), the ability to store and retrieve past observations when most relevant would be critical. Techniques such as retrieval-augmented generation (RAG) (Lewis et al., 2020) could address this limitation by enhancing long-term memory capabilities.

Another limitation of the presented work is the reliance on expert knowledge for certain components of the proposed methods. In EAGER, we use manually crafted template questions and answers based on limited language abstractions. While feasible in relatively simple environments, extending this approach to more complex domains such as Minecraft would require the automatic generation of large sets of questions, potentially with LLMs generating goal-relevant queries conditioned on the context. In AutotelicLLM, to prevent goal collapse, our method requires expert-defined categories constraining the generation of goals. This restricts the approach to environments where humans already possess a sufficiently accurate understanding of the goal space. However, in many domains—such as scientific discovery—the goal space is not known in advance. Instead, humans progressively create new goal categories based on their prior knowledge. Replacing expert-defined categories with those generated by LLMs, thereby leveraging human-like priors on promising goals, would be closer in spirit to the approach of (Zhang et al., 2024b). Moreover, HERAKLES and MAGELLAN assume access to the complete goal space, which is unrealistic in truly open-ended environments. Still, both methods can accommodate a dynamic goal space and could be combined with goal-generation techniques. Finally, MAGELLAN resorts to ε-greedy exploration when all goals are estimated to have similar ALP. While this is viable in a finite goal space, it breaks down in continuous goal spaces (e.g., the joint positions of a robotic arm).

In the introduction of this thesis, we have defined three orthogonal dimensions that are required for an open-ended agent: the ability to model and internalize its environment, a continual exploration of the goal space in new directions, and efficiently acquiring new skills. We have presented several methods, each dealing with one of these dimensions. In this section, we acknowledged their current limitations and we discussed ways to scale these approaches. We emphasize that to scale, our methods would need to operate within less constrained action spaces, handle a richer diversity of observations through improved memory management, and function without reliance on expert input. Our work has opened many avenues for further work. Despite these challenges, our work opens promising directions and provides a foundation for further research. In the following section, we discuss potential avenues expanding and strengthening these contributions.

15.3 Future Directions

This thesis has primarily studied tools that address orthogonal dimensions of open-ended agents. A natural next step is to investigate how these tools can operate in synergy. HERAKLES was a first step in this direction, reusing MAGELLAN with a GLAM-like

LLM agent. An extension could be to consider a setting in which AutotelicLLM generates goals for HERAKLES. Furthermore, an improved version of WorldLLM could generate rules that are passed to the different components of the agent. In this way, as the agent progresses in the environment and discovers or constructs new artifacts, it could update its world model with new rules and directly reuse them to expand exploration in new directions.

Another promising direction is to better exploit the embedded goal space learned by MAGELLAN. This space not only encodes semantic relations between goals but also their estimated success probabilities. In our work, MAGELLAN is used to place a new goal and estimate its likelihood of success. A natural extension would be to sample directly from this embedding space and generate new goals from the sampled points. Another way to leverage this space is to acknowledge its nature as a scalar field and study its evolution over time. This would allow us to guide exploration toward a particular set of goals. The objective would be to identify the most direct trajectory—defined by the number of goals sampled during training—that enables the agent to move from an initial region, where most goals have low success probabilities, to a target region where the desired goals are highly achievable. Such a process would lead to directed exploration, which is especially valuable when the types of goals are known but the training pathway is unclear, and when the environment is too rich and complex for purely open-ended methods to reliably discover and master the required behaviors within a reasonable timeframe.

Bibliography

Abdou, M., Kulmizev, A., Hershcovich, D., Frank, S., Pavlick, E., and Søgaard, A.
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.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H.,

Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B.
Gpt-4 technical report, 2024.
URL https://arxiv.org/abs/2303.08774.

Addyman, C. and Mareschal, D.
Local redundancy governs infants' spontaneous orienting to visual-temporal sequences.
Child Development, 84(4):1137–1144, 2013.
ISSN 00093920, 14678624.
URL http://www.jstor.org/stable/23469368.

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S.
Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms.
In Ku, L., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 12248–12267. Association for Computational Linguistics, 2024.
doi: 10.18653/V1/2024.ACL-LONG.662.
URL https://doi.org/10.18653/v1/2024.acl-long.662.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A.
Do as i can, not as i say: Grounding language in robotic affordances, 2022.
URL https://arxiv.org/abs/2204.01691.

Ahuja, A., Kopparapu, K., Fergus, R., and Dasgupta, I.
Hierarchical reinforcement learning with natural language subgoals, 2023.
URL https://arxiv.org/abs/2309.11564.

Aissi, M. S., Grislain, C., Chetouani, M., Sigaud, O., Soulier, L., and Thome, N.
VIPER: visual perception and explainable reasoning for sequential decision-making.
CoRR, abs/2503.15108, 2025a.
doi: 10.48550/ARXIV.2503.15108.
URL https://doi.org/10.48550/arXiv.2503.15108.

Aissi, M. S., Romac, C., Carta, T., Lamprier, S., Oudeyer, P., Sigaud, O., Soulier, L., and Thome, N.
Reinforcement learning for aligning large language models agents with interactive environments: Quantifying and mitigating prompt overfitting.
In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 7030–7046. Association for Computational Linguistics, 2025b.
doi: 10.18653/V1/2025.FINDINGS-NAACL.390.
URL https://doi.org/10.18653/v1/2025.findings-naacl.390.

Akakzia, A., Colas, C., Oudeyer, P.-Y., Chetouani, M., and Sigaud, O.
Grounding language to autonomously-acquired skills via goal generation.
In *International Conference on Learning Representations*, 2021.

Aki, F., Ikeda, R., Saito, T., Regan, C., and Oka, M.
Llm-poet: Evolving complex environments using large language models.
In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 243–246, 2024.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Binkowski, M., Barreira, R., Vinyals, O., Zisserman, A., and Simonyan, K.
Flamingo: a visual language model for few-shot learning.
ArXiv, abs/2204.14198, 2022.

Alderson-Day, B. and Fernyhough, C.
Inner speech: Development, cognitive functions, phenomenology, and neurobiology.
Psychological Bulletin, 141(5):931–965, September 2015.
doi: 10.1037/bul0000021.

Allen, C. and Bekoff, M.
Species of Mind: The Philosophy and Biology of Cognitive Ethology.
MIT Press, Cambridge, MA, 1999.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., and Mané, D.
Concrete problems in AI safety.
CoRR, abs/1606.06565, 2016.
URL http://arxiv.org/abs/1606.06565.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Henge, A.
Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments.
In *CVPR*, 2018.

Andreas, J.
Language models as agent models.

In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 5769–5779. Association for Computational Linguistics, 2022.
doi: 10.18653/V1/2022.FINDINGS-EMNLP.423.
URL https://doi.org/10.18653/v1/2022.findings-emnlp.423.

Andreas, J., Klein, D., and Levine., S.
Modular multitask reinforcement learning with policy sketches.
In *International Conference on Machine Learning (ICML)*, 2017.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W.
Hindsight experience replay.
In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

Assran, M., Bardes, A., Fan, D., Garrido, Q., Howes, R., Mojtaba, Komeili, Muckley, M., Rizvi, A., Roberts, C., Sinha, K., Zholus, A., Arnaud, S., Gejji, A., Martin, A., Hogan, F. R., Dugas, D., Bojanowski, P., Khalidov, V., Labatut, P., Massa, F., Szafraniec, M., Krishnakumar, K., Li, Y., Ma, X., Chandar, S., Meier, F., LeCun, Y., Rabbat, M., and Ballas, N.
V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025.
URL https://arxiv.org/abs/2506.09985.

Aubret, A., Matignon, L., and Hassas, S.
A survey on intrinsic motivation in reinforcement learning.
arXiv preprint arXiv:1908.06976, 2019.

Ay, N. and Polani, D.
Information flows in causal networks.
Adv. Complex Syst., 11(1):17–41, 2008.
doi: 10.1142/S0219525908001465.
URL https://doi.org/10.1142/S0219525908001465.

Azar, M. G., Guo, Z. D., Piot, B., Munos, R., Rowland, M., Valko, M., and Calandriello, D.
A general theoretical paradigm to understand learning from human preferences.
In Dasgupta, S., Mandt, S., and Li, Y. (eds.), *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*, volume 238 of *Proceedings of Machine Learning Research*, pp. 4447–4455. PMLR, 2024.
URL https://proceedings.mlr.press/v238/gheshlaghi-azar24a.html.

Ba, J. L., Kiros, J. R., and Hinton, G. E.
Layer normalization.
arXiv preprint arXiv:1607.06450, 2016.

Bacon, P.-L., Harb, J., and Precup, D.
The option-critic architecture.
Proceedings of the AAAI Conference on Artificial Intelligence, 31(1), Feb. 2017.
doi: 10.1609/aaai.v31i1.10916.
URL https://ojs.aaai.org/index.php/AAAI/article/view/10916.

Bagaria, A., Senthil, J. K., and Konidaris, G.
Skill discovery for exploration and planning using deep skill graphs.
In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 521–531. PMLR, 2021.
URL http://proceedings.mlr.press/v139/bagaria21a.html.

Bahdanau, D., Cho, K., and Bengio, Y.
Neural machine translation by jointly learning to align and translate.
In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
URL http://arxiv.org/abs/1409.0473.

Bahdanau, D., Hill, F., Leike, J., Hughes, E., Hosseini, S. A., Kohli, P., and Grefenstette, E.
Learning to understand goal specifications by modelling reward.
In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
URL https://openreview.net/forum?id=H1xsSjC9Ym.

Baldassarre, G. and Mirolli, M.
Intrinsically motivated learning systems: an overview.
Intrinsically motivated learning in natural and artificial systems, pp. 1–14, 2012.

Baldassarre, G., Stafford, T., Mirolli, M., Redgrave, P., Ryan, R. M., and Barto, A.
Intrinsic motivations and open-ended development in animals, humans, and robots: An overview.
Frontiers in Psychology, 5:985, 2014.

Balzeau, A., Gilissen, E., Holloway, R. L., Prima, S., and Grimaud-Hervé, D.
Variations in size, shape and asymmetries of the third frontal convolution in hominids: Paleoneurological implications for hominin evolution and the origin of language.
Journal of Human Evolution, 76:116–128, 2014.
ISSN 0047-2484.
doi: https://doi.org/10.1016/j.jhevol.2014.06.006.
URL https://www.sciencedirect.com/science/article/pii/S0047248414001535.

Baranes, A. and Oudeyer, P.-Y.
R-IAC: Robust intrinsically motivated exploration and active learning.
IEEE Transactions on Autonomous Mental Development, 1(3):155–169, 2009.
ISSN 1943-0612.
doi: 10.1109/TAMD.2009.2037513.
Conference Name: IEEE Transactions on Autonomous Mental Development.

Baranes, A. and Oudeyer, P.-Y.
Active learning of inverse models with intrinsically motivated goal exploration in robots.
Robotics and Autonomous Systems, 61(1):49–73, 2013.
ISSN 0921-8890.
doi: https://doi.org/10.1016/j.robot.2012.05.008.
URL https://www.sciencedirect.com/science/article/pii/S0921889012000644.

Barreto, A., Hou, S., Borsa, D., Silver, D., and Precup, D.
Fast reinforcement learning with generalized policy updates.
Proc. Natl. Acad. Sci. USA, 117(48):30079–30087, 2020.
doi: 10.1073/PNAS.1907370117.
URL https://doi.org/10.1073/pnas.1907370117.

Bauer, J., Baumli, K., Baveja, S., Behbahani, F., Bhoopchand, A., Bradley-Schmieg, N., Chang, M., Clay, N., Collister, A., Dasagi, V., Gonzalez, L., Gregor, K., Hughes, E., Kashem, S., Loks-Thompson, M., Openshaw, H., Parker-Holder, J., Pathak, S., Perez-Nieves, N., Rakicevic, N., Rocktäschel, T., Schroecker, Y., Sygnowski, J., Tuyls, K., York, S., Zacherl, A., and Zhang, L.
Human-timescale adaptation in an open-ended task space, 2023.
URL https://arxiv.org/abs/2301.07608.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R.
Unifying count-based exploration and intrinsic motivation.
In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1471–1479, 2016.
URL https://proceedings.neurips.cc/paper/2016/hash/afda332245e2af431fb7b672a68b659d-Abstract.html.

Bellemare, M. G., Dabney, W., and Munos, R.
A distributional perspective on reinforcement learning.
In *International conference on machine learning*, pp. 449–458. PMLR, 2017.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z.
Autonomous navigation of stratospheric balloons using reinforcement learning.
Nature, 588(7836):77–82, 2020.
doi: 10.1038/s41586-020-2939-8.

Bellman, R.
Dynamic programming.
science, 153(3731):34–37, 1966.

Bender, E. M. and Koller, A.
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*, pp. 5185–5198, Online, July 2020. Association for Computational Linguistics.
doi: 10.18653/v1/2020.acl-main.463.
URL https://aclanthology.org/2020.acl-main.463.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C.
A neural probabilistic language model.
Journal of machine learning research, 3(Feb):1137–1155, 2003.

Benureau, F. C. Y. and Oudeyer, P.
Behavioral diversity generation in autonomous exploration through reuse of past experience.
Frontiers Robotics AI, 3:8, 2016.
doi: 10.3389/FROBT.2016.00008.
URL https://doi.org/10.3389/frobt.2016.00008.

Berlyne, D. E.
A theory of human curiosity.
British Journal of Psychology, 1954.

Bisk, Y., Holtzman, A., Thomason, J., Andreas, J., Bengio, Y., Chai, J., Lapata, M., Lazaridou, A., May, J., Nisnevich, A., Pinto, N., and Turian, J.
Experience Grounds Language.
In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8718–8735, Online, November 2020. Association for Computational Linguistics.
doi: 10.18653/v1/2020.emnlp-main.703.

Blaes, S., Vlastelica Pogančić, M., Zhu, J., and Martius, G.
Control what you can: Intrinsically motivated task-planning agent.
In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Blukis, V., Terme, Y., Niklasson, E., Knepper, R. A., and Artzi, Y.
Learning to map natural language instructions to physical quadcopter control using simulated flight.
In Kaelbling, L. P., Kragic, D., and Sugiura, K. (eds.), *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1415–1438. PMLR, 2019.
URL http://proceedings.mlr.press/v100/blukis20a.html.

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

Bonawitz, E. B., van Schijndel, T. J., Friel, D., and Schulz, L.
Children balance theories and evidence in exploration, explanation, and learning.
Cognitive psychology, 64(4):215–234, 2012.

Bougie, N. and Ichise, R.
Skill-based curiosity for intrinsically motivated reinforcement learning.
Mach. Learn., 109(3):493–512, 2020.
doi: 10.1007/S10994-019-05845-8.
URL https://doi.org/10.1007/s10994-019-05845-8.

Brooks, R. A.
Elephants don't play chess.
Robotics and Autonomous Systems, 6(1):3–15, 1990.
ISSN 0921-8890.
doi: https://doi.org/10.1016/S0921-8890(05)80025-9.
URL https://www.sciencedirect.com/science/article/pii/S0921889005800259.
Designing Autonomous Agents.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T. J., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D.
Language models are few-shot learners.
ArXiv, abs/2005.14165, 2020.

Bryson, A. and Ho, Y.-C.
Applied Optimal Control.
Blaisdell, Waltham, MA, 1969.

Brändle, Franziska Stocks, L. J., Tenenbaum, J. B., Gershman, S. J., and Schulz, E.
Empowerment contributes to exploration behaviour in a creative video game.
Nature Human Behaviour, 7(9):1481–1489, September 2023.
doi: 10.1038/s41562-023-01661-2.

Buchanan, B. G. and Feigenbaum, E. A.
Dendral and meta-dendral: Their applications dimension.
Artificial Intelligence, 11(1):5–24, 1978.
ISSN 0004-3702.
doi: https://doi.org/10.1016/0004-3702(78)90010-3.
URL https://www.sciencedirect.com/science/article/pii/0004370278900103.
Applications to the Sciences and Medicine.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A.
Large-scale study of curiosity-driven learning.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *7th International Conference on Learning Representations (ICLR 2019)*. Curran Associates, Inc., 2019a.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O.
Exploration by random network distillation, 2019b.

Cangelosi, A. and Riga, T.
An embodied model for sensorimotor grounding and grounding transfer: Experiments with epigenetic robots.
Cogn. Sci., 30(4):673–689, 2006.
doi: 10.1207/S15516709COG0000_72.
URL https://doi.org/10.1207/s15516709cog0000_72.

Cangelosi, A. and Schlesinger, M.
From babies to robots: The contribution of developmental robotics to developmental psychology.
Child Development Perspectives, 12(3):183–188, 2018.
ISSN 1750-8592.
doi: 10.1111/cdep.12282.

Cangelosi, A. and Stramandinoli, F.
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.

Cangelosi, A., Metta, G., Sagerer, G., Nolfi, S., Nehaniv, C., Fischer, K., Tani, J., Belpaeme, T., Sandini, G., Nori, F., Fadiga, L., Wrede, B., Rohlfing, K., Tuci, E., Dautenhahn, K., Saunders, J., and Zeschel, A.
Integration of action and language knowledge: A roadmap for developmental robotics.
Autonomous Mental Development, IEEE Transactions on, 2:167 – 195, 10 2010.
doi: 10.1109/TAMD.2010.2053034.

Capasso, L., Michetti, E., and D'Anastasio, R.
A *Homo erectus* hyoid bone: Possible implications for the origin of the human capability for speech.
Collegium Antropologicum, 32(4):1007–1011, December 2008.
ISSN 0350-6134.

Carruthers, P.
Thinking in language? evolution and a modularist possibility.
In Carruthers, P. and Boucher, J. (eds.), *Language and Thought*. Cambridge University Press, Cambridge, 1 edition, 1998.

Carta, T., Oudeyer, P., Sigaud, O., and Lamprier, S.
EAGER: asking and answering questions for automatic reward shaping in language-guided RL.
In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
URL http://papers.nips.cc/paper_files/paper/2022/hash/50eb39ab717507cccbe2b8590de32030-Abstract-Conference.html.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y.
Grounding large language models in interactive environments with online reinforcement learning.
In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3676–3713. PMLR, 2023.
URL http://dblp.uni-trier.de/db/conf/icml/icml2023.html#CartaRWLSO23.

Carta, T., Romac, C., Gaven, L., Oudeyer, P.-Y., Sigaud, O., and Lamprier, S.
Herakles: Hierarchical skill compilation for open-ended llm agents, 2025.
URL https://arxiv.org/abs/2508.14751.

Castanet, N., Sigaud, O., and Lamprier, S.
Stein variational goal generation for adaptive exploration in multi-goal reinforcement learning.
In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3714–3731. PMLR, 2023.

Castanet, N., Sigaud, O., and Lamprier, S.
Imagine beyond! distributionally robust auto-encoding for state space coverage in online reinforcement learning.
CoRR, abs/2505.17830, 2025.
doi: 10.48550/ARXIV.2505.17830.
URL https://doi.org/10.48550/arXiv.2505.17830.

Chater, N. and Vitányi, P.
Simplicity: a unifying principle in cognitive science?
Trends in Cognitive Sciences, 7(1):19–22, 2003.
ISSN 1364-6613.
doi: https://doi.org/10.1016/S1364-6613(02)00005-0.
URL https://www.sciencedirect.com/science/article/pii/S1364661302000050.

Chaudhury, S., Murugesan, K., Carta, T., Talamadupula, K., and Tatsubori, M.
Leveraging visual handicaps for text-based reinforcement learning.
In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024, Seoul, Republic of Korea, April 14-19, 2024*, pp. 12376–12380. IEEE, 2024.
doi: 10.1109/ICASSP48485.2024.10447123.
URL https://doi.org/10.1109/ICASSP48485.2024.10447123.

Che, T., Li, Y., Zhang, R., Hjelm, R. D., Li, W., Song, Y., and Bengio, Y.
Maximum-likelihood augmented discrete generative adversarial networks.
CoRR, abs/1702.07983, 2017.
URL http://arxiv.org/abs/1702.07983.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y.
Babyai: A platform to study the sample efficiency of grounded language learning.
In *International Conference on Learning Representations (ICLR)*, 2019.

Chomsky, N.
Syntactic Structures.
Mouton, The Hague, 1957.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N. M., Prabhakaran, V., Reif, E., Du, N.,

Hutchinson, B. C., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., García, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Díaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K. S., Eck, D., Dean, J., Petrov, S., and Fiedel, N.
Palm: Scaling language modeling with pathways.
ArXiv, abs/2204.02311, 2022.

Christiano, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D.
Deep reinforcement learning from human preferences.
In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4299–4307, 2017.
URL https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

Chua, K., Calandra, R., McAllister, R., and Levine, S.
Deep reinforcement learning in a handful of trials using probabilistic dynamics models.
In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 4759–4770, 2018.
URL https://proceedings.neurips.cc/paper/2018/hash/3de568f8597b94bda53149c7d7f5958c-Abstract.html.

Clark, A.
Magic words: How language augments human computation.
In Carruthers, P. and Boucher, J. (eds.), *Language and Thought*. Cambridge University Press, Cambridge, 1 edition, 1998.

Clark, A. and Grush, R.
Towards a cognitive robotics.
Adapt Behav, 7(1):5–16, December 1999.
ISSN 1059-7123.
doi: 10.1177/105971239900700101.

Clement, B., Roy, D., Oudeyer, P.-Y., and Lopes, M.
Multi-armed bandits for intelligent tutoring systems.
Journal of Educational Data Mining, 7(2), 2015.

Clune, J.
Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence.
CoRR, abs/1905.10985, 2019.
URL http://arxiv.org/abs/1905.10985.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J.
Leveraging procedural generation to benchmark reinforcement learning.
In *International Conference on Machine Learning (ICML)*, 2020.

Colas, C., Sigaud, O., and Oudeyer, P.-Y.
GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning algorithms.
In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1039–1048. PMLR, 10–15 Jul 2018.
URL https://proceedings.mlr.press/v80/colas18a.html.

Colas, C., Oudeyer, P.-Y., Sigaud, O., Fournier, P., and Chetouani, M.
Curious: Intrinsically motivated modular multi-goal reinforcement learning.
In *Proceedings of the 36th International Conference on Machine Learning (ICML)*. PMLR, 2019.
Available on arXiv: arXiv:1810.06284.

Colas, C., Karch, T., Lair, N., Dussoux, J., Moulin-Frier, C., Dominey, P. F., and Oudeyer, P.
Language as a cognitive tool to imagine goals in curiosity driven exploration.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a.
URL https://proceedings.neurips.cc/paper/2020/hash/274e6fcf4a583de4a81c6376f17673e7-Abstract.html.

Colas, C., Karch, T., Lair, N., Moulin-Frier, C., Dussoux, J.-M., Dominey, P. F., and Oudeyer, P.-Y.
Language as a Cognitive Tool to Imagine Goals in Curiosity Driven Exploration.
In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, 2020b.
URL https://arxiv.org/abs/2002.09253.

Colas, C., Karch, T., Moulin-Frier, C., and Oudeyer, P.-Y.
Language and culture internalization for human-like autotelic ai.
Nature Machine Intelligence, 4(12):1068–1076, December 2022a.
ISSN 2522-5839.
doi: 10.1038/s42256-022-00591-4.
URL http://dx.doi.org/10.1038/s42256-022-00591-4.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y.
Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey.
Journal of Artificial Intelligence Research, 2022b.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y.
Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: A short survey.
J. Artif. Int. Res., 74, September 2022c.

ISSN 1076-9757.
doi: 10.1613/jair.1.13554.
URL https://doi.org/10.1613/jair.1.13554.

Colas, C., Teodorescu, L., Oudeyer, P.-Y., Yuan, X., and Côté, M.-A.
Augmenting autotelic agents with large language models.
In Chandar, S., Pascanu, R., Sedghi, H., and Precup, D. (eds.), *Proceedings of The 2nd Conference on Lifelong Learning Agents*, volume 232 of *Proceedings of Machine Learning Research*, pp. 205–226. PMLR, 22–25 Aug 2023.
URL https://proceedings.mlr.press/v232/colas23a.html.

Conti, E., Madhavan, V., Petroski Such, F., Lehman, J., Stanley, K., and Clune, J.
Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents.
In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
URL https://proceedings.neurips.cc/paper_files/paper/2018/file/b1301141feffabac455e1f90a7de2054-Paper.pdf.

Côté, M.-A., Kádár, Á., Yuan, X., Kybartas, B. A., Barnes, T., Fine, E., Moore, J., Hausknecht, M. J., Asri, L. E., Adada, M., Tay, W., and Trischler, A.
Textworld: A learning environment for text-based games.
In *CGW@IJCAI*, 2018.

Craik, K. J. W.
The nature of explanation.
The Philosophical Review, 53(5):503–504, 1944.

Cross, L., Xiang, V., Bhatia, A., Yamins, D. L., and Haber, N.
Hypothetical Minds: Scaffolding Theory of Mind for Multi-Agent Tasks with Large Language Models.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *13th International Conference on Learning Representations (ICLR 2025)*. Curran Associates, Inc., 2025.
URL https://openreview.net/forum?id=otW0TJOUYF.

Csikszentmihalyi, M.
Finding Flow: The Psychology of Engagement with Everyday Life.
Hachette UK, 1997.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R.
Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context, June 2019.
URL http://arxiv.org/abs/1901.02860.
arXiv:1901.02860 [cs, stat].

Dainese, N., Marttinen, P., and Ilin, A.
Reader: Model-based language-instructed reinforcement learning.

In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 16583–16599, Singapore, December 2023. Association for Computational Linguistics.
URL https://aclanthology.org/2023.emnlp-main.1032.

Darlow, L., Regan, C., Risi, S., Seely, J., and Jones, L.
Continuous thought machines.
CoRR, abs/2505.05522, 2025.
doi: 10.48550/ARXIV.2505.05522.
URL https://doi.org/10.48550/arXiv.2505.05522.

Dasgupta, I., Chen, C. K., Marino, K., Ahuja, A., Babayan, S., Hill, F., and Fergus, R.
Collaborating with language models for embodied reasoning.
In *Advances in Neural Information Processing Systems (NeurIPS) LaReL workshop*, 2022.

Dayan, P. and Hinton, G. E.
Feudal reinforcement learning.
In Hanson, S., Cowan, J., and Giles, C. (eds.), *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992.
URL https://proceedings.neurips.cc/paper_files/paper/1992/file/d14220ee66aeec73c49038385428ec4c-Paper.pdf.

de Saussure, F.
Course in General Linguistics.
Duckworth, London, 1983.
(trans. Roy Harris).

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., and Li, S. S.
Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.
CoRR, abs/2501.12948, 2025.
doi: 10.48550/ARXIV.2501.12948.
URL https://doi.org/10.48550/arXiv.2501.12948.

Deisenroth, M. and Rasmussen, C. E.
Pilco: A model-based and data-efficient approach to policy search.
Proceedings of the 28th International Conference on Machine Learning (ICML), 2011.

Denison, S., Reed, C., and Xu, F.
The emergence of probabilistic reasoning in very young infants: evidence from 4.5-and 6-month-olds.

Developmental psychology, 49(2):243, 2013.

Dennett, D. C.
Consciousness Explained.
Penguin UK, London, 1993.

Dennis, M., Jaques, N., Vinitsky, E., Bayen, A. M., Russell, S., Critch, A., and Levine, S.
Emergent complexity and zero-shot transfer via unsupervised environment design.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
URL https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html.

Depeweg, S., Hernandez-Lobato, J., Doshi-Velez, F., and Udluft, S.
Learning and policy search in stochastic dynamical systems with bayesian neural networks.
arXiv preprint, arXiv:1605.07127, 2016.
URL https://arxiv.org/abs/1605.07127.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L.
QLoRA: Efficient finetuning of quantized LLMs.
In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K.
Bert: Pre-training of deep bidirectional transformers for language understanding.
In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Doroudi, S., Aleven, V., and Brunskill, E.
Where's the reward? a review of reinforcement learning for instructional sequencing.
International Journal of Artificial Intelligence in Education, 29:568–620, 2019.

Dreyfus, S.
The numerical solution of variational problems.
Journal of Mathematical Analysis and Applications, 5(1):30–45, 1962.
ISSN 0022-247X.
doi: 10.1016/0022-247X(62)90004-5.

Du, Y., Watkins, O., Wang, Z., Colas, C., Darrell, T., Abbeel, P., Gupta, A., and Andreas, J.
Guiding pretraining in reinforcement learning with large language models.
In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8657–8677. PMLR, 23–29 Jul 2023.
URL https://proceedings.mlr.press/v202/du23f.html.

Duminy, N., Nguyen, S. M., Zhu, J., Duhaut, D., and Kerdreux, J.
Intrinsically motivated open-ended multi-task learning using transfer learning to discover task hierarchy.

Applied Sciences, 11(3):975, January 2021.
ISSN 2076-3417.
doi: 10.3390/app11030975.
URL http://dx.doi.org/10.3390/app11030975.

Edward, H., D, D. M., Jack, P.-H., Feryal, B., Aditi, M., Yuge, S., Tom, S., and Tim, R.
Position: Open-endedness is essential for artificial superhuman intelligence.
In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20597–20616. PMLR, 21–27 Jul 2024.
URL https://proceedings.mlr.press/v235/hughes24a.html.

Elfwing, S., Uchibe, E., and Doya, K.
Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017.
URL https://arxiv.org/abs/1702.03118.

Elkan, C. and Greiner, R.
Building large knowledge-based systems: Representation and inference in the cyc project: D.b. lenat and r.v. guha.
Artificial Intelligence, 61(1):41–52, 1993.
ISSN 0004-3702.
doi: https://doi.org/10.1016/0004-3702(93)90092-P.
URL https://www.sciencedirect.com/science/article/pii/000437029390092P.

Elliot, A. J. and Fryer, J. W.
The goal construct in psychology.
In *Handbook of Motivation Science*, pp. 235–250. Guilford Press, New York, 1 edition, 2008.

Ellis, B. D.
Novelty and curiosity as determinants of exploratory behaviour.
British Journal of Psychology. General Section, 41(1-2):68–80, 1950.
doi: https://doi.org/10.1111/j.2044-8295.1950.tb00262.x.
URL https://bpspsychub.onlinelibrary.wiley.com/doi/abs/10.1111/j.2044-8295.1950.tb00262.x.

Face, H.
Open r1: A fully open reproduction of deepseek-r1, January 2025.
URL https://github.com/huggingface/open-r1.

Faldor, M., Zhang, J., Cully, A., and Clune, J.
Omni-epic: Open-endedness via models of human notions of interestingness with environments programmed in code, 2025.
URL https://arxiv.org/abs/2405.15568.

Fan, C., Chen, J., Jin, Y., and He, H.
Can large language models serve as rational players in game theory? a systematic analysis.

In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024.
ISBN 978-1-57735-887-9.
doi: 10.1609/aaai.v38i16.29751.
URL https://doi.org/10.1609/aaai.v38i16.29751.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D., Zhu, Y., and Anandkumar, A.
Minedojo: Building open-ended embodied agents with internet-scale knowledge.
In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
URL http://papers.nips.cc/paper_files/paper/2022/hash/74a67268c5cc5910f64938cac4526a90-Abstract-Datasets_and_Benchmarks.html.

Fang, M., Deng, S., Zhang, Y., Shi, Z., Chen, L., Pechenizkiy, M., and Wang, J.
Large language models are neurosymbolic reasoners.
Proceedings of the AAAI Conference on Artificial Intelligence, 38(16):17985–17993, Mar. 2024.
doi: 10.1609/aaai.v38i16.29754.
URL https://ojs.aaai.org/index.php/AAAI/article/view/29754.

Fedzechkina, M., Jaeger, T. F., and Newport, E. L.
Language learners restructure their input to facilitate efficient communication.
Proceedings of the National Academy of Sciences, 109(44):17897–17902, 2012.
doi: 10.1073/pnas.1215776109.
URL https://www.pnas.org/doi/abs/10.1073/pnas.1215776109.

Felgenbaum, E. A.
The art of artificial intelligence: themes and case studies of knowledge engineering.
In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'77, pp. 1014–1029, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc.

Feng, J., Huang, S., Qu, X., Zhang, G., Qin, Y., Zhong, B., Jiang, C., Chi, J., and Zhong, W.
Retool: Reinforcement learning for strategic tool use in llms, 2025.
URL https://arxiv.org/abs/2504.11536.

Fernyhough, C.
The dialogic mind: A dialogic approach to the higher mental functions.
New Ideas in Psychology, 14(1):47–62, 1996.
ISSN 0732-118X.
doi: https://doi.org/10.1016/0732-118X(95)00024-B.
URL https://www.sciencedirect.com/science/article/pii/0732118X9500024B.

Fernyhough, C.
Alien voices and inner dialogue: towards a developmental account of auditory verbal hallucinations.
New Ideas in Psychology, 22(1):49–68, 2004.
ISSN 0732-118X.
doi: https://doi.org/10.1016/j.newideapsych.2004.09.001.
URL https://www.sciencedirect.com/science/article/pii/S0732118X04000054.

Ferrer-i Cancho, R., Hernández-Fernández, A., Lusseau, D., Agoramoorthy, G., Hsu, M., and Semple, S.
Compression as a universal principle of animal behavior.
Cognitive science, 37, 07 2013.
doi: 10.1111/cogs.12061.

Fitts, P. M. and Posner, M. I.
Human performance.
Brooks/Cole, 1967.

Florensa, C., Held, D., Geng, X., and Abbeel, P.
Automatic goal generation for reinforcement learning agents.
In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1514–1523. PMLR, 2018.
URL http://proceedings.mlr.press/v80/florensa18a.html.

Forestier, S. and Oudeyer, P.-Y.
Modular active curiosity-driven discovery of tool use.
In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016.

Forestier, S., Portelas, R., Mollard, Y., and Oudeyer, P.-Y.
Intrinsically motivated goal exploration processes with automatic curriculum learning.
J. Mach. Learn. Res., 23(1), January 2022.
ISSN 1532-4435.

Fournier, P., Sigaud, O., Chetouani, M., and Oudeyer, P.-Y.
Accuracy-based curriculum learning in deep reinforcement learning, 2018.

Frederick, J.
Statistical methods for speech recognition, 1999.

Gal, Y., McAllister, R., and Rasmussen, C. E.
Improving pilco with bayesian neural network dynamics models.
arXiv preprint, arXiv:1605.05966, 2016.
URL https://arxiv.org/abs/1605.05966.

Gao, C., Lan, X., Lu, Z., Mao, J., Piao, J., Wang, H., Jin, D., and Li, Y.
S^3: Social-network simulation system with large language model-empowered agents.
CoRR, abs/2307.14984, 2023.

doi: 10.48550/ARXIV.2307.14984.
URL https://doi.org/10.48550/arXiv.2307.14984.

Gaven, L., Romac, C., Carta, T., Lamprier, S., Sigaud, O., and Oudeyer, P.
SAC-GLAM: improving online RL for LLM agents with soft actor-critic and hindsight relabeling.
CoRR, abs/2410.12481, 2024.
doi: 10.48550/ARXIV.2410.12481.
URL https://doi.org/10.48550/arXiv.2410.12481.

Gaven, L., Carta, T., Romac, C., Colas, C., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y.
Magellan: Metacognitive predictions of learning progress guide autotelic llm agents in large goal spaces.
In *International Conference on Machine Learning (ICML)*, 2025.
URL https://arxiv.org/abs/2502.07709.

Gibson, J. J.
The Ecological Approach to Visual Perception.
Houghton Mifflin, Boston, 1979.

Goldberg, A. E.
The emergence of the semantics of argument structure constructions.
In MacWhinney, B. (ed.), *The Emergence of Language*, pp. 197–212. Psychology Press, Mahwah, NJ, 1999.

Gopnik, A.
The scientist as child.
63(4):485–514, 1996.
ISSN 0031-8248.
URL https://www.jstor.org/stable/188064.
Publisher: [Cambridge University Press, The University of Chicago Press, Philosophy of Science Association].

Gopnik, A.
Childhood as a solution to explore-exploit tensions.
Philosophical Transactions of the Royal Society B: Biological Sciences, 375(1803):20190502, July 2020.
doi: 10.1098/rstb.2019.0502.

Gopnik, A. and Meltzoff, A.
The development of categorization in the second year and its relation to other cognitive and linguistic developments.
In *Child development*, 1987.

Gopnik, A., Sobel, D. M., Schulz, L. E., and Glymour, C.
Causal learning mechanisms in very young children: Two-, three-, and four-year-olds infer causal relations from patterns of variation and covariation.
Developmental Psychology, 37(5):620–629, 2001.

Gottlieb, J. and Oudeyer, P.-Y.
Towards a neuroscience of active sampling and curiosity.
Nature Reviews Neuroscience, 19(12):758–770, December 2018.
URL https://inria.hal.science/hal-01965608.

Goyal, Niekum, and Mooney.
Using natural language for reward shaping in reinforcement learning.
In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Goyal, A., Lamb, A., Zhang, Y., Zhang, S., Courville, A. C., and Bengio, Y.
Professor forcing: A new algorithm for training recurrent networks.
In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 4601–4609, 2016.
URL https://proceedings.neurips.cc/paper/2016/hash/16026d60ff9b54410b3435b403afd226-Abstract.html.

Goyal, P., Niekum, S., and Mooney, R. J.
Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards.
In Kober, J., Ramos, F., and Tomlin, C. J. (eds.), *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pp. 485–497. PMLR, 2020.
URL https://proceedings.mlr.press/v155/goyal21a.html.

Gramopadhye, M. and Szafir, D.
Generating executable action plans with environmentally-aware language models.
In *IROS*, pp. 3568–3575, 2023.
doi: 10.1109/IROS55552.2023.10341989.
URL https://doi.org/10.1109/IROS55552.2023.10341989.

Guentulle, V., Nussbaum, M., Castillo, F., Chiuminatto, P., Spector, J. M., and Rojas, M.
The relationship between creativity and language as measured by linguistic maturity and text production.
Thinking Skills and Creativity, 54:101636, 2024.
ISSN 1871-1871.
doi: https://doi.org/10.1016/j.tsc.2024.101636.
URL https://www.sciencedirect.com/science/article/pii/S1871187124001743.

Guha, R. V. and Lenat, D. B.
Enabling agents to work together.
Commun. ACM, 37(7):126–142, 1994.
doi: 10.1145/176789.176804.
URL https://doi.org/10.1145/176789.176804.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H.,

Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z.
Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
URL https://arxiv.org/abs/2501.12948.

Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N. V., Wiest, O., and Zhang, X.
Large language model based multi-agents: A survey of progress and challenges.
In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pp. 8048–8057. ijcai.org, 2024.
URL https://www.ijcai.org/proceedings/2024/890.

Gupta, T., Karkus, P., Che, T., Xu, D., and Pavone, M.
Foundation models for semantic novelty in reinforcement learning.
ArXiv, abs/2211.04878, 2022.

Gómez, R. L. and Gerken, L.
Infant artificial language learning and language acquisition.
Trends in Cognitive Sciences, 4(5):178–186, 2000.
ISSN 1364-6613.
doi: https://doi.org/10.1016/S1364-6613(00)01467-4.
URL https://www.sciencedirect.com/science/article/pii/S1364661300014674.

Ha, D. and Schmidhuber, J.
World models.
CoRR, abs/1803.10122, 2018.
URL http://arxiv.org/abs/1803.10122.

Haber, N., Mrowca, D., Wang, S., Fei-Fei, L., and Yamins, D. L. K.
Learning to play with intrinsically-motivated, self-aware agents.
In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 8398–8409, 2018.

URL https://proceedings.neurips.cc/paper/2018/hash/71e63ef5b7249cfc60852f0e0f5bf4c8-Abstract.html.

Hafner, D.
Benchmarking the spectrum of agent capabilities, 2022.
URL https://arxiv.org/abs/2109.06780.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J.
Learning latent dynamics for planning from pixels.
In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 2555–2565. PMLR, 2019.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M.
Dream to Control: Learning Behaviors by Latent Imagination.
In *8th International Conference on Learning Representations (ICLR 2020)*. Curran Associates, Inc., 2020.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J.
Mastering Atari with Discrete World Models, 2021.
URL http://arxiv.org/abs/2010.02193.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T.
Mastering diverse control tasks through world models.
Nature, 640(8059):647–653, April 2025.
ISSN 1476-4687.
doi: 10.1038/s41586-025-08744-2.
URL https://www.nature.com/articles/s41586-025-08744-2.
Publisher: Nature Publishing Group.

Hamon, G.
transformerXL_PPO_JAX, July 2024.
URL https://inria.hal.science/hal-04659863.

Hansen, N., Wang, X., and Su, H.
Temporal difference learning for model predictive control.
arXiv preprint, arXiv:2203.04955, 2022.
URL https://arxiv.org/abs/2203.04955.

Hao, S., Gu, Y., Ma, H., Hong, J., Wang, Z., Wang, D., and Hu, Z.
Reasoning with language model is planning with world model.
In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, 2023. Association for Computational Linguistics.
doi: 10.18653/v1/2023.emnlp-main.507.

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

Harris, Z. S.
Distributional Structure.
In Harris, Z. S. and Hiż, H. (eds.), *Papers on Syntax*, Synthese Language Library, pp. 3–22. Springer Netherlands, Dordrecht, 1981.
ISBN 978-94-009-8467-7.
doi: 10.1007/978-94-009-8467-7_1.

Harrison, B., Ehsan, U., and Riedl, M. O.
Guiding reinforcement learning exploration using natural language.
In André, E., Koenig, S., Dastani, M., and Sukthankar, G. (eds.), *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pp. 1956–1958. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018.
URL http://dl.acm.org/citation.cfm?id=3238036.

Harvey, I.
The artificial evolution of behavior.
In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. The MIT Press, 02 1991.
ISBN 9780262256674.
doi: 10.7551/mitpress/3115.003.0054.
URL https://doi.org/10.7551/mitpress/3115.003.0054.

Hausknecht, M. and Stone, P.
Deep recurrent q-learning for partially observable mdps.
In *arXiv preprint arXiv:1507.06527*, 2015.

He, J., Ostendorf, M., He, X., Chen, J., Gao, J., Li, L., and Deng, L.
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.

He, K., Zhang, X., Ren, S., and Sun, J.
Deep residual learning for image recognition, 2015.
URL https://arxiv.org/abs/1512.03385.

Hebb, D.
The Organization of Behavior.
Wiley Book in Clinical Psychology, 1949.

Held, D., Geng, X., Florensa, C., and Abbeel, P.
Automatic goal generation for reinforcement learning agents.
In *International Conference on Machine Learning*, 2017.

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W. M., Jaderberg, M., and Teplyashin, D.
Grounded language learning in a simulated 3d world.
In *arXiv preprint arXiv:1706.06551*, 2017.

Hesse, M.
The cognitive claims of metaphor.
The Journal of Speculative Philosophy, 1988.

Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A.
Emergent systematic generalization in a situated agent.
In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020a.
URL https://openreview.net/forum?id=H1gNOeHKDS.

Hill, F., Mokra, S., Wong, N., and Harley, T.
Human Instruction-Following with Deep Reinforcement Learning via Transfer-Learning from Text.
arXiv:2005.09382 [cs], May 2020b.
arXiv: 2005.09382.

Hon, G. and Goldstein, B. R.
J. j. thomson's plum-pudding atomic model: The making of a scientific myth.
Annalen der Physik, 525(8-9):A129–A133, 2013.
doi: https://doi.org/10.1002/andp.201300732.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.201300732.

Hu, C., Fu, J., Du, C., Luo, S., Zhao, J., and Zhao, H.
Chatdb: Augmenting llms with databases as their symbolic memory.
CoRR, abs/2306.03901, 2023.
doi: 10.48550/ARXIV.2306.03901.
URL https://doi.org/10.48550/arXiv.2306.03901.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W.
LoRA: Low-rank adaptation of large language models.
In *International Conference on Learning Representations*, 2022.

Hu, H., Yarats, D., Gong, Q., Tian, Y., and Lewis, M.
Hierarchical decision making by generating and following natural language instructions.
In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
URL https://proceedings.neurips.cc/paper_files/paper/2019/file/7967cc8e3ab559e68cc944c44b1cf3e8-Paper.pdf.

Huang, L., Wang, W., Chen, J., and Wei, X.
Attention on attention for image captioning.
In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 4633–4642. IEEE, 2019.
doi: 10.1109/ICCV.2019.00473.
URL https://doi.org/10.1109/ICCV.2019.00473.

Huang, T.-H. K., Ferraro, F., Mostafazadeh, N., Misra, I., Agrawal, A., Devlin, J., Girshick, R., He, X., Kohli, P., Batra, D., Zitnick, C. L., Parikh, D., Vanderwende, L., Galley, M., and Mitchell, M.

Visual storytelling.
In Knight, K., Nenkova, A., and Rambow, O. (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1233–1239, San Diego, California, June 2016. Association for Computational Linguistics.
doi: 10.18653/v1/N16-1147.
URL https://aclanthology.org/N16-1147/.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I.
Language models as zero-shot planners: Extracting actionable knowledge for embodied agents.
In *International conference on machine learning*, pp. 9118–9147. PMLR, 2022.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Jackson, T., Brown, N., Luu, L., Levine, S., Hausman, K., and ichter, b.
Inner monologue: Embodied reasoning through planning with language models.
In Liu, K., Kulic, D., and Ichnowski, J. (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 14–18 Dec 2023.
URL https://proceedings.mlr.press/v205/huang23c.html.

Intelligence, P., Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Galliker, M. Y., Ghosh, D., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., LeBlanc, D., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Ren, A. Z., Shi, L. X., Smith, L., Springenberg, J. T., Stachowicz, K., Tanner, J., Vuong, Q., Walke, H., Walling, A., Wang, H., Yu, L., and Zhilinsky, U.
$\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025.
URL https://arxiv.org/abs/2504.16054.

Jain, U., Lazebnik, S., and Schwing, A. G.
Two can play this game: Visual dialog with discriminative question generation and answering.
In *Proc. CVPR*, 2018.

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

Jiang, M., Luketina, J., Nardelli, N., Minervini, P., Torr, P., Whiteson, S., and Rocktäschel, T.
Wordcraft: An environment for benchmarking commonsense agents.
In *Language in Reinforcement Learning Workshop at ICML 2020*, 2020.

Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J., Grefenstette, E., and Rocktäschel, T.
Replay-guided adversarial environment design.
In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, pp. 1884–1897, Red Hook, NY, USA, 2021a. Curran Associates Inc.
ISBN 978-1-71384-539-3.

Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J. N., Grefenstette, E., and Rocktäschel, T.
Replay-guided adversarial environment design.
In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 1884–1897, 2021b.
URL https://proceedings.neurips.cc/paper/2021/hash/0e915db6326b6fb6a3c56546980a8c93-Abstract.html.

Jiang, M., Grefenstette, E., and Rocktäschel, T.
Prioritized level replay.
In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950. PMLR, 2021c.
URL http://proceedings.mlr.press/v139/jiang21b.html.

Jiang, M., Rocktäschel, T., and Grefenstette, E.
General intelligence requires rethinking exploration.
Royal Society Open Science, 10(6):230539, 2023.

Jiang, Y., Gu, S. S., Murphy, K. P., and Finn, C.
Language as an abstraction for hierarchical deep reinforcement learning.
In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
URL https://proceedings.neurips.cc/paper_files/paper/2019/file/0af787945872196b42c9f73ead2565c8-Paper.pdf.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. J.
Vima: General robot manipulation with multimodal prompts.
ArXiv, abs/2210.03094, 2022.

Johnson, M., Hofmann, K., Hutton, T., and Bignell, D.
The malmo platform for artificial intelligence experimentation.
In *Ijcai*, volume 16, pp. 4246–4247, 2016.

Jonathan Ho, S. E.
Generative adversarial imitation learning.
In *arXiv preprint arXiv:1606.03476v1*, 2016.

Jordan, M. I. and Rumelhart, D. E.
Forward models: Supervised learning with a distal teacher.
Cognitive Science, 16(3):307–354, 1992.
ISSN 0364-0213.
doi: 10.1016/0364-0213(92)90036-T.

Kaelbling, L. P.
Learning to achieve goals.

In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1094–1099. Citeseer, 1993.

Kálmán, R. E.
Contributions to the theory of optimal control.
In *Matematica Mexicana*, 1960.
URL https://api.semanticscholar.org/CorpusID:845554.

Kanitscheider, I., Huizinga, J., Farhi, D., Guss, W. H., Houghton, B., Sampedro, R., Zhokhov, P., Baker, B., Ecoffet, A., Tang, J., Klimov, O., and Clune, J.
Multi-task curriculum learning in a complex, visual, hard-exploration domain: Minecraft, 2021.

Kaplan, F. and Oudeyer, P.-Y.
Maximizing learning progress: an internal reward system for development.
In *Embodied Artificial Intelligence: International Seminar, Dagstuhl Castle, Germany, July 7-11, 2003. Revised Papers*, pp. 259–270. Springer, 2003.

Kaplan, F. and Oudeyer, P.-Y.
In search of the neural circuits of intrinsic motivation.
Frontiers in neuroscience, 1:9, 2007.

Kaplan, J., McCandlish, S., Henighan, T. J., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D.
Scaling laws for neural language models.
ArXiv, abs/2001.08361, 2020.

Kaplan, R., Sauer, C., and Sosa, A.
Beating atari with natural language guided reinforcement learning.
CoRR, abs/1704.05539, 2017.
URL http://arxiv.org/abs/1704.05539.

Kasaei, S. H., Tomé, A. M., and Lopes, L. S.
Hierarchical object representation for open-ended object category learning and recognition.
In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
URL https://proceedings.neurips.cc/paper_files/paper/2016/file/299a23a2291e2126b91d54f3601ec162-Paper.pdf.

Kelley, H.
An on-line algorithm for dynamic reinforcement learning and planning in reactive environments.
Journal of the Franklin Institute, 30(10):947–954, 1960.

Kidd, C. and Hayden, B. Y.
The psychology and neuroscience of curiosity.
Neuron, 88(3):449–460, 2015.

Kim, K., Sano, M., Freitas, J. D., Haber, N., and Yamins, D.
Active world model learning with progress curiosity.

In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5306–5315. PMLR, 2020.
URL http://proceedings.mlr.press/v119/kim20e.html.

Kim, Y., Denton, C., Hoang, L., and Rush, A. M.
Structured attention networks.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
URL https://openreview.net/forum?id=HkE0Nvqlg.

Kingma, D. P. and Ba, J.
Adam: A method for stochastic optimization.
In *arXiv preprint arXiv:1412.6980v9*, 2014.

Kirby, S., Cornish, H., and Smith, K.
Cumulative cultural evolution in the laboratory: An experimental approach to the origins of structure in human language.
Proceedings of the National Academy of Sciences, 105(31):10681–10686, 2008.
doi: 10.1073/pnas.0707835105.
URL https://www.pnas.org/doi/abs/10.1073/pnas.0707835105.

Klissarov, M. and Precup, D.
Flexible option learning.
In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 4632–4646, 2021.
URL https://proceedings.neurips.cc/paper/2021/hash/24cceab7ffc1118f5daaace13c670885-Abstract.html.

Klissarov, M., Bagaria, A., Luo, Z., Konidaris, G. D., Precup, D., and Machado, M. C.
Discovering temporal structure: An overview of hierarchical reinforcement learning.
CoRR, abs/2506.14045, 2025.
doi: 10.48550/ARXIV.2506.14045.
URL https://doi.org/10.48550/arXiv.2506.14045.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y.
Large language models are zero-shot reasoners.
In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213. Curran Associates, Inc., 2022.
URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf.

Kombrink, S., Mikolov, T., Karafiát, M., and Burget, L.
Recurrent neural network based language modeling in meeting recognition.
In *Interspeech*, volume 11, pp. 2877–2880, 2011.

Konidaris, G. D. and Barto, A. G.
Building portable options: Skill transfer in reinforcement learning.
In Veloso, M. M. (ed.), *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pp. 895–900, 2007.
URL http://ijcai.org/Proceedings/07/Papers/144.pdf.

Kostrikov, I., Nair, A., and Levine, S.
Offline reinforcement learning with implicit q-learning.
In *arXiv preprint arXiv:2110.06169v1*, 2021.

Kovač, G., Laversanne-Finot, A., and Oudeyer, P.-Y.
Grimgep: Learning progress for robust goal sampling in visual deep reinforcement learning.
IEEE Transactions on Cognitive and Developmental Systems, 15(3):1396–1407, 2023.
doi: 10.1109/TCDS.2022.3216911.

Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., and Tenenbaum, J. B.
Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, 2016.
URL https://arxiv.org/abs/1604.06057.

Kumar, N., Silver, T., McClinton, W., Zhao, L., Proulx, S., Lozano-Pérez, T., Kaelbling, L. P., and Barry, J.
Practice makes perfect: Planning to learn skill parameter policies.
In *Robotics: Science and Systems (RSS)*, 2024.

Küttler, H., Nardelli, N., Miller, A. H., Raileanu, R., Selvatici, M., Grefenstette, E., and Rocktäschel, T.
The nethack learning environment.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
URL https://proceedings.neurips.cc/paper/2020/hash/569ff987c643b4bedf504efda8f786c2-Abstract.html.

Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D.
Reward design with language models.
In *The Eleventh International Conference on Learning Representations*, 2023.
URL https://openreview.net/forum?id=10uNUgI5Kl.

Lakoff, G. and Johnson, M.
Metaphors We Live By.
University of Chicago Press, 1981.

Laversanne-Finot, A., Pere, A., and Oudeyer, P.-Y.
Curiosity driven exploration of learned disentangled goal spaces.
In Billard, A., Dragan, A., Peters, J., and Morimoto, J. (eds.), *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 487–504. PMLR, 29–31 Oct 2018a.

Laversanne-Finot, A., Pere, A., and Oudeyer, P.-Y.
Curiosity driven exploration of learned disentangled goal spaces.
In *Conference on Robot Learning*, pp. 487–504. PMLR, 2018b.

Le, H. M., Jiang, N., Agarwal, A., Dudík, M., Yue, Y., and III, H. D.
Hierarchical imitation and reinforcement learning.
In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2923–2932. PMLR, 2018.
URL http://proceedings.mlr.press/v80/le18a.html.

Le Scao, T. and Rush, A.
How many data points is a prompt worth?
In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2627–2636, Online, June 2021. Association for Computational Linguistics.
doi: 10.18653/v1/2021.naacl-main.208.
URL https://aclanthology.org/2021.naacl-main.208.

LeCun, Y.
A path towards autonomous machine intelligence version 0.9.2.
OpenReview, 62(1):1–62, 2022.
https://openreview.net/pdf?id=BZ5a1r-kVsf.

LeCun, Y., Bengio, Y., and Hinton, G.
Deep learning.
Nature, 521:436–44, 05 2015.
doi: 10.1038/nature14539.

Lehmann, M.
The definitive guide to policy gradients in deep reinforcement learning: Theory, algorithms and implementations.
CoRR, abs/2401.13662, 2024.
doi: 10.48550/ARXIV.2401.13662.
URL https://doi.org/10.48550/arXiv.2401.13662.

Leiber, J. F.
Noam Chomsky: A Philosophic Overview.
G. K. Hall, Boston, 1975.

Leonard, J. A., Cordrey, S. R., Liu, H. Z., and Mackey, A. P.
Young children calibrate effort based on the trajectory of their performance.
Developmental Psychology, 59(3):609, 2023.

Levine, S.
Language models in plato's cave, Jun 2025.
URL https://sergeylevine.substack.com/p/language-models-in-platos-cave.

Levy, G., Colas, C., Oudeyer, P., Carta, T., and Romac, C.
Worldllm: Improving llms' world modeling using curiosity-driven theory-making.

CoRR, abs/2506.06725, 2025.
doi: 10.48550/ARXIV.2506.06725.
URL https://doi.org/10.48550/arXiv.2506.06725.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D.
Retrieval-augmented generation for knowledge-intensive NLP tasks.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Li, B., Lee-Urban, S., Johnston, G., and Riedl, M. O.
Story generation with crowdsourced plot graphs.
In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pp. 598–604. AAAI Press, 2013.

Li, G., Hammoud, H. A. A. K., Itani, H., Khizbullin, D., and Ghanem, B.
CAMEL: communicative agents for "mind" exploration of large scale language model society.
CoRR, abs/2303.17760, 2023a.
doi: 10.48550/ARXIV.2303.17760.
URL https://doi.org/10.48550/arXiv.2303.17760.

Li, H., Chong, Y. Q., Stepputtis, S., Campbell, J., Hughes, D., Lewis, C., and Sycara, K. P.
Theory of mind for multi-agent collaboration via large language models.
In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 180–192. Association for Computational Linguistics, 2023b.
doi: 10.18653/V1/2023.EMNLP-MAIN.13.
URL https://doi.org/10.18653/v1/2023.emnlp-main.13.

Li, H., Yang, X., Wang, Z., Zhu, X., Zhou, J., Qiao, Y., Wang, X., Li, H., Lu, L., and Dai, J.
Auto mc-reward: Automated dense reward design with large language models for minecraft.
In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 16426–16435. IEEE, 2024a.
doi: 10.1109/CVPR52733.2024.01554.
URL https://doi.org/10.1109/CVPR52733.2024.01554.

Li, M. Y., Fox, E. B., and Goodman, N. D.
Automated statistical model discovery with language models.
In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pp. 27791–27807, Vienna, Austria, July 2024b. JMLR.org.

Li, S., Puig, X., Du, Y., Wang, C. J., Akyürek, E., Torralba, A., Andreas, J., and Mordatch, I.
Pre-trained language models for interactive decision-making.
ArXiv, abs/2202.01771, 2022.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P. R., and Zeng, A.
Code as policies: Language model programs for embodied control.
ArXiv, abs/2209.07753, 2022.

Lică, M., Shirekar, O., Colle, B., and Raman, C.
Mindforge: Empowering embodied agents with theory of mind for lifelong cultural learning, 2025.
URL https://arxiv.org/abs/2411.12977.

Lidayan, A., Du, Y., Kosoy, E., Rufova, M., Abbeel, P., and Gopnik, A.
Intrinsically-motivated humans and agents in open-world exploration, 2025.
URL https://arxiv.org/abs/2503.23631.

Lin, B. Y., Fu, Y., Yang, K., Brahman, F., Huang, S., Bhagavatula, C., Ammanabrolu, P., Choi, Y., and Ren, X.
Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks.
In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 23813–23825. Curran Associates, Inc., 2023a.
URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4b0eea69deea512c9e2c469187643dc2-Paper-Conference.pdf.

Lin, J., Zhao, H., Zhang, A., Wu, Y., Ping, H., and Chen, Q.
Agentsims: An open-source sandbox for large language model evaluation.
CoRR, abs/2308.04026, 2023b.
doi: 10.48550/ARXIV.2308.04026.
URL https://doi.org/10.48550/arXiv.2308.04026.

Lin, J., Du, Y., Watkins, O., Hafner, D., Abbeel, P., Klein, D., and Dragan, A.
Learning to model the world with language.
In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Liu, G., Hu, E., Cheng, P., Lee, H., and Sun, S.
Hierarchical programmatic reinforcement learning via learning to compose programs.
In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21672–21697. PMLR, 2023.
URL https://proceedings.mlr.press/v202/liu23p.html.

Liu, H., Xue, W., Chen, Y., Chen, D., Zhao, X., Wang, K., Hou, L., Li, R., and Peng, W.
A survey on hallucination in large vision-language models.
arXiv preprint arXiv:2402.00253, 2024.

Liu, I.-J., Yuan, X., Côté, M.-A., Oudeyer, P.-Y., and Schwing, A. G.
Asking for knowledge: Training rl agents to query external knowledge using language.
In *International Conference on Machine Learning (ICML)*, 2022.

Lopes, M. and Oudeyer, P.-Y.
The strategic student approach for life-long exploration and learning.
In *2012 IEEE international conference on development and learning and epigenetic robotics (ICDL)*, pp. 1–8. IEEE, 2012a.

Lopes, M. and Oudeyer, P.-Y.
The strategic student approach for life-long exploration and learning.
In *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pp. 1–8, 2012b.
doi: 10.1109/DevLrn.2012.6400807.
ISSN: 2161-9476.

Lopes, M., Lang, T., Toussaint, M., and Oudeyer, P.-y.
Exploration in model-based reinforcement learning by empirically estimating learning progress.
In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
URL https://proceedings.neurips.cc/paper/2012/hash/a0a080f42e6f13b3a2df133f073095dd-Abstract.html.

Lovelace, A. and Menabrea, L.
Sketch of the analytical engine invented by charles babbage esq.
Scientific Memoirs, pp. 694, 1843.

Lu, J., Clark, C., Zellers, R., Mottaghi, R., and Kembhavi, A.
UNIFIED-IO: A Unified Model for Vision, Language, and Multi-modal Tasks.
February 2023.
URL https://openreview.net/forum?id=E01k9048soZ.

Lu, Y., Hausman, K., Chebotar, Y., Yan, M., Jang, E., Herzog, A., Xiao, T., Irpan, A., Khansari, M., Kalashnikov, D., and Levine, S.
Aw-opt: Learning robotic skills with imitation and reinforcement at scale.
In *Conference on Robot Learning*, 2021.

Lucas, C. G., Bridgers, S., Griffiths, T. L., and Gopnik, A.
When children are better (or at least more open-minded) learners than adults: Developmental differences in learning the forms of causal relationships.
Cognition, 131:284–299, 2014.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T.
A survey of reinforcement learning informed by natural language.
arXiv preprint arXiv:1906.03926, 2019a.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T.
A survey of reinforcement learning informed by natural language, 2019b.
URL https://arxiv.org/abs/1906.03926.

Lupyan, G.
Carving nature at its joints and carving joints into nature: How labels augment category representations.
In *Modeling Language, Cognition and Action*. World Scientific, 2005.

Lynch, C. and Sermanet, P.
Grounding language in play.
In *arXiv preprint arXiv:2005.07648*, 2020.

Lynch, C. and Sermanet, P.
Language conditioned imitation learning over unstructured data.
In *Robotics: Science and Systems XVII*, 2021.

Maaten, L. v. d. and Hinton, G.
Visualizing Data using t-SNE.
Journal of Machine Learning Research, 9(86):2579–2605, 2008.
ISSN 1533-7928.
URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Machado, M. C., Bellemare, M. G., and Bowling, M. H.
A laplacian framework for option discovery in reinforcement learning.
In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2295–2304. PMLR, 2017.
URL http://proceedings.mlr.press/v70/machado17a.html.

Mahowald, K., Ivanova, A. A., Blank, I. A., Kanwisher, N. G., Tenenbaum, J. B., and Fedorenko, E.
Dissociating language and thought in large language models: a cognitive perspective.
ArXiv, abs/2301.06627, 2023.

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., and Wu, J.
The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision.
CoRR, abs/1904.12584, 2019.

Markov, A. A.
Essai d'une recherche statistique sur le texte du roman "Eugene Onegin" illustrant la liaison des epreuve en chain ('Example of a statistical investigation of the text of "Eugene Onegin" illustrating the dependence between samples in chain').
Izvistia Imperatorskoi Akademii Nauk (Bulletin de l'Académie Impériale des Sciences de St.-Pétersbourg), 7:153–162, 1913.
English translation by Morris Halle, 1956.

Martinet, A.
Éléments de linguistique générale.
Colin, Paris, 1960.

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J.
Teacher–student curriculum learning, 2020.

Matthews, M., Beukman, M., Ellis, B., Samvelyan, M., Jackson, M., Coward, S., and Foerster, J.
Craftax: A Lightning-Fast Benchmark for Open-Ended Reinforcement Learning, February 2024a.
URL http://arxiv.org/abs/2402.16801.
arXiv:2402.16801 [cs].

Matthews, M., Beukman, M., Lu, C., and Foerster, J.
Kinetix: Investigating the training of general agents through open-ended physics-based control tasks, 2024b.

McCulloch, W. S. and Pitts, W.
A logical calculus of the ideas immanent in nervous activity.
The Bulletin of Mathematical Biophysics, 5(4):115–133, 1943.
doi: 10.1007/bf02478259.

Mei, H., Bansal, M., and Walter, M. R.
Listen, attend, and walk: Neural mapping of navigational instructions to action sequences.
In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2016a.

Mei, H., Bansal, M., and Walter, M. R.
Listen, attend, and walk: Neural mapping of navigational instructions to action sequences.
In Schuurmans, D. and Wellman, M. P. (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 2772–2778. AAAI Press, 2016b.
doi: 10.1609/AAAI.V30I1.10364.
URL https://doi.org/10.1609/aaai.v30i1.10364.

Melnikoff, D. E., Carlson, R. W., and Stillman, P. E.
A computational theory of the subjective experience of flow.
Nature Communications, 13(1):2252, 2022.
ISSN 2041-1723.
doi: 10.1038/s41467-022-29742-2.
URL https://doi.org/10.1038/s41467-022-29742-2.

Meyer, J.-A. and Wilson, S. W.
A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers, pp. 222–227.
1991.

Mikolov, T., Chen, K., Corrado, G., and Dean, J.
Efficient estimation of word representations in vector space.
In Bengio, Y. and LeCun, Y. (eds.), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.
URL http://arxiv.org/abs/1301.3781.

Minervini, P., Bosnjak, M., Rocktäschel, T., and Riedel, S.
Towards neural theorem proving at scale.
In *NAMPI 2018*, 2018.

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

Mirolli, M. and Baldassarre, G.
Functions and mechanisms of intrinsic motivations.
In Baldassarre, G. and Mirolli, M. (eds.), *Intrinsically Motivated Learning in Natural and Artificial Systems*, pp. 49–72. Springer, 2013.
doi: 10.1007/978-3-642-32375-1_3.
URL https://doi.org/10.1007/978-3-642-32375-1_3.

Misra, D., Langford, J., and Artzi, Y.
Mapping instructions and visual observations to actions with reinforcement learning.
In *arXiv preprint arXiv:1704.08795v2*, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al.
Human-level control through deep reinforcement learning.
nature, 518(7540):529–533, 2015a.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al.
Human-level control through deep reinforcement learning.
Nature, 518(7540):529–533, 2015b.
doi: 10.1038/nature14236.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K.
Asynchronous methods for deep reinforcement learning.
In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Moerland, T. M., Broekens, J., Plaat, A., and Jonker, C. M.
Model-based Reinforcement Learning: A Survey.
Foundations and Trends® in Machine Learning, 16(1):1–118, January 2023.
ISSN 1935-8237, 1935-8245.
doi: 10.1561/2200000086.
URL https://www.nowpublishers.com/article/Details/MAL-086.
Publisher: Now Publishers, Inc.

Moon, S., Yeom, J., Park, B., and Song, H. O.
Discovering hierarchical achievements in reinforcement learning via contrastive learning.
In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 63674–63686. Curran Associates, Inc., 2023.
URL https://proceedings.neurips.cc/paper_files/paper/2023/file/c919a2b5ec1de69f2629f9119676e336-Paper-Conference.pdf.

Moore, R. and Palazzolo, G.
Animal Communication.
In Zalta, E. N. and Nodelman, U. (eds.), *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2024 edition, 2024.

Moulin-Frier, C. and Oudeyer, P.
Exploration strategies in developmental robotics: A unified probabilistic framework.
In *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics, ICDL-EPIROB 2013, Osaka, Japan, August 18-22, 2013*, pp. 1–6. IEEE, 2013.
doi: 10.1109/DEVLRN.2013.6652535.
URL https://doi.org/10.1109/DevLrn.2013.6652535.

Moulin-Frier, C. and Oudeyer, P.-Y.
Exploration strategies in developmental robotics: A unified probabilistic framework.
In *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pp. 1–6, 2013.
doi: 10.1109/DevLrn.2013.6652535.
ISSN: 2161-9476.

Moulin-Frier, C., Nguyen, S. M., and Oudeyer, P.-Y.
Self-organization of early vocal development in infants and machines: the role of intrinsic motivation.
Frontiers in Psychology, 4, 2014.
ISSN 1664-1078.
doi: 10.3389/fpsyg.2013.01006.
Publisher: Frontiers.

Mu, J., Zhong, V., Raileanu, R., Jiang, M., Goodman, N., Rocktäschel, T., and Grefenstette, E.
Improving intrinsic exploration with language abstractions.
In *arXiv preprint arXiv:2202.08938v1*, 2022.

Müller, T. and Raviv, L.
Communication games: Social interaction in the formation of novel communication systems, 04 2024.

Nachum, O., Gu, S., Lee, H., and Levine, S.
Data-efficient hierarchical reinforcement learning.
In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 3307–3317, 2018a.
URL https://proceedings.neurips.cc/paper/2018/hash/e6384711491713d29bc63fc5eeb5ba4f-Abstract.html.

Nachum, O., Gu, S., Lee, H., and Levine, S.
Near-optimal representation learning for hierarchical reinforcement learning.
CoRR, abs/1810.01257, 2018b.
URL http://dblp.uni-trier.de/db/journals/corr/corr1810.html#abs-1810-01257.

Nachum, O., Gu, S. S., Lee, H., and Levine, S.
Data-efficient hierarchical reinforcement learning.

In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018c.
URL https://proceedings.neurips.cc/paper_files/paper/2018/file/e6384711491713d29bc63fc5eeb5ba4f-Paper.pdf.

Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S.
Visual reinforcement learning with imagined goals.
In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pp. 9209–9220, Montréal, Canada, December 2018. Curran Associates, Inc.

Newell, A. and Simon, H. A.
The logic theory machine. a complex information processing system.
Journal of Symbolic Logic, 22(3):331–332, 1957.
doi: 10.2307/2963663.

Newell, A. and Simon, H. A.
Computer science as empirical inquiry: symbols and search.
Commun. ACM, 19(3):113–126, March 1976.
ISSN 0001-0782.
doi: 10.1145/360018.360022.
URL https://doi.org/10.1145/360018.360022.

Ng, A. Y., Harada, D., and Russell, S.
Policy invariance under reward transformations: Theory and application to reward shaping.
In *International Conference on Machine Learning (ICML), volume 99, pages 278–287*, 1999.

Nguyen, D. and Widrow, B.
The truck backer-upper: An example of self-learning in neural networks.
In Miller, W. T., Sutton, R. S., and Werbos, P. J. (eds.), *Neural Networks for Control*, pp. 287–299. MIT Press, Cambridge, MA, USA, 1990.
ISBN 0-262-13261-3.

Oh, J., Singh, S., Lee, H., and Kohli, P.
Zero-shot task generalization with multi-task deep reinforcement learning.
In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2661–2670. PMLR, 2017.
URL http://proceedings.mlr.press/v70/oh17a.html.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L.
Solving rubik's cube with a robot hand.
CoRR, abs/1910.07113, 2019.
URL http://arxiv.org/abs/1910.07113.

Oudeyer, P.-Y. and Kaplan, F.
What is intrinsic motivation? a typology of computational approaches.
Frontiers in neurorobotics, 1:108, 2007a.

Oudeyer, P.-Y. and Kaplan, F.
What is intrinsic motivation? a typology of computational approaches.
Frontiers in Neurorobotics, 1:6, 2007b.
doi: 10.3389/neuro.12.006.2007.

Oudeyer, P.-Y. and Kaplan, F.
What is intrinsic motivation? a typology of computational approaches.
1, 2007c.
ISSN 1662-5218.
doi: 10.3389/neuro.12.006.2007.
URL https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/neuro.12.006.2007/full.
Publisher: Frontiers.

Oudeyer, P.-Y. and Smith, L. B.
How evolution may work through curiosity-driven developmental process.
Topics in Cognitive Science, 8(2):492–502, 2016.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V.
Intrinsic motivation systems for autonomous mental development.
IEEE Transactions on Evolutionary Computation, 11(2):265–286, 2007.
ISSN 1941-0026.
doi: 10.1109/TEVC.2006.890271.
Conference Name: IEEE Transactions on Evolutionary Computation.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R.
Training language models to follow instructions with human feedback.
In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022a.
URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L. E., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. J.
Training language models to follow instructions with human feedback.
ArXiv, abs/2203.02155, 2022b.

Pan, Y., Yao, T., Li, Y., and Mei, T.
X-linear attention networks for image captioning.

In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 10968–10977. Computer Vision Foundation / IEEE, 2020.
doi: 10.1109/CVPR42600.2020.01098.
URL https://openaccess.thecvf.com/content_CVPR_2020/html/Pan_X-Linear_Attention_Networks_for_Image_Captioning_CVPR_2020_paper.html.

Parisotto, E., Mohamed, A., Singh, R., Li, L., Zhou, D., and Kohli, P.
Neuro-symbolic program synthesis.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
URL https://openreview.net/forum?id=rJ0JwFcex.

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S.
Generative agents: Interactive simulacra of human behavior.
In Follmer, S., Han, J., Steimle, J., and Riche, N. H. (eds.), *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pp. 2:1–2:22. ACM, 2023.
doi: 10.1145/3586183.3606763.
URL https://doi.org/10.1145/3586183.3606763.

Pashevich, A., Schmid, C., and Sun, C.
Episodic transformer for vision-and-language navigation.
In *arXiv preprint arXiv:2105.06453v2*, 2021.

Patel, R. and Pavlick, E.-J.
Mapping language models to grounded conceptual spaces.
In *International Conference on Learning Representations*, 2022.

Patel, R., Pavlick, E., and Tellex, S.
Grounding Language to Non-Markovian Tasks with No Supervision of Task Specifications.
Robotics: Science and Systems XVI, July 2020.
doi: 10.15607/RSS.2020.XVI.016.
URL http://www.roboticsproceedings.org/rss16/p016.pdf.
Conference Name: Robotics: Science and Systems 2020 ISBN: 9780992374761 Publisher: Robotics: Science and Systems Foundation.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T.
Curiosity-driven exploration by self-supervised prediction.
In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2778–2787. PMLR, 2017a.
ISSN: 2640-3498.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T.
Curiosity-driven exploration by self-supervised prediction.
In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2778–2787. PMLR, 06–11 Aug 2017b.
URL https://proceedings.mlr.press/v70/pathak17a.html.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S.
Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019.
URL https://arxiv.org/abs/1910.00177.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A.
Film: Visual reasoning with a general conditioning layer.
In *Adaptive Computation and Machine Learning*, 2017a.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C.
Film: Visual reasoning with a general conditioning layer.
CoRR, abs/1709.07871, 2017b.
URL http://arxiv.org/abs/1709.07871.

Piaget, J.
Children's philosophies.
In Murchison, C. (ed.), *A Handbook of Child Psychology*, pp. 377–391. Clark University Press, 1931.

Piaget, J.
The construction of reality in the child.
The construction of reality in the child. Basic Books/Hachette Book Group, New York, NY, US, 1954.
doi: 10.1037/11168-000.
Pages: xiii, 386.

Pignatelli, E., Ferret, J., Rockäschel, T., Grefenstette, E., Paglieri, D., Coward, S., and Toni, L.
Assessing the zero-shot capabilities of llms for action evaluation in rl, 2024.
URL https://arxiv.org/abs/2409.12798.

Piriyakulkij, W. T., Langenfeld, C., Le, T. A., and Ellis, K.
Doing Experiments and Revising Rules with Natural Language and Probabilistic Reasoning.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 53102–53137. Curran Associates, Inc., 2024.
URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5f1b350fc0c2affd56f465faa36be343-Paper-Conference.pdf.

Pitis, S., Chan, H., Zhao, S., Stadie, B., and Ba, J.
Maximum Entropy Gain Exploration for Long Horizon Multi-goal Reinforcement Learning.
In *Proceedings of the 37th International Conference on Machine Learning*, pp. 7750–7761. PMLR, November 2020.
ISSN: 2640-3498.

Poesia, G., Broman, D., Haber, N., and Goodman, N. D.
Learning formal mathematics from intrinsic motivation.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 43032–43057. Curran Associates, Inc., 2024.

URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4b8001fc75f0532827472ea5a16af9ca-Paper-Conference.pdf.

Poli, F., Meyer, M., Mars, R. B., and Hunnius, S.
Contributions of expected learning progress and perceptual novelty to curiosity-driven exploration.
Cognition, 225:105119, 2022.
ISSN 0010-0277.
doi: https://doi.org/10.1016/j.cognition.2022.105119.
URL https://www.sciencedirect.com/science/article/pii/S001002772200107X.

Poli, F., Meyer, M., Mars, R. B., and Hunnius, S.
Exploration in 4-year-old children is guided by learning progress and novelty.
Child Development, 2024a.

Poli, F., O'Reilly, J. X., Mars, R. B., and Hunnius, S.
Curiosity and the dynamics of optimal exploration.
Trends in Cognitive Sciences, 28(5):441–453, 2024b.

Pong, V., Dalal, M., Lin, S., Nair, A., Bahl, S., and Levine, S.
Skew-fit: State-covering self-supervised reinforcement learning.
In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7783–7792. PMLR, 2020.
URL http://proceedings.mlr.press/v119/pong20a.html.

Pong, V. H., Dalal, M., Lin, S., Nair, A., Bahl, S., and Levine, S.
Skew-fit: State-covering self-supervised reinforcement learning.
In *International Conference on Machine Learning*, 2019.

Portelas, R., Colas, C., Hofmann, K., and Oudeyer, P.-Y.
Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments.
In *Conference on Robot Learning*, pp. 835–853. PMLR, 2020a.

Portelas, R., Colas, C., Hofmann, K., and Oudeyer, P.-Y.
Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments.
In *Conference on Robot Learning*, pp. 835–853. PMLR, 2020b.

Portelas, R., Colas, C., Hofmann, K., and Oudeyer, P.-Y.
Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments.
In *Conference on Robot Learning*, pp. 835–853. PMLR, 2020c.

Portelas, R., Colas, C., Weng, L., Hofmann, K., and Oudeyer, P.
Automatic curriculum learning for deep RL: A short survey.
In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pp. 4819–4825. ijcai.org, 2020d.
doi: 10.24963/IJCAI.2020/671.
URL https://doi.org/10.24963/ijcai.2020/671.

Portelas, R., Colas, C., Weng, L., Hofmann, K., and Oudeyer, P.-Y.
Automatic curriculum learning for deep rl: A short survey.
arXiv preprint arXiv:2003.04664, 2020e.

Portelas, R., Colas, C., Weng, L., Hofmann, K., and Oudeyer, P.-Y.
Automatic curriculum learning for deep rl: A short survey.
In *International Joint Conference on Artificial Intelligence*, 2020f.

Pourcel, G., Carta, T., Kovač, G., and Oudeyer, P.-Y.
Autotelic LLM-based exploration for goal-conditioned RL.
In *IMOL@NeurIPS 2024 - Intrinsically Motivated Open-ended Learning Workshop at NeurIPS 2024*, Vancouver, Canada, December 2024a.
URL https://inria.hal.science/hal-04861896.

Pourcel, J., Colas, C., Molinaro, G., Oudeyer, P.-Y., and Teodorescu, L.
ACES: Generating a diversity of challenging programming puzzles with autotelic generative models.
In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b.

Precup, D. and Sutton, R. S.
Temporal abstraction in reinforcement learning.
PhD thesis, University of Massachusetts, 2000.
AAI9978540.

Pushkin, A.
Eugene Onegin.
Dedalus Ebooks, 2010.
ISBN 9781907650109.
URL https://books.google.fr/books?id=mawDH06yv3kC.

Puterman, M. L.
Markov decision processes: discrete stochastic dynamic programming.
John Wiley & Sons, 2014.

Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., Xu, J., Li, D., Liu, Z., and Sun, M.
Chatdev: Communicative agents for software development.
In Ku, L., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 15174–15186. Association for Computational Linguistics, 2024.
doi: 10.18653/V1/2024.ACL-LONG.810.
URL https://doi.org/10.18653/v1/2024.acl-long.810.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., and Sun, M.
Toolllm: Facilitating large language models to master 16000+ real-world apis.
In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
URL https://openreview.net/forum?id=dHng2O0Jjr.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z.
Qwen2.5 technical report, 2025.
URL https://arxiv.org/abs/2412.15115.

Racaniere, S., Lampinen, A., Santoro, A., Reichert, D., Firoiu, V., and Lillicrap, T.
Automated curriculum generation through setter-solver interactions.
In *International Conference on Learning Representations*, 2020.

Radford, A., Jozefowicz, R., and Sutskever, I.
Learning to generate reviews and discovering sentiment.
arXiv preprint arXiv:1704.01444, 2017.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.
Improving language understanding by generative pre-training.
2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.
Language models are unsupervised multitask learners.
OpenAI blog, 1(8):9, 2019.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I.
Learning transferable visual models from natural language supervision.
In *arXiv preprint arXiv:2103.00020v1*, 2021.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., Rutherford, E., Hennigan, T., Menick, J., Cassirer, A., Powell, R., van den Driessche, G., Hendricks, L. A., Rauh, M., Huang, P.-S., Glaese, A., Welbl, J., Dathathri, S., Huang, S., Uesato, J., Mellor, J. F. J., Higgins, I., Creswell, A., McAleese, N., Wu, A., Elsen, E., Jayakumar, S. M., Buchatskaya, E., Budden, D., Sutherland, E., Simonyan, K., Paganini, M., Sifre, L., Martens, L., Li, X. L., Kuncoro, A., Nematzadeh, A., Gribovskaya, E., Donato, D., Lazaridou, A., Mensch, A., Lespiau, J.-B., Tsimpoukelli, M., Grigorev, N. K., Fritz, D., Sottiaux, T., Pajarskas, M., Pohlen, T., Gong, Z., Toyama, D., de Masson d'Autume, C., Li, Y., Terzi, T., Mikulik, V., Babuschkin, I., Clark, A., de Las Casas, D., Guy, A., Jones, C., Bradbury, J., Johnson, M. G., Hechtman, B. A., Weidinger, L., Gabriel, I., Isaac, W. S., Lockhart, E., Osindero, S., Rimell, L., Dyer, C., Vinyals, O., Ayoub, K. W., Stanway, J., Bennett, L. L., Hassabis, D., Kavukcuoglu, K., and Irving, G.
Scaling language models: Methods, analysis & insights from training gopher.
ArXiv, abs/2112.11446, 2021.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C.
Direct preference optimization: Your language model is secretly a reward model.
In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural*

Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023.
URL http://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J.
Exploring the limits of transfer learning with a unified text-to-text transformer.
Journal of Machine Learning Research, 21(140):1–67, 2020.
ISSN 1533-7928.

Raileanu, R. and Rocktäschel, T.
RIDE: rewarding impact-driven exploration for procedurally-generated environments.
In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
URL https://openreview.net/forum?id=rkg-TJBFPB.

Raileanu, R. and Rocktäschel, T.
Ride: Rewarding impact-driven exploration for procedurally-generated environments.
In *International Conference on Learning Representations (ICLR)*, 2020.

Ramamurthy, R., Ammanabrolu, P., Brantley, K., Hessel, J., Sifa, R., Bauckhage, C., Hajishirzi, H., and Choi, Y.
Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization.
ArXiv, abs/2210.01241, 2022.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I.
Zero-shot text-to-image generation.
ArXiv, abs/2102.12092, 2021.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W.
Sequence level training with recurrent neural networks.
In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.
URL http://arxiv.org/abs/1511.06732.

Rasmussen, C. E. and Williams, C. K. I.
Gaussian processes for machine learning.
In *Adaptive Computation and Machine Learning*, 2005.

Raviv, L., Meyer, A., and Lev-Ari, S.
Larger communities create more systematic languages.
Proceedings of the Royal Society B: Biological Sciences, 286:20191262, 07 2019.
doi: 10.1098/rspb.2019.1262.

Reader, A.
Steps towards genuine artificial intelligence.

Acta Psychologica, 29:279–289, 1969.
ISSN 0001-6918.
doi: https://doi.org/10.1016/0001-6918(69)90021-3.
URL https://www.sciencedirect.com/science/article/pii/0001691869900213.

Rebuffel, C., Scialom, T., Soulier, L., Piwowarski, B., Lamprier, S., Staiano, J., Scoutheeten, G., and Gallinari, P.
Data-QuestEval: A referenceless metric for data-to-text semantic evaluation.
In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 8029–8036, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.
doi: 10.18653/v1/2021.emnlp-main.633.
URL https://aclanthology.org/2021.emnlp-main.633.

Reid, M., Yamada, Y., and Gu, S. S.
Can wikipedia help offline reinforcement learning?
ArXiv, abs/2201.12122, 2022.

Riesenhuber, M. and Poggio, T.
Hierarchical models of object recognition in cortex.
Nat Neurosci, 2(11):1019–1025, 1999.
doi: 10.1038/14819.
Place: UNITED STATES ISBN: 1097-6256.

Romac, C., Portelas, R., Hofmann, K., and Oudeyer, P.-Y.
TeachMyAgent: a benchmark for automatic curriculum learning in deep RL.
In *International Conference on Machine Learning*, pp. 9052–9063. PMLR, 2021.
ISSN: 2640-3498.

Romac, C., Carta, T., and Oudeyer, P.-Y.
Les algorithmes des IA peuvent-ils comprendre notre monde ?
Pour la science, February 2024.
URL https://inria.hal.science/hal-04855190.

Rosenberg, N.
The unbound prometheus: Technological change and industrial development in western europe from 1750 to the present. by david s. landes. cambridge, england: Cambridge university press, 1969. pp. 566. 8.50*cloth*,2.95 paper.
The Journal of Economic History, 31:497–500, 06 2010.
doi: 10.1017/S0022050700091269.

Rosenblatt, F.
The Perceptron, a Perceiving and Recognizing Automaton: (Project Para).
Report / Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957.
URL https://books.google.fr/books?id=P_XGPgAACAAJ.

Rosenfeld, R.
Two decades of statistical language modeling: Where do we go from here?
Proceedings of the IEEE, 88(8):1270–1278, 2002.

Ruggeri, A., Stanciu, O., Pelz, M., Gopnik, A., and Schulz, E.
Preschoolers search longer when there is more information to be gained.
Developmental Science, 27(1):e13411, January 2024.
doi: 10.1111/desc.13411.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J.
Learning representations by back-propagating errors.
Nature, 323:533–536, 1986.
URL https://api.semanticscholar.org/CorpusID:205001834.

Russell, S. J. and Wefald, E.
Do the right thing - studies in limited rationality.
MIT Press, 1991.
ISBN 978-0-262-18144-0.

Saegusa, R., Metta, G., Sandini, G., and Sakka, S.
Active motor babbling for sensorimotor learning.
In *IEEE International Conference on Robotics and Biomimetics, ROBIO 2008, February 22-25, 2009, Bangkok, Thailand*, pp. 794–799. IEEE, 2008.
doi: 10.1109/ROBIO.2009.4913101.
URL https://doi.org/10.1109/ROBIO.2009.4913101.

Salge, C., Glackin, C., and Polani, D.
Empowerment - an introduction.
CoRR, abs/1310.1863, 2013.
URL http://arxiv.org/abs/1310.1863.

Samuel, A. L.
Some studies in machine learning using the game of checkers.
IBM J. Res. Dev., 3(3):210–229, 1959.
doi: 10.1147/RD.33.0210.
URL https://doi.org/10.1147/rd.33.0210.

Samvelyan, M., Kirk, R., Kurin, V., Parker-Holder, J., Jiang, M., Hambro, E., Petroni, F., Kuttler, H., Grefenstette, E., and Rocktäschel, T.
Minihack the planet: A sandbox for open-ended reinforcement learning research.
In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.
URL https://openreview.net/forum?id=skFwlyefkWJ.

Santhanam, S., Cheng, Z., Mather, B., Dorr, B. J., Bhatia, A., Hebenstreit, B., Zemel, A., Dalton, A., Strzalkowski, T., and Shaikh, S.
Learning to plan and realize separately for open-ended dialogue systems.
CoRR, abs/2009.12506, 2020.
URL https://arxiv.org/abs/2009.12506.

Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P. W., and Lillicrap, T.
A simple neural network module for relational reasoning.

In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4967–4976, 2017.
URL https://proceedings.neurips.cc/paper/2017/hash/e6acf4b0f69f6f6e60e9a815938aa1ff-Abstract.html.

Santucci, V. G., Baldassarre, G., and Mirolli, M.
Which is the best intrinsic motivation signal for learning multiple skills?
Frontiers Neurorobotics, 7:22, 2013.
doi: 10.3389/FNBOT.2013.00022.
URL https://doi.org/10.3389/fnbot.2013.00022.

Sayalı, C., Heling, E., and Cools, R.
Learning progress mediates the link between cognitive effort and task engagement.
Cognition, 236:105418, 2023.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E.-J., Ili'c, S., Hesslow, D., Castagn'e, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S. R., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P. O., Sanh, V., Laurenccon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Etxabe, A. S., Aji, A. F., Alfassy, A., Rogers, A., Nitzav, A. K., Xu, C., Mou, C., Emezue, C. C., Klamm, C., Leong, C., van Strien, D. A., Adelani, D. I., Radev, D. R., Ponferrada, E. G., Levkovizh, E., Kim, E., Natan, E. B., Toni, F. D., Dupont, G., Kruszewski, G., Pistilli, G., ElSahar, H., Benyamina, H., Tran, H., Yu, I., Abdulmumin, I., Johnson, I., Gonzalez-Dios, I., de la Rosa, J., Chim, J., Dodge, J., Zhu, J., Chang, J., Frohberg, J., Tobing, J. L., Bhattacharjee, J., Almubarak, K., Chen, K., Lo, K., von Werra, L., Weber, L., Phan, L., Allal, L. B., Tanguy, L., Dey, M., Muñoz, M. R., Masoud, M., Grandury, M., vSavsko, M., Huang, M., Coavoux, M., Singh, M., Jiang, M. T.-J., Vu, M. C., Jauhar, M. A., Ghaleb, M., Subramani, N., Kassner, N., Khamis, N., Nguyen, O., Espejel, O., de Gibert, O., Villegas, P., Henderson, P., Colombo, P., Amuok, P., Lhoest, Q., Harliman, R., Bommasani, R., L'opez, R., Ribeiro, R., Osei, S., Pyysalo, S., Nagel, S., Bose, S., Muhammad, S. H., Sharma, S., Longpre, S., Nikpoor, S., Silberberg, S., Pai, S., Zink, S., Torrent, T. T., Schick, T., Thrush, T., Danchev, V., Nikoulina, V., Laippala, V., Lepercq, V., Prabhu, V., Alyafeai, Z., Talat, Z., Raja, A., Heinzerling, B., Si, C., Salesky, E., Mielke, S. J., Lee, W. Y., Sharma, A., Santilli, A., Chaffin, A., Stiegler, A., Datta, D., Szczechla, E., Chhablani, G., Wang, H., Pandey, H., Strobelt, H., Fries, J. A., Rozen, J., Gao, L., Sutawika, L., Bari, M. S., Al-shaibani, M. S., Manica, M., Nayak, N. V., Teehan, R., Albanie, S., Shen, S., Ben-David, S., Bach, S. H., Kim, T., Bers, T., Févry, T., Neeraj, T., Thakker, U., Raunak, V., Tang, X., Yong, Z. X., Sun, Z., Brody, S., Uri, Y., Tojarieh, H., Roberts, A., Chung, H. W., Tae, J., Phang, J., Press, O., Li, C., Narayanan, D., Bourfoune, H., Casper, J., Rasley, J., Ryabinin, M., Mishra, M., Zhang, M., Shoeybi, M., Peyrounette, M., Patry, N., Tazi, N., Sanseviero, O., von Platen, P., Cornette, P., Lavall'ee, P. F., Lacroix, R., Rajbhandari, S., Gandhi, S., Smith, S., Requena, S., Patil, S., Dettmers, T., Baruwa, A., Singh, A., Cheveleva, A., Ligozat, A.-L., Subramonian, A., N'ev'eol, A., Lovering, C., Garrette, D. H., Tunuguntla, D. R., Reiter, E., Taktasheva, E., Voloshina, E., Bogdanov, E., Winata, G. I., Schoelkopf, H., Kalo, J.-C., Novikova, J.,

Forde, J. Z., Clive, J., Kasai, J., Kawamura, K., Hazan, L., Carpuat, M., Clinciu, M., Kim, N., Cheng, N., Serikov, O., Antverg, O., van der Wal, O., Zhang, R., Zhang, R., Gehrmann, S., Pais, S. O., Shavrina, T., Scialom, T., Yun, T., Limisiewicz, T., Rieser, V., Protasov, V., Mikhailov, V., Pruksachatkun, Y., Belinkov, Y., Bamberger, Z., Kasner, Z., Rueda, A., Pestana, A., Feizpour, A., Khan, A., Faranak, A., Santos, A. S. R., Hevia, A., Unldreaj, A., Aghagol, A., Abdollahi, A., Tammour, A., HajiHosseini, A., Behroozi, B., Ajibade, B. O., Saxena, B. K., Ferrandis, C. M., Contractor, D., Lansky, D. M., David, D., Kiela, D., Nguyen, D. A., Tan, E., Baylor, E., Ozoani, E., Mirza, F. T., Ononiwu, F., Rezanejad, H., Jones, H., Bhattacharya, I., Solaiman, I., Sedenko, I., Nejadgholi, I., Passmore, J., Seltzer, J., Sanz, J. B., Fort, K., Dutra, L. M., Samagaio, M., Elbadri, M., Mieskes, M., Gerchick, M., Akinlolu, M., McKenna, M., Qiu, M., Ghauri, M. K. K., Burynok, M., Abrar, N., Rajani, N., Elkott, N., Fahmy, N., Samuel, O. M., An, R., Kromann, R. P., Hao, R., Alizadeh, S., Shubber, S., Wang, S. L., Roy, S., Viguier, S., Le, T.-C., Oyebade, T., Le, T. N. H., Yang, Y., Nguyen, Z. K., Kashyap, A. R., Palasciano, A., Callahan, A., Shukla, A., Miranda-Escalada, A., Singh, A. K., Beilharz, B., Wang, B., de Brito, C. M. F., Zhou, C., Jain, C., Xu, C., Fourrier, C., Perin'an, D. L., Molano, D., Yu, D., Manjavacas, E., Barth, F., Fuhrimann, F., Altay, G., Bayrak, G., Burns, G. A., Vrabec, H. U., Bello, I. I., Dash, I., Kang, J. S., Giorgi, J., Golde, J., Posada, J. D., Sivaraman, K., Bulchandani, L., Liu, L., Shinzato, L., de Bykhovetz, M. H., Takeuchi, M., Pàmies, M., Castillo, M. A., Nezhurina, M., Sanger, M., Samwald, M., Cullan, M., Weinberg, M., Wolf, M., Mihaljcic, M., Liu, M., Freidank, M., Kang, M., Seelam, N., Dahlberg, N., Broad, N. M., Muellner, N., Fung, P., Haller, P., Chandrasekhar, R., Eisenberg, R., Martin, R., Canalli, R. L., Su, R., Su, R., Cahyawijaya, S., Garda, S., Deshmukh, S. S., Mishra, S., Kiblawi, S., Ott, S., Sang-aroonsiri, S., Kumar, S., Schweter, S., Bharati, S. P., Laud, T. A., Gigant, T., Kainuma, T., Kusa, W., Labrak, Y., Bajaj, Y., Venkatraman, Y., Xu, Y., Xu, Y., chao Xu, Y., Tan, Z. X., Xie, Z., Ye, Z., Bras, M., Belkada, Y., and Wolf, T.
Bloom: A 176b-parameter open-access multilingual language model.
ArXiv, abs/2211.05100, 2022.

Schaul, T., Horgan, D., Gregor, K., and Silver, D.
Universal value function approximators.
In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1312–1320, Lille, France, 2015. JMLR.org.

Schmidhuber, J.
An on-line algorithm for dynamic reinforcement learning and planning in reactive environments.
In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 1990.

Schmidhuber, J.
Curious model-building control systems.
In *Proc. international joint conference on neural networks*, pp. 1458–1463, 1991a.

Schmidhuber, J.
Curious model-building control systems.
In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pp. 1458–1463 vol.2. IEEE, 1991b.

ISBN 978-0-7803-0227-3.
doi: 10.1109/IJCNN.1991.170605.
URL https://ieeexplore.ieee.org/document/170605/.

Schmidhuber, J.
Goedel machines: Self-referential universal problem solvers making provably optimal self-improvements.
CoRR, cs.LO/0309048, 2003.
URL http://arxiv.org/abs/cs/0309048.

Schmidhuber, J.
POWERPLAY: training an increasingly general problem solver by continually searching for the simplest still unsolvable problem.
CoRR, abs/1112.5309, 2011.
URL http://arxiv.org/abs/1112.5309.

Schmidhuber, J.
PowerPlay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem.
Frontiers in Psychology, 4, 2013.
ISSN 1664-1078.
doi: 10.3389/fpsyg.2013.00313.
Publisher: Frontiers.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D.
Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model.
Nature, 588(7839):604–609, December 2020.
ISSN 0028-0836, 1476-4687.
doi: 10.1038/s41586-020-03051-4.
URL http://arxiv.org/abs/1911.08265.
arXiv:1911.08265 [cs].

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov., O.
Proximal policy optimization algorithms.
In *arXiv preprint arXiv:1707.06347*, 2017a.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O.
Proximal policy optimization algorithms.
arXiv preprint arXiv:1707.06347, 2017b.

Schulz, L.
The origins of inquiry: inductive inference and exploration in early childhood.
Trends in Cognitive Sciences, 16(7):382–389, 2012.
ISSN 1364-6613.
doi: https://doi.org/10.1016/j.tics.2012.06.004.
URL https://www.sciencedirect.com/science/article/pii/S1364661312001301.

Schulz, L. E. and Sommerville, J.
God does not play dice: Causal determinism and preschoolers' causal inferences.
Child development, 77(2):427–442, 2006.

Schulz, L. E., Standing, H. R., and Bonawitz, E. B.
Word, thought, and deed: the role of object categories in children's inductive inferences and exploratory play.
Developmental Psychology, 44(5):1266, 2008.

Scialom, T., Dray, P., Lamprier, S., Piwowarski, B., and Staiano, J.
Coldgans: Taming language gans with cautious sampling strategies.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
URL https://proceedings.neurips.cc/paper/2020/hash/db261d4f615f0e982983be499e57ccda-Abstract.html.

Scialom, T., Dray, P.-A., Gallinari, P., Lamprier, S., Piwowarski, B., Staiano, J., and Wang, A.
Questeval: Summarization asks for fact-based evaluation.
In *arXiv preprint arXiv:2103.12693v2*, 2021.

Scott Weng Fai, L.
Exploring seven- to eight-year-olds' use of self-talk strategies.
Early Child Development and Care, 181(6):847–856, 2011.
doi: 10.1080/03004430.2010.494253.
URL https://doi.org/10.1080/03004430.2010.494253.

Shannon, C. E.
Programming a computer for playing chess, pp. 2–13.
Springer-Verlag, Berlin, Heidelberg, 1988.
ISBN 0387913319.

Shi, L. X., Ichter, B., Equi, M., Ke, L., Pertsch, K., Vuong, Q., Tanner, J., Walling, A., Wang, H., Fusai, N., Li-Bell, A., Driess, D., Groom, L., Levine, S., and Finn, C.
Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025.
URL https://arxiv.org/abs/2502.19417.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S.
Reflexion: language agents with verbal reinforcement learning.
In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023.

Shortliffe, E.
Computer-based medical consultations: Mycin.
Artificial Intelligence - AI, 388, 10 1976.
doi: 10.1097/00004669-197610000-00011.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D.
Alfred: A benchmark for interpreting grounded instructions for everyday tasks.
In *arXiv preprint arXiv:1912.01734v2*, 2019.

Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. J.
Alfworld: Aligning text and embodied environments for interactive learning.
ArXiv, abs/2010.03768, 2020.

Sigaud, O., Salaun, C., and Padois, V.
On-line regression algorithms for learning mechanical models of robots: A survey.
Robotics and Autonomous Systems, 59(12):1115–1129, 2011.
doi: 10.1016/j.robot.2011.07.006.

Sigaud, O., Baldassarre, G., Colas, C., Doncieux, S., Duro, R. J., Perrin-Gilbert, N., and Santucci, V. G.
A definition of open-ended learning problems for goal-conditioned agents.
ArXiv, abs/2311.00344, 2023.

Sigaud, O., Baldassarre, G., Colas, C., Doncieux, S., Duro, R., Oudeyer, P.-Y., Perrin-Gilbert, N., and Santucci, V. G.
A definition of open-ended learning problems for goal-conditioned agents, 2024.
URL https://arxiv.org/abs/2311.00344.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D.
Mastering the game of go without human knowledge.
Nature, 550(7676):354–359, 2017.
doi: 10.1038/nature24270.

Skyrms, B.
Signals: Evolution, Learning, and Information.
Oxford University Press, 04 2010.
ISBN 9780199580828.
doi: 10.1093/acprof:oso/9780199580828.001.0001.
URL https://doi.org/10.1093/acprof:oso/9780199580828.001.0001.

Smock, C. D. and Holt, B. G.
Children's reactions to novelty: An experimental study of "curiosity motivation".
Child Development, pp. 631–642, 1962.

Sperber, D., Premack, D., and Premack, A. J.
Causal Cognition: A Multidisciplinary Debate.
Clarendon Press, Oxford, 1995.

Stanley, K. O. and Miikkulainen, R.
Evolving neural networks through augmenting topologies.
Evol. Comput., 10(2):99–127, June 2002.
ISSN 1063-6560.
doi: 10.1162/106365602320169811.
URL https://doi.org/10.1162/106365602320169811.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R. J., Voss, C., Radford, A., Amodei, D., and Christiano, P.

Learning to summarize from human feedback.
ArXiv, abs/2009.01325, 2020.

Stolcke, A.
SRILM - an extensible language modeling toolkit.
In Hansen, J. H. L. and Pellom, B. L. (eds.), *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*, pp. 901–904. ISCA, 2002.
doi: 10.21437/ICSLP.2002-303.
URL https://doi.org/10.21437/ICSLP.2002-303.

Stout, A. and Barto, A. G.
Competence progress intrinsic motivation.
In Kuipers, B., Shultz, T. R., Stoytchev, A., and Yu, C. (eds.), *2010 IEEE 9th International Conference on Development and Learning, ICDL 2010, Ann Arbor, MI, USA, August 18-21, 2010*, pp. 257–262. IEEE, 2010a.
doi: 10.1109/DEVLRN.2010.5578835.
URL https://doi.org/10.1109/DEVLRN.2010.5578835.

Stout, A. and Barto, A. G.
Competence progress intrinsic motivation.
In *2010 IEEE 9th International Conference on Development and Learning*, pp. 257–262, 2010b.
doi: 10.1109/DEVLRN.2010.5578835.
ISSN: 2161-9476.

Sudhakaran, S., Duque, M. G., Freiberger, M., Glanois, C., Najarro, E., and Risi, S.
Mariogpt: Open-ended text2level generation through large language models.
In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.
URL http://papers.nips.cc/paper_files/paper/2023/hash/a9bbeb2858dfbdbd4c19814e5d80ec60-Abstract-Conference.html.

Sundermeyer, M., Schlüter, R., and Ney, H.
Lstm neural networks for language modeling.
In *Interspeech*, volume 2012, pp. 194–197, 2012.

Sutton, R. S.
Dyna, an integrated architecture for learning, planning, and reacting.
SIGART Bulletin, 2(4):160–163, 1991.
ISSN 0163-5719.
doi: 10.1145/122344.122377.

Sutton, R. S. and Barto, A. G.
Reinforcement Learning: An Introduction.
MIT press, 2018a.

Sutton, R. S. and Barto, A. G.
Reinforcement Learning: An Introduction.
MIT Press, Cambridge, MA, 2 edition, 2018b.

Sutton, R. S. and Barto, A. G.
Reinforcement Learning: An Introduction.
The MIT Press, second edition, 2018c.
URL http://incompleteideas.net/book/the-book-2nd.html.

Sutton, R. S., Precup, D., and Singh, S.
Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning.
Artif. Intell., 112(1-2):181–211, 1999.
URL http://dblp.uni-trier.de/db/journals/ai/ai112.html#SuttonPS99.

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

Tam, A. C., Rabinowitz, N. C., Lampinen, A. K., Roy, N. A., Chan, S. C., Strouse, D., Wang, J. X., Banino, A., and Hill, F.
Semantic exploration from language abstractions and pretrained representations.
arXiv preprint arXiv:2204.05080, 2022.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P.
#exploration: A study of count-based exploration for deep reinforcement learning.
In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 2753–2762, 2017.
URL https://proceedings.neurips.cc/paper/2017/hash/3a20f62a0af1aa152670bab3c602feed-Abstract.html.

Tang, H., Key, D., and Ellis, K.
WorldCoder, a Model-Based LLM Agent: Building World Models by Writing Code and Interacting with the Environment.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 70148–70212. Curran Associates, Inc., 2024a.
URL https://proceedings.neurips.cc/paper_files/paper/2024/file/820c61a0cd419163ccbd2c33b268816e-Paper-Conference.pdf.

Tang, X., Li, J., Liang, Y., Zhu, S.-c., Zhang, M., and Zheng, Z.
Mars: Situated Inductive Reasoning in an Open-World Environment.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 17830–17869. Curran Associates, Inc., 2024b.

URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1fb6d0b52f5e41b11392841a66dbfe7d-Paper-Datasets_and_Benchmarks_Track.pdf.

Tang, X., Zou, A., Zhang, Z., Li, Z., Zhao, Y., Zhang, X., Cohan, A., and Gerstein, M.
Medagents: Large language models as collaborators for zero-shot medical reasoning.
In Ku, L., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pp. 599–621. Association for Computational Linguistics, 2024c.
doi: 10.18653/V1/2024.FINDINGS-ACL.33.
URL https://doi.org/10.18653/v1/2024.findings-acl.33.

Tasrin, T., Nahian, M. S. A., Perera, H., and Harrison, B.
Influencing reinforcement learning through natural language guidance.
In Bell, E. and Keshtkar, F. (eds.), *Proceedings of the Thirty-Fourth International Florida Artificial Intelligence Research Society Conference, North Miami Beach, Florida, USA, May 17-19, 2021*. Florida Online Journals, 2021.
doi: 10.32473/FLAIRS.V34I1.128472.
URL https://doi.org/10.32473/flairs.v34i1.128472.

Team, S., Raad, M. A., Ahuja, A., Barros, C., Besse, F., Bolt, A., Bolton, A., Brownfield, B., Buttimore, G., Cant, M., Chakera, S., Chan, S. C. Y., Clune, J., Collister, A., Copeman, V., Cullum, A., Dasgupta, I., de Cesare, D., Trapani, J. D., Donchev, Y., Dunleavy, E., Engelcke, M., Faulkner, R., Garcia, F., Gbadamosi, C., Gong, Z., Gonzalez, L., Gupta, K., Gregor, K., Hallingstad, A. O., Harley, T., Haves, S., Hill, F., Hirst, E., Hudson, D. A., Hudson, J., Hughes-Fitt, S., Rezende, D. J., Jasarevic, M., Kampis, L., Ke, N. R., Keck, T., Kim, J., Knagg, O., Kopparapu, K., Lampinen, A. K., Legg, S., Lerchner, A., Limont, M., Liu, Y., Loks-Thompson, M., Marino, J., Cussons, K. M., Matthey, L., Mcloughlin, S., Mendolicchio, P., Merzic, H., Mitenkova, A., Moufarek, A., Oliveira, V., Oliveira, Y. G., Openshaw, H., Pan, R., Pappu, A., Platonov, A., Purkiss, O., Reichert, D. P., Reid, J., Richemond, P. H., Roberts, T., Ruscoe, G., Elias, J. S., Sandars, T., Sawyer, D. P., Scholtes, T., Simmons, G., Slater, D., Soyer, H., Strathmann, H., Stys, P., Tam, A. C., Teplyashin, D., Terzi, T., Vercelli, D., Vujatovic, B., Wainwright, M., Wang, J. X., Wang, Z., Wierstra, D., Williams, D., Wong, N., York, S., and Young, N.
Scaling instructable agents across many simulated worlds.
CoRR, abs/2404.10179, 2024.
doi: 10.48550/ARXIV.2404.10179.
URL https://doi.org/10.48550/arXiv.2404.10179.

Ten, A., Gottlieb, J., and Oudeyer, P.-Y.
Intrinsic rewards in human curiosity-driven exploration: An empirical study.
Proceedings of the Annual Meeting of the Cognitive Science Society, 43(43), 2021a.

Ten, A., Kaushik, P., Oudeyer, P.-Y., and Gottlieb, J.
Humans monitor learning progress in curiosity-driven exploration.
Nature Communications, 12(1):5972, 2021b.
ISSN 2041-1723.
doi: 10.1038/s41467-021-26196-w.
URL https://doi.org/10.1038/s41467-021-26196-w.

Ten, A., Kaushik, P., Oudeyer, P.-Y., and Gottlieb, J.
Humans monitor learning progress in curiosity-driven exploration.
Nature Communications, 12(1):5972, 2021c.
ISSN 2041-1723.
doi: 10.1038/s41467-021-26196-w.
Publisher: Nature Publishing Group.

Teodorescu, L., Colas, C., Bowers, M., Carta, T., and Oudeyer, P.-Y.
Codeplay: Autotelic Learning through Collaborative Self-Play in Programming Environments.
In *IMOL 2023 - Intrinsically Motivated Open-ended Learning workshop at NeurIPS 2023*, New Orleans, United States, December 2023.
URL https://hal.science/hal-04374993.

Thill, S., n, S. P., and Ziemke, T.
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.

Thorndike, E. L.
Animal Intelligence: An Experimental Study of the Associative Processes in Animals, volume 2 of *Psychological Review Monograph Supplements*.
Macmillan, New York, 1898.

Thorndike, E. L.
A proof of the law of effect.
Science, 1933.

Thrun, S.
A lifelong learning perspective for mobile robot control.
In *Intelligent Robots and Systems*, pp. 201–214. Elsevier Science BV, January 1995.

Touati, A. and Ollivier, Y.
Learning one representation to optimize all rewards.
In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 13–23, 2021.
URL https://proceedings.neurips.cc/paper/2021/hash/003dd617c12d444ff9c80f717c3fa982-Abstract.html.

Tsay, J. S., Kim, H. E., McDougle, S. D., Taylor, J. A., Haith, A., Avraham, G., Krakauer, J. W., Collins, A. G., and Ivry, R. B.
Fundamental processes in sensorimotor learning: Reasoning, refinement, and retrieval.
eLife, 13:e91839, aug 2024.
ISSN 2050-084X.
doi: 10.7554/eLife.91839.
URL https://doi.org/10.7554/eLife.91839.

Tsividis, P. A., Loula, J., Burga, J., Foss, N., Campero, A., Pouncy, T., Gershman, S. J., and Tenenbaum, J. B.
Human-level reinforcement learning through theory-based modeling, exploration, and planning, 2021.
URL http://arxiv.org/abs/2107.12544.

Turing, A.
Intelligent machinery (1948).
B. Jack Copeland, pp. 395, 2004.

Turing, A. M.
Computing machinery and intelligence.
Mind, 59(236):433–460, 1950.
ISSN 00264423.
URL http://www.jstor.org/stable/2251299.

Vafa, K., Chen, J. Y., Rambachan, A., Kleinberg, J., and Mullainathan, S.
Evaluating the World Model Implicit in a Generative Model.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 26941–26975. Curran Associates, Inc., 2024.
URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2f6a6317bada76b26a4f61bb70a7db59-Paper-Conference.pdf.

Valmeekam, K., Olmo, A., Sreedharan, S., and Kambhampati, S.
Large language models still can't plan (a benchmark for llms on planning and reasoning about change).
ArXiv, abs/2206.10498, 2022.

Valmeekam, K., Sreedharan, S., Marquez, M., Olmo, A., and Kambhampati, S.
On the planning abilities of large language models (a critical investigation with a proposed benchmark), 2023.
URL https://arxiv.org/abs/2302.06706.

van der Maaten, L. and Hinton, G.
Visualizing data using t-sne.
Journal of Machine Learning Research, 9(86):2579–2605, 2008.

Van Hasselt, H., Guez, A., and Silver, D.
Deep reinforcement learning with double q-learning.
In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I.
Attention is all you need.
Advances in neural information processing systems, 30, 2017.

Veeriah, V., Zahavy, T., Hessel, M., Xu, Z., Oh, J., Kemaev, I., van Hasselt, H., Silver, D., and Singh, S.
Discovery of options via meta-learned subgoals.

In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 29861–29873, 2021.
URL https://proceedings.neurips.cc/paper/2021/hash/fa246d0262c3925617b0c72bb20eeb1d-Abstract.html.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K.
Feudal networks for hierarchical reinforcement learning, 2017.
URL https://arxiv.org/abs/1703.01161.

Vijayakumar, S., D'Souza, A., and Schaal, S.
Incremental online learning in high dimensions.
Neural Computation, 17(12):2602–2634, 2005.
doi: 10.1162/089976605774320539.

Villalobos, P., Ho, A., Sevilla, J., Besiroglu, T., Heim, L., and Hobbhahn, M.
Will we run out of data? limits of llm scaling based on human-generated data, 2024.
URL https://arxiv.org/abs/2211.04325.

Vygotsky, L.
Thought and language.
The vygotsky reader, 1934.

Vyshedskiy, A.
Language evolution to revolution: The leap from rich-vocabulary non-recursive communication system to recursive language 70,000 years ago was associated with acquisition of a novel component of imagination, called prefrontal synthesis, enabled by a mutation that slowed down the prefrontal cortex maturation simultaneously in two or more children – the romulus and remus hypothesis.
Research Ideas and Outcomes, 5, 2019.
doi: 10.3897/rio.5.e38546.
URL https://doi.org/10.3897/rio.5.e38546.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A.
Voyager: An open-ended embodied agent with large language models.
arXiv preprint arXiv:2305.16291, 2023a.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A.
Voyager: An open-ended embodied agent with large language models, 2023b.
URL https://arxiv.org/abs/2305.16291.

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

Wang, R., Lehman, J., Clune, J., and Stanley, K. O.
Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions, 2019.
URL https://arxiv.org/abs/1901.01753.

Wang, R., Lehman, J., Rawal, A., Zhi, J., Li, Y., Clune, J., and Stanley, K.
Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions.
In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9940–9951. PMLR, 13–18 Jul 2020.
URL https://proceedings.mlr.press/v119/wang20l.html.

Wang, R., Jansen, P. A., Côté, M.-A., and Ammanabrolu, P.
Scienceworld: Is your agent smarter than a 5th grader?
ArXiv, abs/2203.07540, 2022.

Wang, R., Zelikman, E., Poesia, G., Pu, Y., Haber, N., and Goodman, N. D.
Hypothesis search: Inductive reasoning with language models.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *12th International Conference on Learning Representations (ICLR 2024)*. Curran Associates, Inc., 2024b.

Wang, S., Liu, C., Zheng, Z., Qi, S., Chen, S., Yang, Q., Zhao, A., Wang, C., Song, S., and Huang, G.
Avalon's game of thoughts: Battle against deception through recursive contemplation.
CoRR, abs/2310.01320, 2023c.
doi: 10.48550/ARXIV.2310.01320.
URL https://doi.org/10.48550/arXiv.2310.01320.

Wang, Y., Yang, M., Dong, R., Sun, B., Liu, F., et al.
Efficient potential-based exploration in reinforcement learning using inverse dynamic bisimulation metric.
Advances in Neural Information Processing Systems, 36:38786–38797, 2023d.

Wang, Y., Zhao, K., Liu, F., et al.
Rethinking exploration in reinforcement learning with effective metric-based exploration bonus.
Advances in Neural Information Processing Systems, 37:57765–57792, 2024c.

Wang, Z., He, W., Wu, H., Wu, H., Li, W., Wang, H., and Chen, E.
Chinese poetry generation with planning based neural network.
CoRR, abs/1610.09889, 2016a.
URL http://arxiv.org/abs/1610.09889.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N.
Dueling network architectures for deep reinforcement learning.
In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016b.

Warde-Farley, D., de Wiele, T. V., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V.
Unsupervised control through non-parametric discriminative rewards.
In *International Conference on Learning Representations*, 2019.

Watkins, C. J. and Dayan, P.
Q-learning.
Machine learning, 8(3):279–292, 1992a.

Watkins, C. J. C. H. and Dayan, P.
Technical note q-learning.
Mach. Learn., 8:279–292, 1992b.
doi: 10.1007/BF00992698.
URL https://doi.org/10.1007/BF00992698.

Waxman, S. R. and Markow, D. B.
Words as invitations to form categories: Evidence from 12- to 13-month-old infants.
Cognitive Psychology, 29(3):257–302, 1995.
doi: 10.1006/cogp.1995.1016.

Waytowich, N., Barton, S. L., Lawhern, V., and Warnell, G.
A narration-based reward shaping approach using grounded natural language commands.
In *International Conference on Machine Learning (ICML)*, 2019.

Webb, T., Holyoak, K. J., and Lu, H.
Emergent analogical reasoning in large language models, 2023.
URL https://arxiv.org/abs/2212.09196.

Webb, T. W., Holyoak, K. J., and Lu, H.
Emergent analogical reasoning in large language models.
CoRR, abs/2212.09196, 2022.
doi: 10.48550/ARXIV.2212.09196.
URL https://doi.org/10.48550/arXiv.2212.09196.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W.
Emergent abilities of large language models.
Trans. Mach. Learn. Res., 2022, 2022a.
URL https://openreview.net/forum?id=yzkSU5zdwD.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., hsin Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W.
Emergent abilities of large language models.
ArXiv, abs/2206.07682, 2022b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., ichter, b., Xia, F., Chi, E., Le, Q. V., and Zhou, D.
Chain-of-thought prompting elicits reasoning in large language models.

In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022c.
URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Weiller, D., Märtin, R., Dähne, S., Engel, A., and König, P.
Involving motor capabilities in the formation of sensory space representations.
PLoS One, 2010.
doi: 10.1371/journal.pone.0010377.

Wen, M., Deng, C., Wang, J., Zhang, W., and Wen, Y.
Entropy-regularized token-level policy optimization for large language models, 2024a.

Wen, M., Wan, Z., Wang, J., Zhang, W., and Wen, Y.
Reinforcing llm agents via policy optimization with action decomposition.
In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 103774–103805. Curran Associates, Inc., 2024b.
URL https://proceedings.neurips.cc/paper_files/paper/2024/file/bc09efb501c801ed92e181e26a885c2d-Paper-Conference.pdf.

Wen, M., Wan, Z., Wang, J., Zhang, W., and Wen, Y.
Reinforcing LLM agents via policy optimization with action decomposition, 2024c.

Williams, R. J.
Simple statistical gradient-following algorithms for connectionist reinforcement learning.
Machine Learning, 8:229–256, 1992.

Winograd, T.
Understanding natural language.
British Journal for the Philosophy of Science, 25(1):85–88, 1974.

Wiriyathammabhum, P., Summers-Stay, D., Fermüller, C., and Aloimonos, Y.
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.

Wong, L., Grand, G., Lew, A. K., Goodman, N. D., Mansinghka, V. K., Andreas, J., and Tenenbaum, J. B.
From word models to world models: Translating from natural language to the probabilistic language of thought.
CoRR, abs/2306.12672, 2023.
doi: 10.48550/ARXIV.2306.12672.
URL https://doi.org/10.48550/arXiv.2306.12672.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu,

X., Yin, Z., Dou, S., Weng, R., Qin, W., Zheng, Y., Qiu, X., Huang, X., Zhang, Q., and Gui, T.
The rise and potential of large language model based agents: a survey.
Sci. China Inf. Sci., 68(2), 2025.
doi: 10.1007/S11432-024-4222-0.
URL https://doi.org/10.1007/s11432-024-4222-0.

Xiang, J., Tao, T., Gu, Y., Shu, T., Wang, Z., Yang, Z., and Hu, Z.
Language models meet world models: embodied experiences enhance language models.
In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 75392–75412. Curran Associates Inc., December 2023.

Xiong, H., Wang, Z., Li, X., Bian, J., Xie, Z., Mumtaz, S., Al-Dulaimi, A., and Barnes, L. E.
Converging paradigms: The synergy of symbolic and connectionist ai in llm-empowered autonomous agents, 2024.
URL https://arxiv.org/abs/2407.08516.

Xu, Y., Fang, M., Chen, L., Du, Y., Zhou, J. T., and Zhang, C.
Perceiving the world: Question-guided reinforcement learning for text-based games.
In *arXiv preprint arXiv:2204.09597v2*, 2022.

Xu, Z., Yu, C., Fang, F., Wang, Y., and Wu, Y.
Language agents with reinforcement learning for strategic play in the werewolf game.
In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.
URL https://openreview.net/forum?id=usUPvQH3XK.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y.
React: Synergizing reasoning and acting in language models.
ArXiv, abs/2210.03629, 2022a.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y.
React: Synergizing reasoning and acting in language models.
In *International Conference on Learning Representations (ICLR)*, 2022b.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K.
Tree of thoughts: Deliberate problem solving with large language models.
In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023a.
URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y.
React: Synergizing reasoning and acting in language models.
In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b.
URL https://openreview.net/forum?id=WE_vluYUL-X.

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., and Wu, Y.
Coca: Contrastive captioners are image-text foundation models, 2022.
URL https://arxiv.org/abs/2205.01917.

Yu, L., Zhang, W., Wang, J., and Yu, Y.
Seqgan: Sequence generative adversarial nets with policy gradient.
In Singh, S. and Markovitch, S. (eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 2852–2858. AAAI Press, 2017.
doi: 10.1609/AAAI.V31I1.10804.
URL https://doi.org/10.1609/aaai.v31i1.10804.

Yuan, L., Chen, D., Chen, Y.-L., Codella, N. C. F., Dai, X., Gao, J., Hu, H., Huang, X., Li, B., Li, C., Liu, C., Liu, M., Liu, Z., Lu, Y., Shi, Y., Wang, L., Wang, J., Xiao, B., Xiao, Z., Yang, J., Zeng, M., Zhou, L., and Zhang, P.
Florence: A new foundation model for computer vision.
ArXiv, abs/2111.11432, 2021.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Yue, Y., Song, S., and Huang, G.
Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025.
URL https://arxiv.org/abs/2504.13837.

Zhang, A., Nguyen, K., Tuyls, J., Lin, A., and Narasimhan, K.
Language-guided World Models: A Model-based Approach to AI Control.
In Kordjamshidi, P., Wang, X. E., Zhang, Y., Ma, Z., and Inan, M. (eds.), *Proceedings of the 4th Workshop on Spatial Language Understanding and Grounded Communication for Robotics (SpLU-RoboNLP 2024)*, pp. 1–16, Bangkok, Thailand, August 2024a. Association for Computational Linguistics.
URL https://aclanthology.org/2024.splurobonlp-1.1/.

Zhang, J., Lehman, J., Stanley, K., and Clune, J.
Omni: Open-endedness via models of human notions of interestingness.
arXiv preprint arXiv:2306.01711, 2023a.

Zhang, J., Lehman, J., Stanley, K., and Clune, J.
Omni: Open-endedness via models of human notions of interestingness.
In *The Twelfth International Conference on Learning Representations*, 2023b.

Zhang, J., Zhang, J., Pertsch, K., Liu, Z., Ren, X., Chang, M., Sun, S.-H., and Lim, J. J.
Bootstrap your own skills: Learning to solve new tasks with large language model guidance.
In *7th Annual Conference on Robot Learning*, 2023c.
URL https://openreview.net/forum?id=a0mFRgadGO.

Zhang, J., Lehman, J., Stanley, K., and Clune, J.
OMNI: Open-endedness via models of human notions of interestingness.
In *The Twelfth International Conference on Learning Representations*, 2024b.

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., and Tian, Y.
Noveld: A simple yet effective exploration criterion.

In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 25217–25230, 2021.
URL https://proceedings.neurips.cc/paper/2021/hash/d428d070622e0f4363fceae11f4a3576-Abstract.html.

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

Zheng, Z., Zhang, O., Nguyen, H. L., Rampal, N., Alawadhi, A. H., Rong, Z., Head-Gordon, T., Borgs, C., Chayes, J. T., and Yaghi, O. M.
Chatgpt research group for optimizing the crystallinity of mofs and cofs.
ACS Central Science, 9(11):2161–2170, 2023.
doi: 10.1021/acscentsci.3c01087.
URL https://doi.org/10.1021/acscentsci.3c01087.

Zhong, W., Guo, L., Gao, Q., Ye, H., and Wang, Y.
Memorybank: Enhancing large language models with long-term memory.
In Wooldridge, M. J., Dy, J. G., and Natarajan, S. (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 19724–19731. AAAI Press, 2024.
doi: 10.1609/AAAI.V38I17.29946.
URL https://doi.org/10.1609/aaai.v38i17.29946.

Zhou, Y., Zanette, A., Pan, J., Levine, S., and Kumar, A.
ArCHer: Training language model agents via hierarchical multi-turn RL.
In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62178–62209. PMLR, 21–27 Jul 2024.

Zhu, W., Yan, A., Lu, Y., Xu, W., Wang, X. E., Eckstein, M., and Wang, W. Y.
Visualize before you write: Imagination-guided open-ended text generation, 2023.
URL https://arxiv.org/abs/2210.03765.

Appendices

Appendix A

Appendix of GLAM

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

- Section A.1 presents the BabyAI and BabyAI-Text environments.
- Section A.2, contains several additional results. We report the per-task success rate at the end of the training (A.2.1) and how we have averaged the results over the tasks in A.2.2. 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 A.2.3. We then study the impact of the size of the action space (A.2.4), and the effect of the number of distractors (A.2.4).
- Section A.3 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 A.4, we detail complementary tests for questions Q2 (A.4.2) and Q3 (A.4.3).
- Section A.5 gives details related to the distributed experimental setup.
- Section A.6 reports hyperparameters and implementation details used to finetune the models using PPO or Behavioral Cloning.
- In Section A.7, we detail how the confidence intervals given in Figure 4.12 and Appendix A.4 are obtained.
- In Section A.8, we give the word substitutions used in our generalization experiments from sections 4.5.2 and 4.5.3.

A.1 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 A.1 explains our environment.

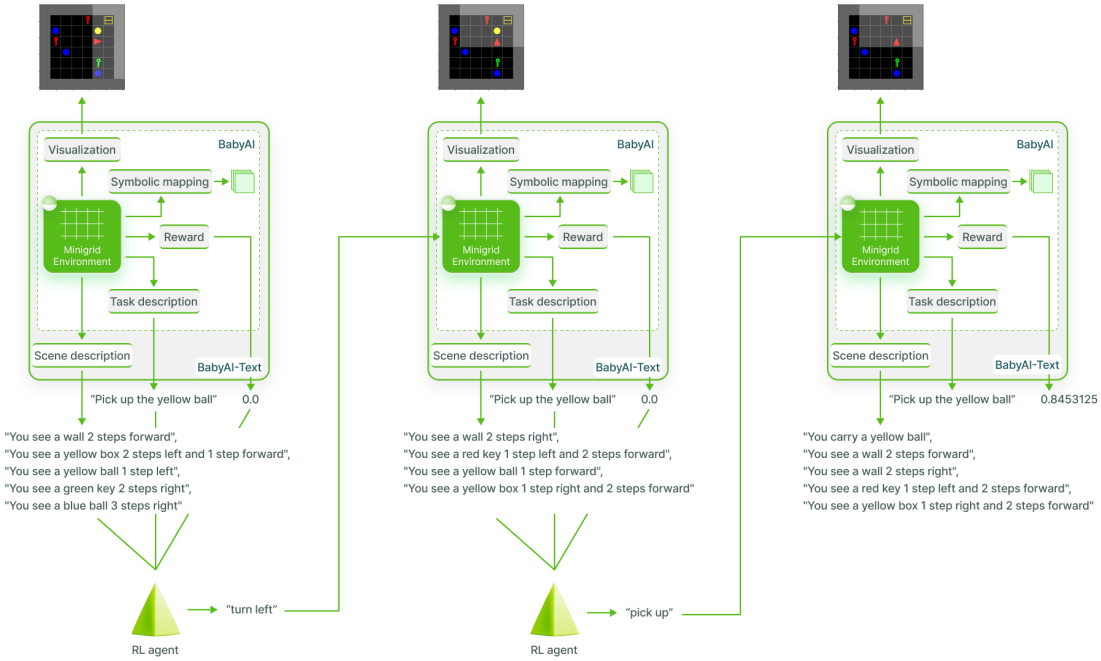


Figure A.1: An illustration of how our BabyAI-Text environment encapsulates BabyAI. We keep the inner minigrid environment as well as task descriptions and reward but map the partial view of the agent to a text description.

A.1.1 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 $r_N = 1 - 0.9\frac{N}{T}$, where T 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.

A.1.2 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 A.1 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 A.1 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).

A.2 Additional results

A.2.1 Per-task success rate

In order to get a better understanding of our agent's capabilities, we report in Figure A.2 the success rate on each task from our "no-change" evaluation in Figure 4.12 (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.

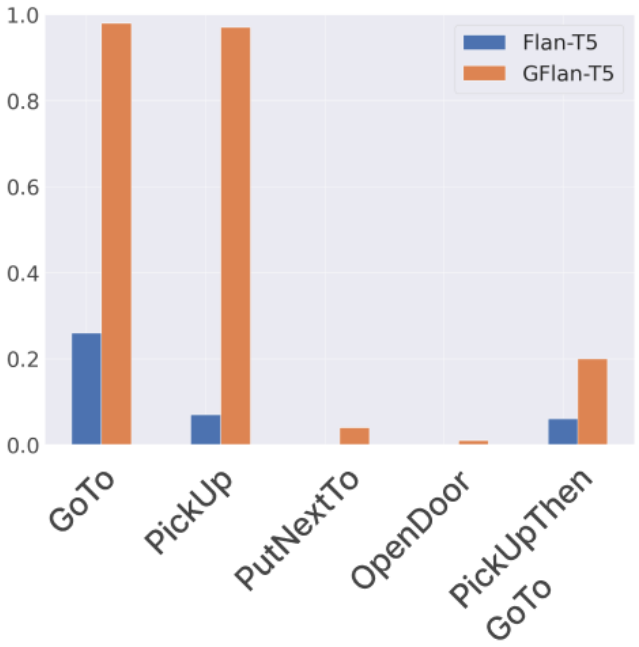


Figure A.2: Per-task success rate for the 1000 evaluation trajectories performed in Figure 4.12.

A.2.2 Averaging success rate over the tasks

In our environments, goals are divided into five types of tasks of varying difficulty. Each goal is sampled randomly from the tasks when the environment is reset. We obtain the success rate for the goals at update u by averaging over the completed trajectories during the collection phase. As several environments run in parallel, goals from easier tasks that are completed more quickly, such as *GoTo* and *PickUp*, tend to be more represented in the buffer.

A.2.3 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.

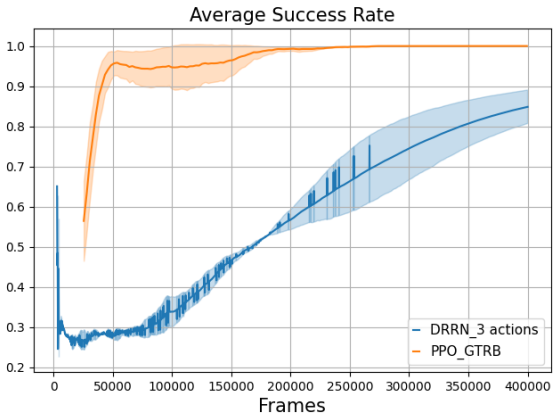


Figure A.3: Average success rate for DRRN and Symbolic-PPO on the *Go To Red Ball* task with standard deviation over two random seeds. The PPO receives symbolic information and the DRRN gets textual observations.

Contrary to what one might assume in Figure A.3 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.

A.2.4 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.

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 A.4 report full learning curves used to draw Figure 4.10a. 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.

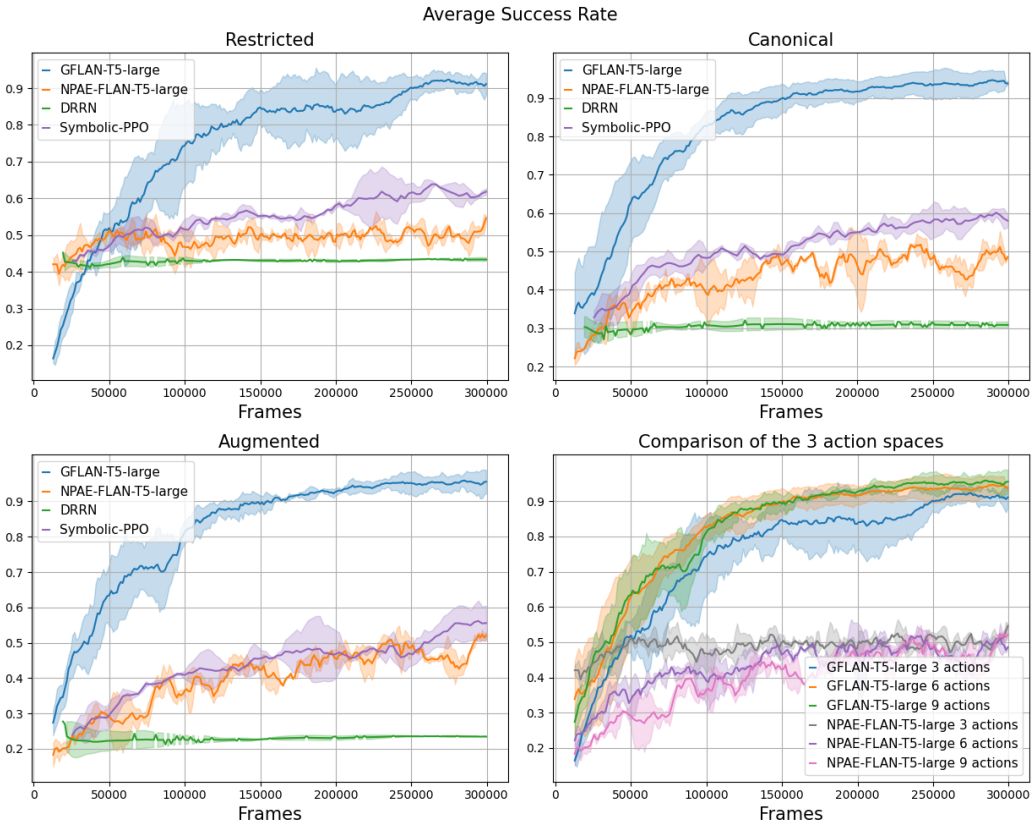


Figure A.4: Learning curves for the agents on the *Go To* task for different sizes of action space (Restricted: 3 actions, Canonical: 6, Augmented: 9, with only the 3 actions that are useful). The success rate is given over 2 seeds along with standard deviation.

Impact of the number of distractors

In Figure 4.10b we have shown that LLMs are less sensitive to variations on task complexity by plotting the evolution of sample efficiency (Equation (4.3)) for different numbers of distractors: 4, 8 and 16. In Figure A.5 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.

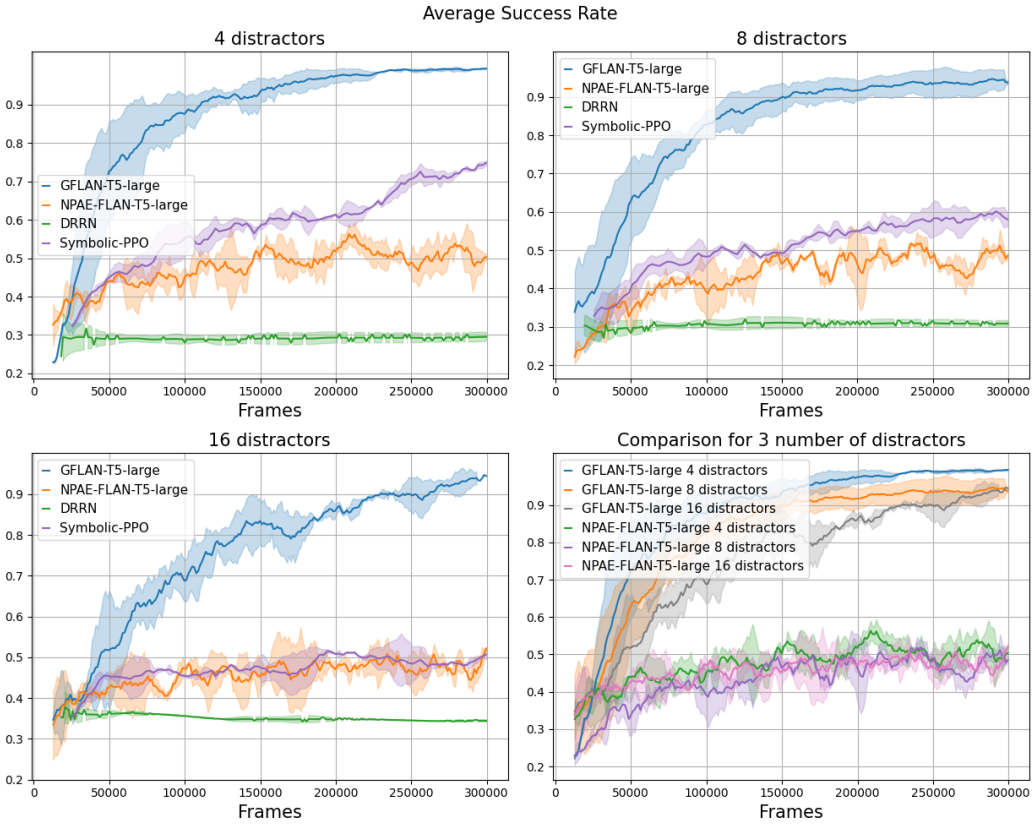


Figure A.5: Learning curves for the agents on the *Go To* task for different number of distractors (4, 8, 16). The success rate is given over 2 seeds with standard deviation.

A.3 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 4.5.1, we test at each update the LLM on 11 prompts listed in Table A.1. We plot in Figure A.6 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 A.6).

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 Section 4.5.3.

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 4.5.3, 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 A.1: 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

ID	Task	Prompt	Comments
0	Go To <object>	**Goal**: 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**: 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**: go forward	Simple navigation task.

Continued on next page

Table A.1: Table A.1 (continued)

ID	Task	Prompt	Comments
2	Open <adj> door	**Goal**: 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**: 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**: **Expected answer**: pick up	Inference task. The agent has to infer that a key of the same color is needed and moves toward it.
4	Open <adj> door	**Goal**: 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.

Continued on next page

Table A.1: Table A.1 (continued)

ID	Task	Prompt	Comments
5	Pick up <object>	**Goal**: 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**: 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**: ou see a wall 2 steps left, You see a red ball 2 steps right, You see a green box 1 step forward, **Action 1**: **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**: 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 task and understand the geometry of the room.
8	Pick up <object A> then go to <object B>	**Goal**: 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 agent's ability to understand temporal concepts.
9	Go to <object B> after you pick up <object A>	**Goal**: 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**: **Expected answer**: turn left	Same as prompt 8 but sentence action order different from execution order.

Continued on next page

Table A.1: Table A.1 (continued)

ID	Task	Prompt	Comments
10	Pick up <object A> after you pick up <object B>	**Goal**: 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.

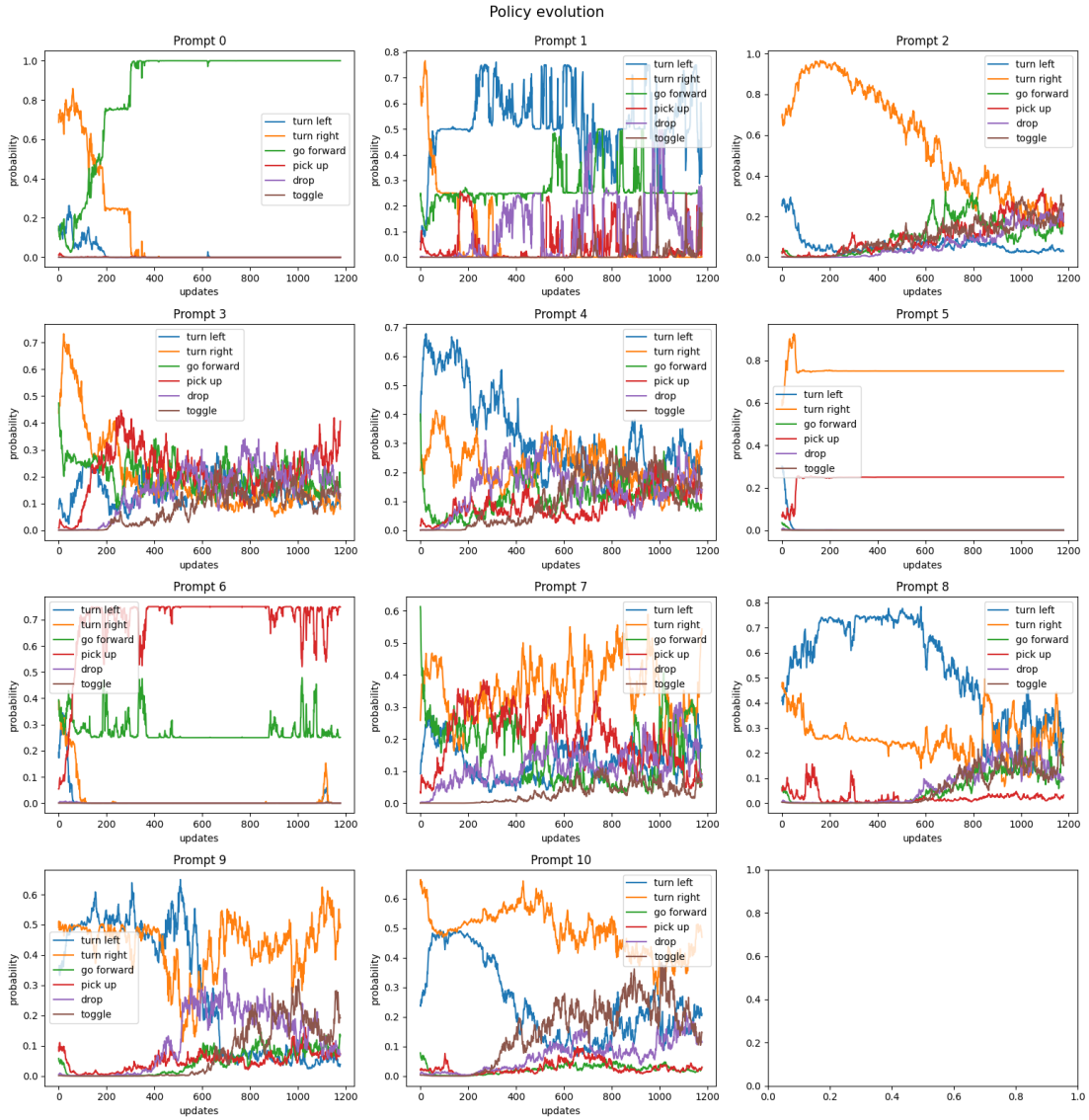


Figure A.6: Evolution of actions' probability over training for test prompts listed in Table A.1.

A.4 Generalization tests details

A.4.1 Recapitulating results table

In this section we summarize the numerical results shown in Figure 4.12 with the confidence intervals calculated as explained in Appendix A.7.

Table A.2: 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	

A.4.2 Complementary tests for Q2

Table A.3: 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 4.5.2, we conduct more systematic tests on different aspects of the generalization to new words. The results are given in Table A.3.

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 4.5.2 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 A.7). 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.

A.4.3 Complementary tests for Q3

Table A.4: 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	

In Section 4.5.3, 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 but actions are in French. In this setting, Table A.4 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.

A.5 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 4.4, 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 4.5 4 instances of our LLM all running in parallel. We load and use LLMs through the Hugging Face Transformers Python library[1]. 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[2] 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[3] 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 4.5.1).

When using **Flan-T5 780M**, each LLM instance is distributed (Vertical Model Parallelism[4]) 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.

[1] https://huggingface.co/docs/transformers/index
[2] https://pytorch.org/docs/stable/distributed.html
[3] https://pytorch.org/tutorials/intermediate/ddp_tutorial.html
[4] Layers are spread across GPUs (https://huggingface.co/docs/transformers/v4.15.0/parallelism)

A.6 finetuning details

A.6.1 PPO finetuning details

We reused PPO's hyperparameters from Ramamurthy et al. (2022) and did not perform any further tuning (see Table A.5). We used an Adam (Kingma & Ba, 2014) optimizer with the hyperparameters listed in Table A.6). For additional heads, we used MLPs with 3 hidden layers of 1024 units with Sigmoid activation.

Table A.5: PPO hyperparameters

Variable	Value
Number of transitions collected between two updates	1280 (32 environments ×40 steps in each environ
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 A.6: Adam hyperparameters

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

A.6.2 Behavioral Cloning

In Section 4.5.4, 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 A.4, 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 4.5 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.

A.7 Confidence interval

In sections 4.5.2 and 4.5.3, 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.

A.7.1 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[5]. Thus we have

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

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

Moreover, one property of normal random variables is that if

$$\begin{cases} V & \sim \mathcal{N}(V_0, \Sigma_V) \\ U|V & \sim \mathcal{N}(U_0 + XV, \Sigma_{U|V}) \end{cases} \tag{A.2}$$

for any X,then

$$\begin{pmatrix} U \\ V \end{pmatrix} \sim \mathcal{N}(\begin{pmatrix} U_0 + XV_0 \\ V_0 \end{pmatrix}, \begin{pmatrix} X\Sigma_V X^T + \Sigma_{U|V} & X\Sigma_V \\ \Sigma_V X^T & \Sigma_V \end{pmatrix}) \tag{A.3}$$

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

By identification with Equation A.1, we have

$$Y_n^i \sim \mathcal{N}(np, (n\tau)^2 + n\,p_i(1-p_i)) \tag{A.4}$$

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

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).

$$SR_i = \frac{Y_n^i}{n} \sim \mathcal{N}(p, \tau^2 + \frac{p_i(1-p_i)}{n}) \tag{A.5}$$

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:

$$SR_i \sim \mathcal{N}(p, \tau^2) \tag{A.6}$$

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

$$\hat{p} \pm \ 2.58\frac{\hat{\tau}}{\sqrt{s}} \tag{A.7}$$

$$\begin{cases} \hat{p} & = \frac{1}{s}\sum_{i=1}^{s} SR_i \\ \hat{\tau}^2 & = \frac{1}{(s-1)}\sum_{i=1}^{s}(\hat{p} - SR_i)^2 \end{cases} \tag{A.8}$$

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

A.7.2 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:

$$\mathbb{P}(|SR_n - p| > \varepsilon) < 2\exp\left(-2n\varepsilon^2\right) = \delta \tag{A.9}$$

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 $\varepsilon = 0.05$.

A.8 Word substitutions for generalization tests

For the generalization tests given in the sections 4.5.2, 4.5.3, and A.4, we use the dictionaries given below to substitute some words by others.

A.8.1 Out of vocabulary

To generate descriptions with out-of-vocabulary nouns and adjectives, we modify the prompt by substituting words as per Table A.7.

Table A.7: 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

A.8.2 Invented words

Similarly to Section A.8.1, we apply the substitutions indicated in Table A.8.

Table A.8: 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

A.8.3 Synonym actions

In A.9, 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 A.9: Synonym actions

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

A.8.4 Translation to French

We give in Table A.10 the chosen translation for the french environment (the adjectives are given in the feminine form as all the objects are feminine).

Table A.10: 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

Appendix B

Appendix of WorldLLM

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

- Section B.1 presents the Little-Zoo environment.
- Section B.2, contains additional details about the method. In particular, in Section B.2 we show the distribution of transitions collected by each oracle and in Section B.2 we give the RL agent architecture its training hyperparameters.
- Section B.3 explains how the syntax of the prompt is modified for the generalization tests.
- Section C.4 we expand on several results presented in Section 5.5.1 such as the evolution of likelihoods of the different methods (Section B.4.1). We also conduct in Section B.4.2 a quantitative analysis of the hypothesis generated by the scientist during training. In Section B.4.3 we present some hypotheses produced by the scientist. Finally, in Section B.4.4 we propose a deeper study of RL-ALP.

B.1 The Playground-Text Environment

We use the Playground-Text environment from (Gaven et al., 2024), itself being a textual adaptation of the Playground environment introduced by (Colas et al., 2020a). This textual environment returns, at each time step, a description of its state and possible actions. In our experiments, we used a modified version of Playground-Text where the current state still describes the whole environment's state but the next state only decribes how the performed action changed the environment (see Figure B.1).

Playground-Text features 4 different classes of objects with different varieties: *Water*, *Plants* (carrot, potato, beet, berry and pea), *Small Herbivores* (pig, cow and sheep) and *Big Herbivores* (elephant, giraffe, rhinoceros). As depicted in Figure 5.2, all objects except *Water* possess two states: young (seed for plants and baby for animals) and grown. These objects transition from their young state to their grown one by consuming other objects.

Three different types of actions are possible: *Go to <object>*, to move to the selected object, *Grasp*, to put the object the agent is standing on in its inventory (which has two slots) and *Release <object>/all* to release one or all objects from the inventory on the object the agent is currently standing on. The resulting transitions can be categorized into 6 different types: *Standing*, *Holding 1*, *Holding 2*, *Grow Plant*, *Grow Small Herbivore* or *Grow Big Herbivore*. Growing transitions are triggered when the appropriate objects are released on a young one (see the technology tree in Figure 5.2). All actions involving non-existent objects in the current scene, as well as release actions that do not result in a transformation (e.g. releasing water onto water) are considered illegal and masked.

State: You see the baby cow, the water, the potato seed, the baby rhinoceros, the water, the pea seed, the water, the potato seed. You are next to the potato seed. In your inventory, there is nothing.
Action: You go to the water.
Change: You are standing on the water.

State: You see the baby cow, the water, the potato seed, the baby rhinoceros, the water, the pea seed, the water, the potato seed. You are next to the water. In your inventory, there is nothing.
Action: You pick up the object.
Change: In your inventory, there is the water.

State: You see the baby cow, the potato seed, the baby rhinoceros, the water, the pea seed, the water, the potato seed. In your inventory, there is the water.
Action: You go to the potato seed.
Change: You are standing on the potato seed.

State: You see the baby cow, the potato seed, the baby rhinoceros, the water, the pea seed, the water, the potato seed. You are next to the potato seed. In your inventory, there is the water.
Action: You give the water.
Change: The objects transform into the potato.

Figure B.1: An example of a trajectory in Playground-Text.

B.2 WorldLLM details

In this section, we provide details about our WorldLLM implementation. All our LLMs were loaded and prompted using the *transformers* library from Hugging Face. We also provide our code for reproduction on the following repository: https://github.com/flowersteam/WorldLLM.

Statistician

The LLM used for the Statistician is *Phi-3-mini-4k-instruct* quantized to 4 bits. The prompt used is shown in Figure B.2.

System Prompt:
You like doing a lot of puzzles. Please answer with a brief answer and be as precise as you can.
User Prompt:
You are in an environment that contains multiple objects. It can contain water, plant seeds(carrot, porato, beet, berry and pea seeds), small herbivores(pig, cow and ship) and large herbivores(elephant, giraffe, rhinoceros). You can move objects, like water, plants or herbivores and release them on another object to make them interact and transform into a new object. You know that:
<hypothesis>
Your objective is to predict the next change in the environment given the state of the environment and the action taken.
The last state was: <state>
The last action was: <action>
The change is: **Assistant Prompt:**
<change>

Figure B.2: Prompt used for the Statistician. The <hypothesis> corresponds to the hypotheses to test. The <state>, <action> and <change> represent the current state, the action taken, and the resulting change, respectively, for the given transition.

Scientist

The LLM used for the Scientist is the same as the one used for the Statistician. We show its prompt in Figure B.3.

Experimenter

For the Experimenter, we provide further details for both our oracles and RL-based policies.

System Prompt:
You like doing a lot of puzzles. Please answer with a brief answer and be as precise as you can.
User Prompt:
You are in an environment that contains multiple objects. It can contain water, plant seeds(carrot, porato, beet, berry and pea seeds), small herbivores(pig, cow and ship) and large herbivores(elephant, giraffe, rhinoceros). You can move objects, like water, plants or herbivores and release them on another object to make them interact and transform into a new object. Your previous experiences were:
<trajectories>
Can you find a set of easily understandable and concise hypotheses to predict how the environment will change based on these trajectories? You can take inspiration from the previous rules:
<previous hypothesis>
You also know that the previous set of rules failed the most on those trajectories:
<worst trajectories>
Answer with just the hypothesis.

Figure B.3: Prompt used for the Scientist. <trajectories> correspond to the trajectories collected by the Experimenter, <previous hypothesis> corresponds to the set of hypotheses that have been accepted at the last iteration and <worst trajectories> to the trajectories where the previous hypotheses obtained the worst log-likelihood.

Oracles

We show in Figure B.4 the distribution of transitions collected by each of our four oracles.

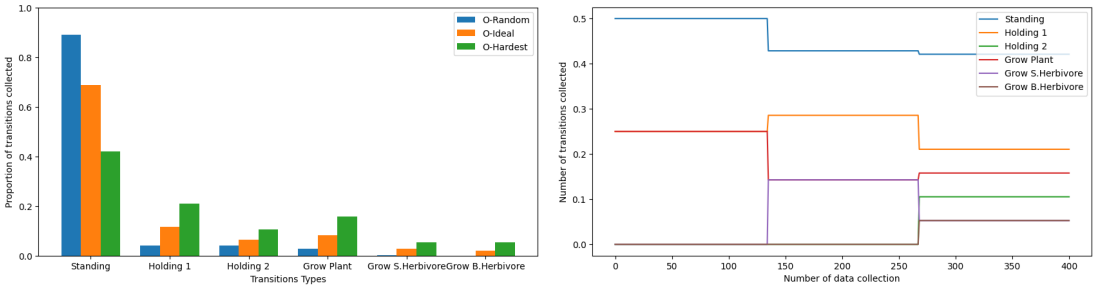


Figure B.4: Proportion of collected transition types for each oracle. We show on the left the distribution of our stationary oracles and on the right the evolving distribution of O-Curriculum.

RL agents

For our RL agents we used Masked PPO from stable-baselines3 contributions[1]. To accelerate training, we used 15 parallel environments and collected 8 trajectories per environment. We show the hyperparameters used in Table B.1.

Concerning the reward signals, we provide in Algorithm 5 the details on how RL-ALPEXP compute its ALP reward. RL-ALPEXP introduces an additional hyperparameter α which regulates the reward's smoothing of the reward. For each transition type, α corresponds to the weight of the previous rewards obtained for this transition type. Its role is highlighted in our study from Appendix B.4.4 but α can be interpreted as an adjustment factor determining the extent to which rewards from previous steps are retained. A value of 0 for α implies that only the reward from the most recent step is considered, which corresponds to RL-ALP, while a value of 1 indicates that the rewards are accumulated from the beginning. After a manual hyperparameter search (see Appendix B.4.4), we chose $\alpha = 0.9$ across all transition types.

hyperparameter	**value**
gamma	0.99
lr	$5e-4$
vf coef	0.5
gae λ	0.95
entropy	0.01
epochs	20
architecture	64×64

Table B.1: PPO hyperparameters used for our RL agents

[1] https://sb3-contrib.readthedocs.io/en/master/modules/ppo_mask.html

Algorithm 5 RL-ALPEXP

Input: α moving average coefficient, $\mathcal{I}$ set of transition types, T total number of iterations, n_{steps} number of Metropolis steps at each iteration, f function returning the type of a transition given a transition, π the Experimenter
Initialize $(m_i)_{i\in\mathcal{I}} \leftarrow 0$ ▷ Initialize moving average for each type
for $i = 0$ to n **do**
  Collect trajectories $\mathcal{D}_t \leftarrow$env(π)
  Perform n_{steps} Metropolis steps
  for $\tau \in \mathcal{D}_t$ **do**
    Compute reward $(r_{alp})_\tau$ ▷ Compute ALP reward using equation 5.2
    $(r_{alpexp})_\tau = \alpha * m_{f(\tau)} + (1-\alpha) * (r_{alp})_\tau$ ▷ Compute ALPEXP reward
  for $i \in \mathcal{I}$ **do**
    $\tilde{m}_\beta = \frac{\sum_{\tau\in\mathcal{D}_t} \mathbf{1}_{f(\tau)=\beta}(r_{alp})_\tau}{\sum_{\tau\in\mathcal{D}_t} \mathbf{1}_{f(\tau)=\beta}}$ ▷ Compute mean over transitions
    $m_\beta = \alpha * m_\beta + (1-\alpha) * \tilde{m}_\beta$ ▷ Update moving average
  Update Experimenter π

B.3 Generalization Tests

We show in Figure B.5 how the syntax of observations was changed in our generalization tests. In particular, we modified the words associated to the transition (e.g. standing, transform) and inverted the sentence. These changes were also applied to the prompts used by the Scientist and the Statistician.

Standing: You are standing on the <*object*>. ⟶ The <*object*> is beneath you.
Holding 1: In your inventory, there is the <*object*>. ⟶ The <*object*> is in your grasp.
Holding 2: In your inventory, there are the <*object*> and the <*object*>. ⟶ The <*object*> and the <*object*> are in your grasp.
Grow Plant: The objects transform into the <*object*>. ⟶ The <*object*> results from combining the objects.
Grow Small Herbivore: The objects transform into the <*object*>. ⟶ The <*object*> results from combining the objects.
Grow Big Herbivore: The objects transform into the <*object*>. ⟶ The <*object*> results from combining the objects.

Figure B.5: Changes in the observations for the generalization environment.

B.4 Additional Results

In this section, we first provide the complete evolution of log-likehoods on our test set from Section 5.5.1. Then, we provide additional studies and experiments to improve one's comprehension of WorldLLM's behavior. These investigations notably explore the reasons behind the failure of RL-ALP and its differences with RL-ALPEXP.

B.4.1 Evolution of likelihoods

We report in Figure B.6 the evolution of log-likelihoods on our test set for all the evaluated methods.

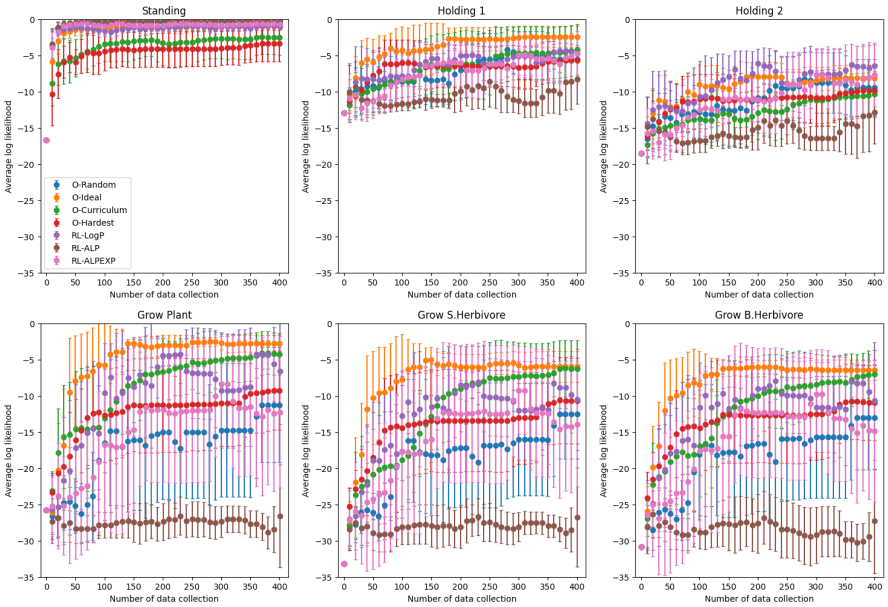


Figure B.6: Evolution of log-likelihoods on the test set for all methods.

B.4.2 Analysis of the Metropolis algorithm in the Scientist

Hypotheses Retention Rate

First, we analyze how the Metropolis algorithm performs when using an LLM as both the proposal and target distribution. In particular, we study the evolution of accepted hypotheses throughout WorldLLM's iterations when using different Experimenters (Figure B.7).

Our results show a high percentage of accepted hypotheses (around 20%) at the beginning of the experiment. However, as iterations progress, it becomes increasingly challenging for the Scientist to generate more effective hypotheses, leading to a decline in the retention rate (it reaches 2%). As highlighted in our manuscript, a developmental strategy is key in WorldLLM. This is here illustrated by O-Curriculum's ability to maintain a higher retention rate than all other oracles. Additionally, preliminary experiments performed with Phi4 suggested that

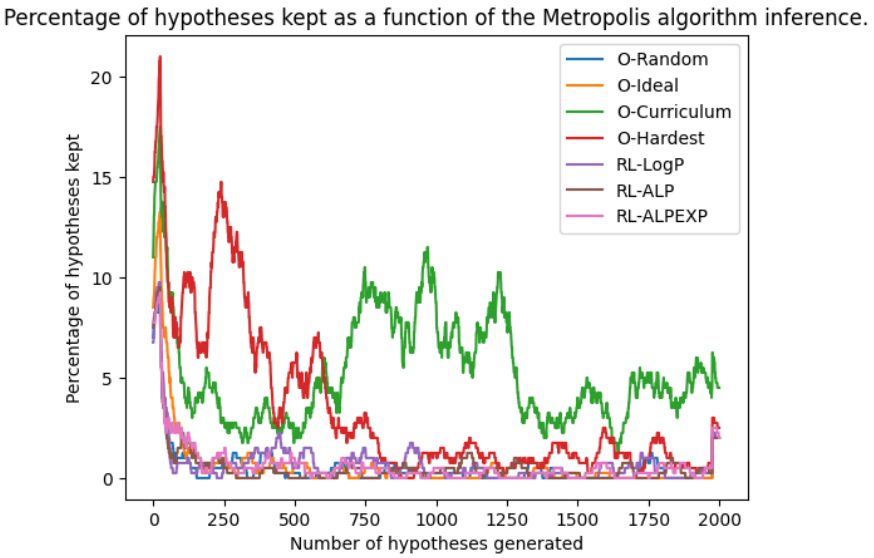


Figure B.7: Evolution of the hypotheses retention rate for the different Experimenters.

employing a larger LLM for the Scientist improves hypotheses generation and leads to higher retention rates for the same Statistician model.

Embeddings of the hypotheses

In order to compare the hypotheses generated by our Scientist, we used a sentence transformer, **all-mpnet-base-v2**[2], to project hypotheses in a latent space. We then used T-SNE(Maaten & Hinton, 2008) to obtain a 2D representation of the embeddings. As the number of generated hypotheses is high, we plotted only 1/10 of them for each Experimenter and seed. We show these embeddings in Figure B.8.

The results show that O-Curriculum and O-Hardest mostly produce similar hypotheses. This indicates a Scientist's tendency to refine previous hypotheses rather than generating brand new ones. This corroborates our results on retention rate from previous section. We also compute an inertia score across all seeds for each Experimenter:

$$\forall \beta \in \mathcal{M}, \quad I = \sum_{i=1}^{n} (x_i - C_\beta)^2 \qquad \text{with} \quad C_\beta = \sum_{i=1}^{n} \frac{x_i}{n} \tag{B.1}$$

with $\mathcal{M}$ the set of Experimenters, x_i the 2D projection from T-SNE for each set of hypotheses and n the total number of hypotheses.

Results indicate that the clusters produced by O-Curriculum and O-Hardest are each produced by a different seed. This highlights the current limitation of our Bayesian Inference approach where a single particle is used and never reset. As a result, our Scientist may easily get stuck in local optima.

[2] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

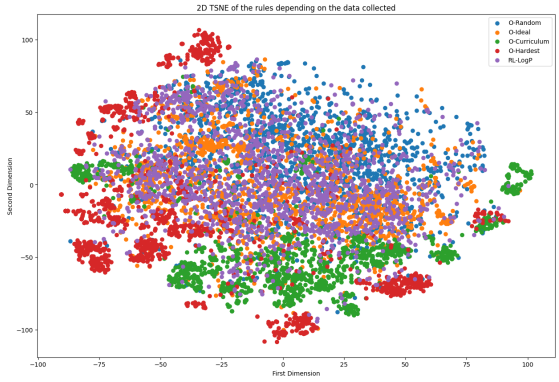


Figure B.8: T-SNE embeddings obtained using the *all-mpnet-base-v2* sentence transformer on the hypotheses generated by different Experimenters.

Methods	Inertia Score
O-Random	4.12e6
O-Ideal	4.38e6
O-Curriculum	2.77e6
O-Hardest	2.88e6
RL-LogP	4.52e6
RL-ALP	3.62e6
RL-ALPEXP	4.92e6

Table B.2: Mean of the inertia score on the T-SNE embeddings across the different seeds for each Experimenter.

B.4.3 Analysis of generated hypotheses

In this section, we propose a qualitative study of hypotheses produced by WorldLLM when using O-Curriculum, O-Ideal and RL-LogP as Experimenters (see Figure B.9).

One can observe that each set of hypotheses covers multiple transition types. This can be explicit, as seen in the hypotheses from O-Ideal, or more abstract, as the ones from O-Curriculum, which replace specific animals and objects with abstract representations. A closer examination of O-Curriculum's hypotheses shows that new entities that do not exist in Playground-Text such as "bear", "mouse" or "wolf" are introduced. This aligns with our argument that natural language hypotheses could help the world model generalize. However, our poor generalization results can be explained by the fact that most hypotheses contain examples of transitions from the environment. Such examples tremendously help our Statistician predict outcomes but also significantly impair its generalization capabilities.

B.4.4 A deeper study RL-ALP

In this section, we provide a deeper analysis of RL-ALP's failure. While expected to be more robust than RL-LogP (e.g. to the "noisy TV" problem), our results showed RL-ALP was unable to collect other transitions than *Standing*, leading to very poor hypotheses. We begin this section by studying how directly using the per-transition ALP as reward for an RL policy is challenging. Then, show how RL-ALPEXP differs and helps stabilizing training.

Failure of RL-ALP as the reward

To get a better grasp of the RL policy's behavior when using RL-ALP, we plot the evolution of rewards obtained by the policy in Figure B.10. We observe that most rewards equal 0. Higher rewards are only obtained in the first data collections when *Standing* transitions are

not well predicted by the Statistician. After gathering such transitions, the policy eventually converges to a random one. This absence of rewards can be explained by our reward definition from Eq. 5.2.

A positive reward is obtained only if a set of hypotheses are accepted between the previous and the current iteration. However, as discussed in Appendix B.4.2, the number of accepted hypotheses tends to quickly drop and eventually reach a very low percentage. As a result, very few improvements (or degradation) are made to the Statistician's log-likelihoods, leading to zero rewards.

Ablation study on RL-ALPEXP

In Appendix B.2, we detailed RL-ALPEXP, which includes a new hyperparameter α. This hyperparameter governs the weighting of smoothing applied to the reward computed on various transition types over time. In this section, we explore the impact of this hyperparameter.

To assess how α influences RL-ALPEXP, we conducted qualitative experiments with α values set to $\{0.6, 0.75, 0.9, 0.95\}$. To gain insights into the underlying behavior, we analyze the transitions collected by the policy. The results can be found in Figure B.11.

From these results, we observe two behaviors depending on the values of α.

On one hand, when α is smaller than 0.9, the RL policy demonstrates a periodic behavior. This phenomenon becomes more pronounced as α decreases, leading to instability that hinders WorldLLM's ability to improve consistently.

It seems that the reward diminishes to zero before the Scientist and Statistician can refine the existing hypothesis. This decline ultimately leads to a collapse in the RL agent's policy. As a result, the distribution of the collected data begins to resemble that of a random agent, as in the first iterations of WorldLLM. In response, the Scientist generates hypotheses that cause the framework to revert to an earlier stage.

On the other hand, as α gets closer to 1, the reward takes increasingly longer to fade. This can be problematic as the reward function used is an estimation of the learning progress. For example, the high rewards collected at the beginning of training are no longer relevant after multiple iterations.

This hyperparameter search explicitly shows that RL-ALPEXP has not fully resolved the sparse reward problem inherent to RL-ALP. If α is too small, the agent's policy collapses to random. If it is too close to 1, the old rewards obtained require too much time to fade. One potential solution could be to add a small fraction of RL-LogP's rewards. This addition would prevent the agent's policy from collapsing but would require a new hyperparameter to balance the reward from RL-ALP and RL-LogP.

O-Curriculum:
1. If the action is 'You go to' and the object is a seed or gentle creature (cow, elephant, sheep, giraffe, bear, mouse, wolf), predict "You are standing on the [object]."
2. If you are standing on an animal's seed and the action is 'You give the water', predict "The objects transform into the [animal]."
3. If the object is a seed or gentle creature (cow, elephant, sheep, giraffe, bear, mouse, or wolf) and the action is 'You give the water', predict "The objects transform into the [animal]."
4. If you are standing on the water and the action is 'You pick up', predict "In your inventory, there is the water."
5. If you are standing on the water and the action is 'You pick up the object', predict "In your inventory'
O-Ideal:
1. The action was: You go to the water, and the change was: You are standing on the water.
2. The action was: You pick up the object, and the change was: In your inventory, there is the water.
3. The action was: You go to the beet seed, and the change was: You are standing on the beet seed.
4. The action was: You give the water, and the change was: The objects transform into the beet.
5. The action was: You pick up the object, and the change was: In your inventory, there is the beet.
6. The action was: You go to the baby cow, and the change was: You are standing on the baby cow.
7. The action was: You give the beet, and the change was: The objects transform into the cow.
8. The action was: You go to the pe'
RL-LogP:
1. Go to the water.
Change: You are standing on the water.
2. Pick up the object.
Change: In your inventory, there is the water.
3. Go to the water.
Change: You are standing on the water.
4. Pick up the object.
Change: In your inventory, there are the water and the water.
5. Go to the berry seed.
Change: You are standing on the berry seed.
6. Give the water.
Change: The objects transform into the berry.
7. Go to the water.
Change: You are standing on the water.
8. Pick up the object.
Change: In your inventory, there are the water and the water.
9. Go to the pea seed.
Change: You are standing on the pea seed.
10. Give the water.
Change: The objects transform into

Figure B.9: Examples of accepted hypotheses.

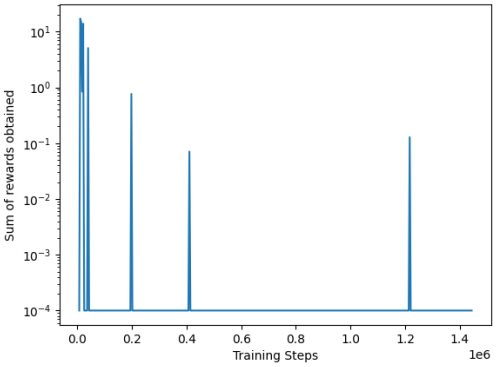


Figure B.10: The evolution of rewards obtained by the RL agent when using RL-ALP for a single seed. We plot the sum of rewards obtained for each PPO update.

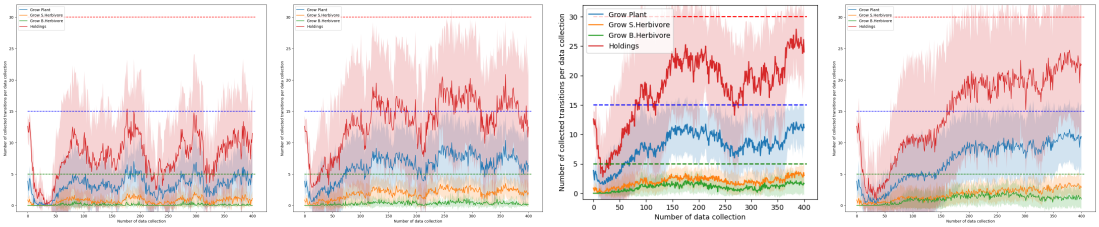


Figure B.11: Transitions collected by the Experimenter when using RL-ALPEXP for $\alpha = \{0.6, 0.75, 0.9, 0.95\}$ respectively.

Appendix C

Appendix of MAGELLAN

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

- Section C.1 details our Little-Zoo environment.
  - Section C.1.1 explains the environment mechanics and how the different actions affect the environment. It also gives more details about optimal trajectories.
  - Section C.1.2 details the goal space generation.
  - Section C.1.3 explains the different types of impossible goals. This section also gives examples for each type of impossible goal.
  - Section C.1.4 contains the goal space's distribution per goal type.
- Section C.2 extends the table from Section 8.4, comparing the previous LP estimation methods.
- Section C.3 contains the implementation details of MAGELLAN and the baselines.
  - Section C.3.1 explains the LLM-based RL agent used in our experiments, the associated hyperparameters and the prompt used.
  - Section C.3.2 details MAGELLAN architecture, hyperparameters and the prompt used.
  - Section C.3.3 contains the additional details for implementing the baselines.
  - Section C.3.4 is about the compute budget.
- Section C.4 extends the results given in the Section 4.5 on the experiments of the paper.
  - Section C.4.1 is an ablation on MAGELLAN's architecture.
  - Section C.4.2 develops the results given in Section 8.5.2 with plots for goal spaces of different size.
  - Section C.4.3 contains several additional results for Section 8.5.3.
    - Section C.4.3 gives the evolution of the success rate per method and per goal type.

- Section C.4.4 develops several aspects of MAGELLAN's generalization abilities from Section 8.5.4.
  - Section C.4.4 specifically examines the ability to generalize success probability estimations to test goals across different goal types.
  - Section C.4.4 describes the embedding evolution during training.
- Section C.4.5 expends on the results found in Section 8.5.5.
  - Section C.4.5 contains all the curves from the experiment about the adaptation to an evolving goal space.
  - Section C.4.5 provides a quantitative analysis of this experiment.

C.1 Little-Zoo environment

C.1.1 Environment mechanics

There are 5 categories of elements the agent can interact with: the objects (i.e. "bookshelf"), the water, the plants (i.e. tomato) which can bee `seed` and `grown`, the herbivores (i.e cow) which can be `baby` or `grown`, and the carnivores (i.e. lion) which can be `baby` or `grown`.

To interact with any element of the environment, multiple actions are accessible. First, the agent can use `"Go to {element}"` to move to any element. You can only move to an element (not a random point in the plane) and after a `"Go to"` action you are placed on the element. When performing the `"Grasp"` action, the element the agent is standing on is added to its inventory (up to 2 items). Some elements can interact with each other. To make two elements interact, the agent must release one element in its inventory by using the `"Release {element}"` action while standing on the other element. Interactions between three elements are also possible: the agent has to hold two elements in its inventory and perform `"Release all"` while standing on the third element. The "Release" action is only accessible if an interaction can be made between the element to release and the one the agent is standing on. Therefore, the agent must carefully manage its inventory to avoid filling it with useless elements without being able to release them. At each turn you get a full description of the environment with `"You see {element_1}, {element_2}, {element_3}, {element_4}"`.

To grow a plant, an herbivore or a carnivore, the agent needs to follow a certain sequence of actions:

- To `"Grow a plant"`, the agent has to `"Release water"` while `"Standing"` on the plant seed.
- To `"Grow an herbivore"`, the agent has to `"Release a grown plant"` while standing on a baby herbivore.
- To `"Grow a carnivore"`, the agent has to `"Release a grown herbivore"` while standing on a baby carnivore.

Table C.1 summarizes the optimal action strategy for each category. In order to maintain a coherent level of difficulty between the different categories, we fix the maximal number of actions allowed to the agent to solve a goal as 150% of the minimal number of actions required to solve the goal. The required exploration to discover strategies has, therefore, the same level of difficulty for all goals.

C.1.2 Goal space generation

To define our goal space, we start from the sets of objects $\mathcal{O}$, plants $\mathcal{P}$, herbivores $\mathcal{H}$, carnivores $\mathcal{C}$ and the element water (each set being of size 6). One goal g in the goal space is defined using 5 elements $(E_0, E_1, E_2, E_3, E_4) \in \{\mathcal{O}, \mathcal{P}, \mathcal{H}, \mathcal{C}, \text{water}\}$ and by using the objective function $\mathcal{F}_g$ from the set $\{$`Grasp{X}`, `Grow{X}`$\}$ on E_0. Thus, we get the goal $g = (\mathcal{F}_g(E_0), E_1, E_2, E_3, E_4)$. As each set contains 6 elements, we get a total of $19,531,250$ goals. The vast majority of these goals are impossible for various reasons (see Appendix C.1.3).

Table C.1: The optimal trajectories per categories.

Categories	Optimal action strategy	Minimal number of actions	Maximal number of actions
Grasp X	Go to X, Grasp	2	3
Grow plant	Go to water, Grasp, Go to plant, Release water	4	6
Grow herbivore	Go to water, Grasp, Go to plant, Release water, Grasp, Go to herbivore, release plant	7	11
Grow carnivore	Go to water, Grasp, Go to plant, Release water, Grasp, Go to herbivore, release plant, Grasp, Go to carnivore, release	10	15

In our experiments, we subsample two goal spaces: a train and held-out test space. To generate them, we sample the total number of impossible goals n_{impo} we want to have, then sample among the possible goals: $\frac{n_{impo}}{5}$ goals where $E_0 \in \mathcal{O}$, $\frac{n_{impo}}{5^2}$ goals where $E_0 \in \mathcal{P}$, $\frac{n_{impo}}{5^3}$ goals where $E_0 \in \mathcal{H}$, and $\frac{n_{impo}}{5^4}$ goals where $E_0 \in \mathcal{C}$. Appendix C.1.4 gives more details on the goal repartition.

C.1.3 Example of impossible goals

In our environment, a goal is defined as a combination of elements present in the environment and an instruction, such as "grasp the rabbit". Some combinations are incompatible such as:
`Grasp rabbit`
`You see: bookshelf, baby lion, desk, baby cow`,
where there is no rabbit. The generation of our environment leads to having 80% of impossible goals in the goal space, which is similar to any complex environment Zhang et al. (2024b); Matthews et al. (2024b). However, conversely to Zhang et al. (2024b), identifying impossible goals and avoiding to select them is not trivial, as there are multiple reasons why a goal is impossible.

The easiest reason for impossibility is the absence of the element the agent has to act on:

- Absence of the element to grasp
  `Goal: Grasp bed`
  `You see: bookshelf, baby lion, desk, baby cow.`
- Absence of the element to grow
  `Goal: Grow deer`
  `You see: baby giraffe, bookshelf, water, tomato seed.`

Another reason is that the element cannot follow the dynamic required in the objective:

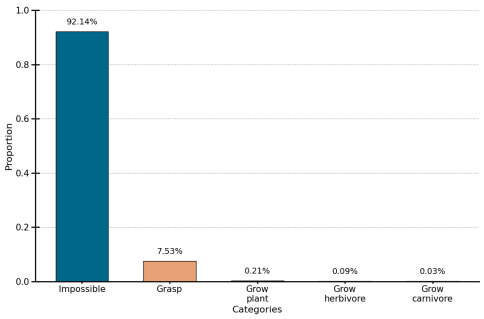
- Growing an element that is neither an animal nor a plant
  `Goal: Grow desk`
  `You see: bed, baby bobcat, baby elephant, door.`

A more difficult reason that requires a better understanding of the underlying mechanisms of the environment is the lack of at least one element necessary to reach the objective grow:

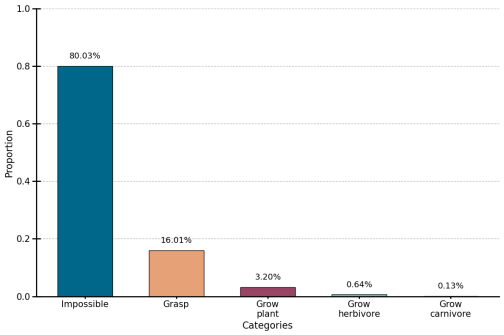
- The lack of water:
  `Goal: Grow cucumber`
  `You see: cucumber seed, carrot seed, baby coyote, baby wolf,`
  `Goal: Grow coyote`
  `You see: cucumber seed, berry seed, baby coyote, baby cow.`
- The lack of a plant preventing the agent to grow an herbivore resulting in the impossibility to grow a carnivore:
  `Goal: Grow coyote`
  `You see: water, baby elephant, baby coyote, baby cow.`

Consequently, to efficiently build a curriculum, being able to understand why a goals is impossible and quickly generalize to infer the other impossible goals is crucial.

C.1.4 Goal repartition



(a) Goal distribution per categories on the full goal space.



(b) Goal distribution per categories in our experiments.

Figure C.1: We first generate the full goal space whom the distribution is given Figure C.1a then for computational reasons we sample a smaller goal space with the following distribution Figure C.1b.

At the end of the goal space generation, we have 19531250 goals distributed as presented in Figure C.1a. However, using all the goals was computationally impossible considering all the comparisons we made. Moreover, the repartition with 92% of impossible goals and 7.53% of "Grasp" goals made the epsilon-greedy exploration of our goal selector too slow to discover LP niches in an acceptable compute budget. Subsequently, we sample a training goal space and a testing goal space following the distribution given in Figure C.1b. Both goal spaces are composed at 80% of impossible goals, the remaining goal categories becoming less and less represented as the difficulty increases. This repartition synthetically simulates how natural

language goals mostly lead to sequences of words without any meaning. This structure makes the strategy of uniformly sampling goals inefficient and underlines the necessity of a curriculum to be able to reach the hardest goals in the allocated training budget (500k episodes in our experiments).

C.2 Comparison of LP methods

We compare prior work computing LP for automatic curriculum learning under the dimensions from Section C.4.2. We show the comparison in Table C.2 evaluating methods:

- **Efficiency**: Computational cost of additional evaluation episodes not used to train the policy.
- **Competence transfer tracking**: How well does the method track all the possible competence transfer.
- **No expert knowledge required**: if they require any external expert knowledge such as pre-defined goal groupings.

We consider a method's efficiency as "high" if it does not require any additional evaluation (i.e. it only uses the performance observed on goals sampled), and as "low" otherwise. We evaluate the competence transfer tracking using the following criteria:

- absence of +: the estimated competence is updated only on sampled goals.
- +: the estimated competence is updated on a predefined goal subset the sampled goal belongs to.
- ++: the estimated competence is updated on a dynamically learned goal subset the sampled goal belongs to.
- +++: the estimated competence is updated on all goals.

Methods	Efficiency	Competence transfer tracking	No expert knowledge required
Oudeyer et al. (2007)	high	++	✓
Baranes & Oudeyer (2009)	high	++	✓
Baranes & Oudeyer (2013)	high	++	✓
Moulin-Frier & Oudeyer (2013)	high	++	✓
Moulin-Frier et al. (2014)	high	++	✓
Portelas et al. (2020b)	high	++	✓
Forestier & Oudeyer (2016)	high	++	✓
Kovač et al. (2023)	high	++	✓
Stout & Barto (2010b)	high	+	×
Lopes & Oudeyer (2012b)	high	+	×
Matiisen et al. (2020)	high	+	×
Fournier et al. (2018)	high	+	×
Colas et al. (2019)	high	+	×
Blaes et al. (2019)	high	+	×
Akakzia et al. (2021)	high	+	×
Kumar et al. (2024)	high	+	×
Kanitscheider et al. (2021)	low	+++	✓
Zhang et al. (2024b)	low	+++	✓

Table C.2: Comparison of prior work w.r.t. their LP estimation approach. We use the following dimensions: computational efficiency, dynamical competence transfer assumptions and no required expert knowledge.

C.3 Implementation details

To facilitate reproduction and future work, we provide our code at `https://github.com/flowersteam/MAGELLAN`.

C.3.1 LLM-based RL agent

We use the SAC-GLAM method from Gaven et al. (2024) with the hyperparameters listed in Table C.3. Following SAC-GLAM, the Q-value head is a two-layer MLP with 1024 ReLU-activated units, applied to the last hidden state of the decoder. The policy and the critic share the same LoRA adapters. Additionally, we apply a warm-up phase of 10 updates, during which only the Q-function is trained.

The prompt used as the LLM agent's observation is shown in Prompt C.3.1.

Prompt C.1: RL Agent observation

Goal: Grow lion
You see: water, carrot seed, baby lion, baby cow
You are standing on: nothing
Inventory (0/2): empty
Action:

Table C.3: SAC hyperparameters

Variable	Value
Update frequency	64
Number of updates	1
Batch size	256
Discount factor	0.99
Optimizer	Adam
Critic learning rate	1×10^{-4}
Actor learning rate	1×10^{-4}
Entropy coefficient	auto
Entropy coefficient initialization	0.05
Target entropy	-0.125
Entropy coefficient learning rate	1×10^{-3}
n-step	3
Replay buffer capacity	500000

C.3.2 MAGELLAN

Our competence estimator uses a single-layer MLP with 128 units and Tanh activations, applied to the last hidden state of the LLM. It is updated every 32 policy updates, with batch sampling that prioritizes recent data. The sampling distribution is determined by $\frac{i}{\sum_{j=1}^{M} j}$, where i represents the position of a data point in the buffer $\mathcal{D}$ (with 1 being the oldest and M the buffer size). This strategy ensures that the competence predictor remains responsive while preserving batch diversity, incorporating data from different versions of the learning agent to help mitigate noise introduced by the fluctuations in the RL agent's learning process.

To compute the LP, we store the weights of our competence estimator in the buffer $\mathcal{B}$. This is done by saving the LoRA adapters and the MLP weights whenever the competence estimator is updated. The window for LP computation is defined by $|\mathcal{B}| \times$ *update frequency*. The LP is calculated by comparing the competence estimation obtained with the oldest and most recent weights in the buffer.

The MAGELLAN hyperparameters are provided in Table C.4.

The prompt given to the competence estimator is the same as the prompt given to the RL agent Prompt C.3.2.

Prompt C.2: MAGELLAN prompt

Goal: Grow lion
You see: water, carrot seed, baby lion, baby cow
You are standing on: nothing
Inventory (0/2): empty
Action:

Table C.4: MAGELLAN hyperparameters

Variable	Value
ϵ start	1
ϵ end	0.2
ϵ decay period	320
$\mathcal{B}$ size	100
$\mathcal{D}$ size	5000
Batch size	256
Optimizer	Adam
Learning rate	1×10^{-4}
Update frequency	32

C.3.3 Baselines

We compared our method against four baselines: Online-ALP, Eval-ALP, EK-Online-ALP, and EK-Eval-ALP. Below, we outline the implementation details for the different families of methods:

- **Non-EK methods** For methods that do not rely on expert knowledge, competence and LP estimations are computed individually for each goal.
- **EK methods** For methods incorporating expert knowledge, goals are grouped into five buckets, with competence and LP computed at the bucket level.
- **Online methods** In online methods, a buffer of size M stores goal-outcome pairs. The agent's current competence is estimated from the most recent half of the buffer, while past competence is derived from the oldest half. Non-EK methods maintain a separate buffer for each goal, whereas EK methods use a buffer for each bucket.
- **Eval methods** In evaluation-based methods, the agent's competence is assessed every N episodes. For non-EK methods, the agent is evaluated k times on each individual goal, while for EK methods, evaluations are performed k times per bucket.

The hyperparameters for Online methods are presented in Table C.5, while those for Eval methods are shown in Table C.6.

Table C.5: Online methods hyperparameters

Variable	Value
ϵ start	1
ϵ end	0.2
ϵ decay period	320
Buffer size	100

Table C.6: Eval methods hyperparameters

Variable	Value
ϵ start	1
ϵ end	0.2
ϵ decay period	320
Eval frequency	1000
Number of eval	2048

C.3.4 Compute budget

To optimize GPU VRAM usage during training, we employ 4-bit quantization techniques as described in (Dettmers et al., 2023). We use a vectorized version of Little-Zoo with 32 instances of the environment running (synchronously) in parallel. In order to accelerate training of our LLM agent (both its policy with online RL and MAGELLAN), we leverage Lamorel[1] to deploy 2 instances of the LLM in parallel. Thus, we distribute both the forward passes to compute actions' probability or LP and training in a Data Parallelism setting. When using Flan-T5 250M, each LLM instance is distributed (Vertical Model Parallelism) over one Nvidia H100 80GB GPUs requiring thus a total of 2 Nvidia H100 80GB GPUs to run an experiment (1 GPU $\times$ 2 LLM instances). For one seed of one ALP method trained in Section 8.5.3, performing 500k training episodes requires 80 GPU hours on the Nvidia H100 80GB.

[1] https://github.com/flowersteam/lamorel

C.4 Additional results

C.4.1 Ablations on the MAGELLAN architecture

In order to implement MAGELLAN, we try various architectures presented in Figure C.2. In all of them, to reduce the computational cost, we froze the LLM and used LoRA adapters **?** for training the policy (along with its Q-value) and the SR estimator. The architectures tested are:

- **A: Different adapters for the policy and MAGELLAN** (Figure C.2a). This is the architecture adopted in the paper. The environment representation for the policy and the goal space representation for the competence estimator are learned separately on different adapters.
- **B: Shared adapters for the policy and MAGELLAN**(Figure C.2b). We test whether a shared useful representation between the policy and competence estimator can emerge through training.
- **C: Shared adapters but only trained with the policy loss**(Figure C.2c). We use a single set of LoRA adapters to train both the policy and the competence estimation module, but apply the latter's gradient only to the MLP outputting the competence. This ablation tests if the policy loss on its own creates semantical relationships between goals. We argue that training of the policy should push towards a clustering of the goal space.
- **D: Adapter only for the policy, frozen LLM representation for MAGELLAN**(Figure C.2d). In this ablation, we test if MAGELLAN can learn to predict competence using the LLM's original latent space. Learning to estimate competence over a fixed representation is closer to prior works such as ALP-GMM Portelas et al. (2020b).

We compare the learning dynamics of the four architectures in Figure C.3. We observe no difference in their ability at learning "grasp" goals. However, the agents with Architecture A continue to progress on other goal types whereas the ones using Architectures B, C, and D stagnate between 50k and 100k episodes before progressing again. At the end of training (500k episodes), agents using architecture A have learned all goal types. The ones based on Architectures B and C learned up to "grow herbivore" goal types. The agents that utilize Architecture D succeed plateaued on "grow plant".

To gain a finer comprehension of the learning dynamics implied by the different architectures, we look in Figure C.4 at the embeddings at the end of training for each architecture.

Architecture A (Figure C.4a) and architecture B (Figure C.4b) shared very similar representations with in both cases goals clustered between "grasp", "grow" and impossible goals. Inside the cluster "grow" the different element types (i.e. plants, herbivores, carnivores) are also clustered. Nonetheless, as seen in Figure C.3, Architecture B does not master "grow carnivore" goals and classified them as impossible. This indicates that obtaining a shared latent space for the policy and MAGELLAN is possible but slows down skill acquisition.

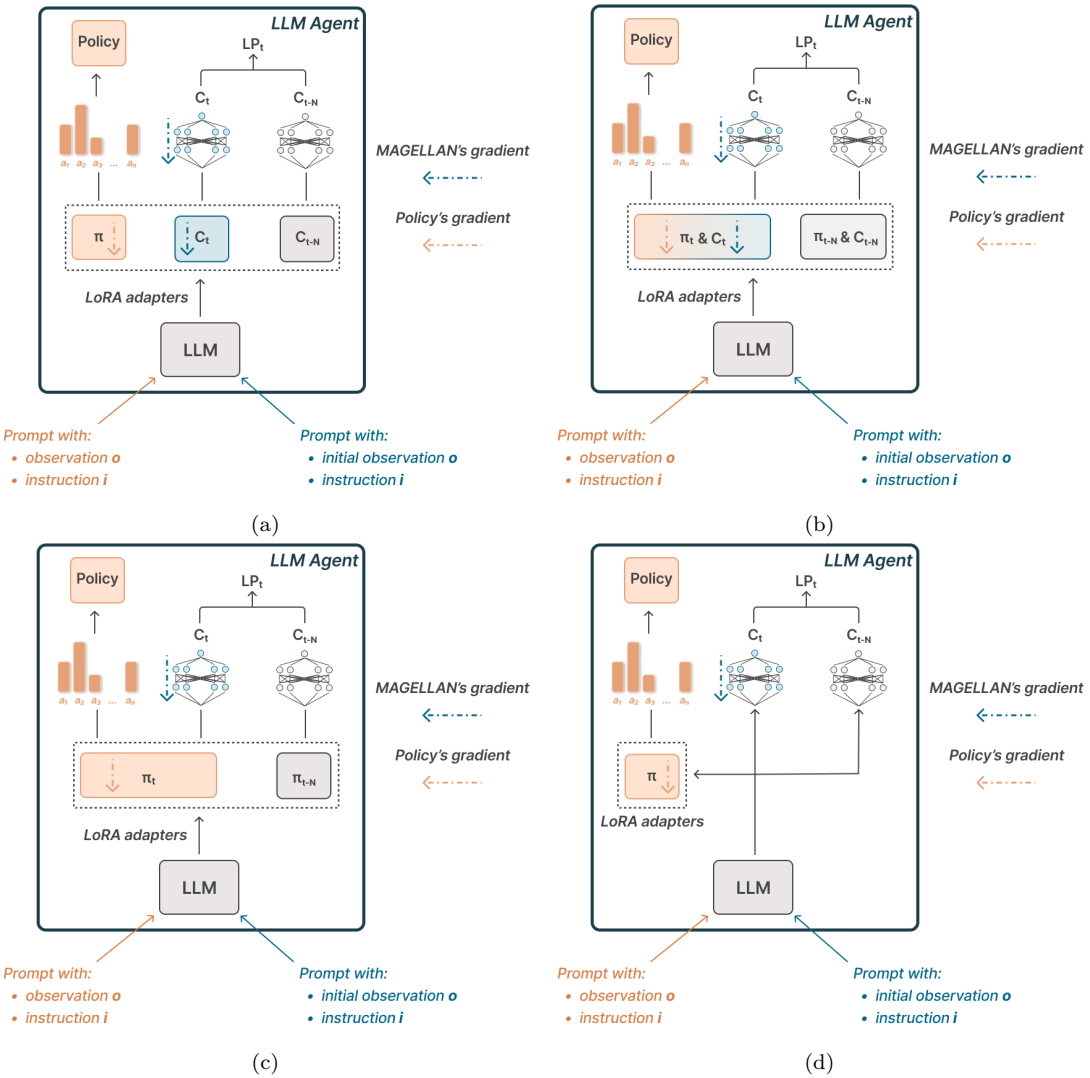


Figure C.2: Different architectural choices in MAGELLAN: (a) we learn separate LoRA adapters between the policy and MAGELLAN (used in the paper); (b) we share adapters and update them using both the policy and MAGELLAN gradient; (c) we share adapters but they are only updated by the policy gradient; (d) MAGELLAN directly uses the latent representation produced by the pretrained LLM.

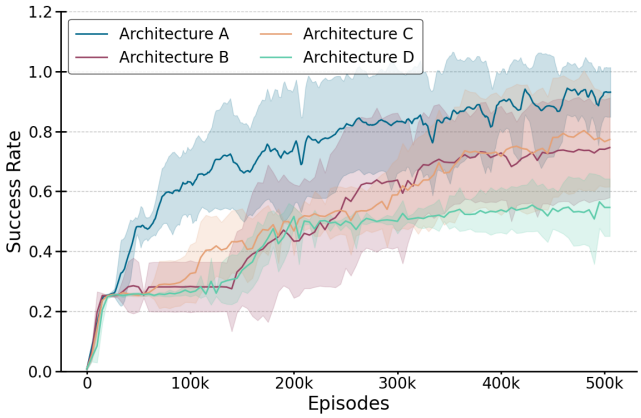
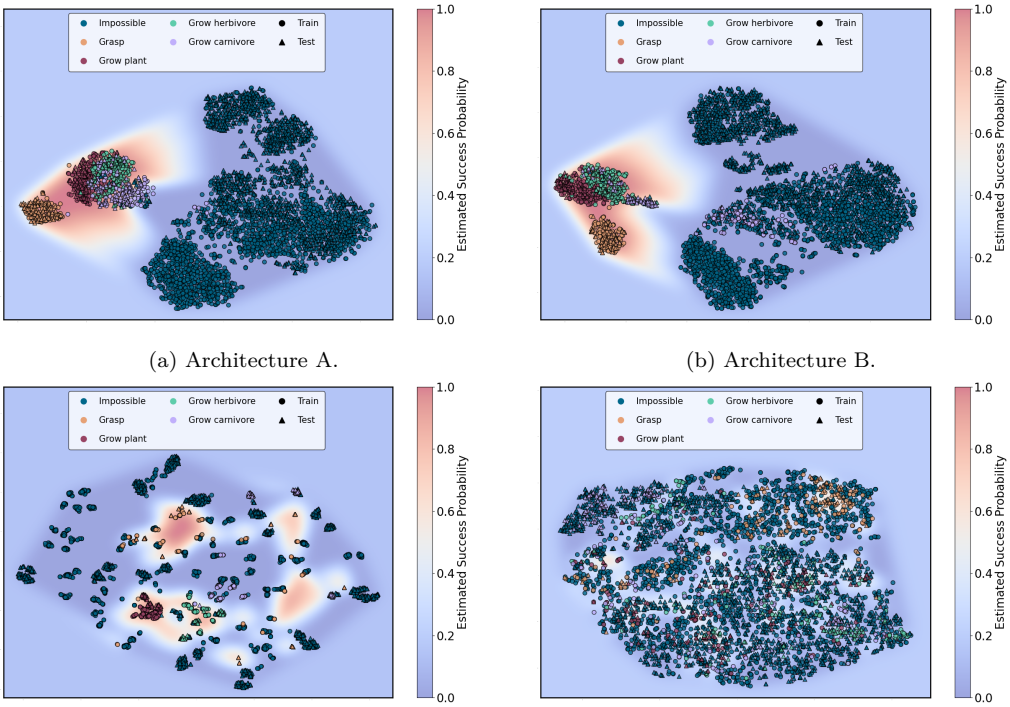


Figure C.3: Training curves of the four different possible architecture for MAGELLAN. We use 8 seeds to plot the mean and the standard deviation (shadow area around the solid line).

Architecture C (Figure C.4c) where MAGELLAN uses the representation solely changed by the policy still manages to accurately estimate competence. However, predicting competence is much more difficult using this architecture as the goal space is not modified ease competence estimation. Yet, using only the policy gradient still improves the initial latent space of the LLM as shown by Architecture D (Figure C.4d).



(a) Architecture A.

(b) Architecture B.

(c) Architecture C.

(d) Architecture D.

Figure C.4: The LLM embedding space of MAGELLAN displayed using t-SNE with goals used in Q2 (Train) and Q3 (Test), along with MAGELLAN's estimated success probability and linear interpolation between goals. We show the embedding space for a single seed for the four architectures described in Appendix C.4.1 at the end of the 500k training episodes.

C.4.2 Q1. Competence estimation properties

Per-goal competence estimation

In this section, we report the evolution of the per-goal competence estimation of experiments from Section 8.5.2. We show it separately for the three goal space size: 25k (Figure C.5), 50k (Figure C.6) and 100k (Figure C.7).

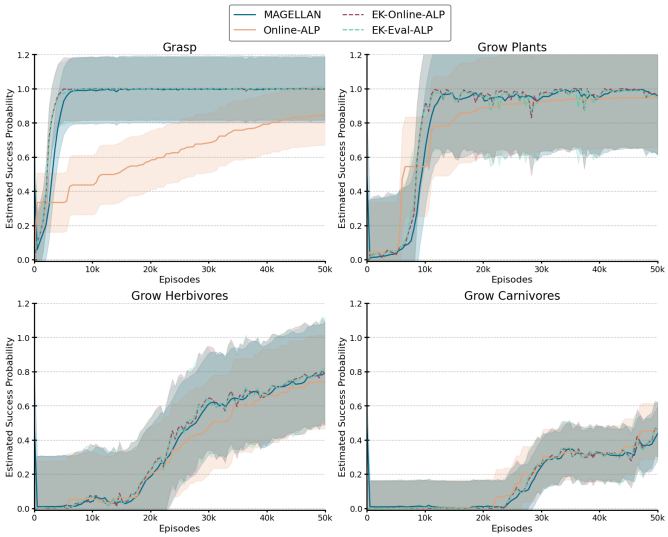


Figure C.5: Evolution of competence estimation for each ALP method on each goal category for 25k goals. We show the average competence and its standard deviation across 8 seeds that use EK-Eval-ALP to sample goals.

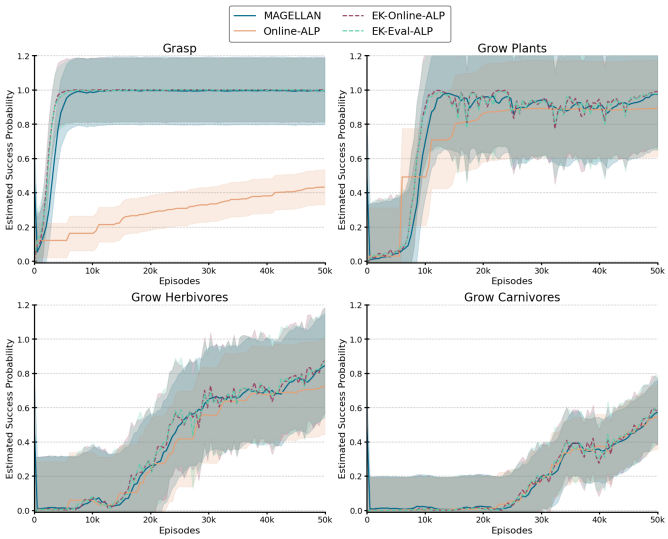


Figure C.6: Evolution of competence estimation for each ALP method on each goal category for 50k goals. We show the average competence and its standard deviation across 8 seeds that use EK-Eval-ALP to sample goals.

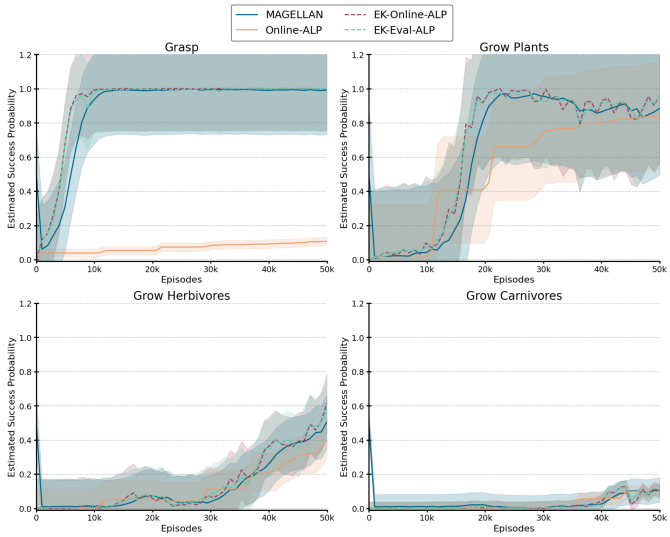


Figure C.7: Evolution of competence estimation for each ALP method on each goal category for 100k goals. We show the average competence and its standard deviation across 8 seeds that use EK-Eval-ALP to sample goals.

Competence estimation on the BabyAI-text goal-space

We replicated the experiment from Section 8.5.2, simulating a learning agent and estimating its competence online using MAGELLAN and Online-ALP. In this setting, the goal space is drawn from the BabyAI-Text environment, consisting of five goal categories of increasing difficulty: Go to *"object"*, Pick up *"object"*, Open *"door"*, Put *"object A"* next to *"object B"*, and Pick up *"object A"* then go to *"object B"*. As shown in Figure C.8, MAGELLAN outperforms Online-ALP, successfully tracking the agent's competence progression across the different task categories.

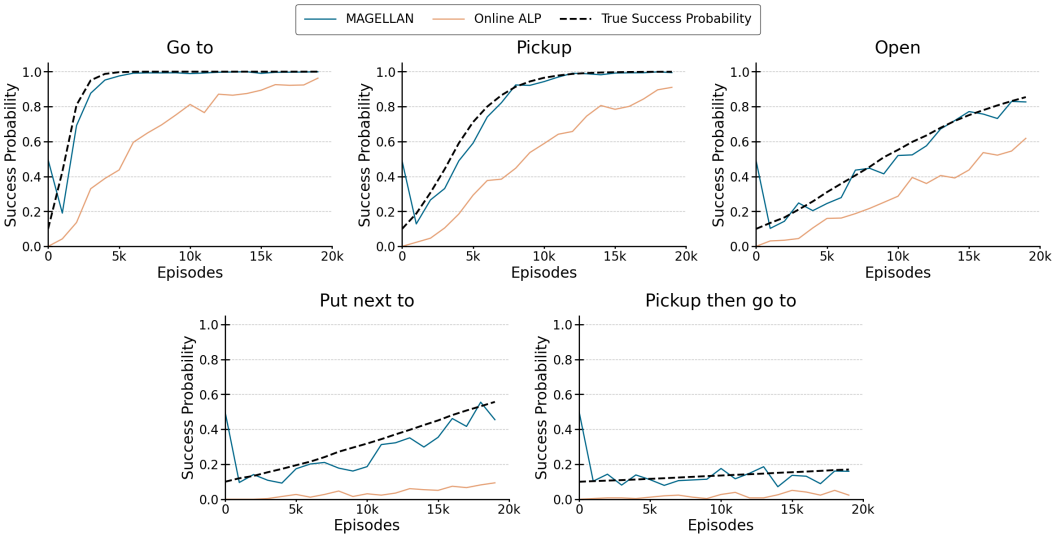


Figure C.8: Competence estimation on BabyAI-Text. MAGELLAN accurately tracks competence across five goal types of increasing difficulty and consistently outperforms Online-ALP.

Impact of the LLM used in MAGELLAN

In this section, we compare different LLMs (Flan-T5-small, Flan-T5-base, and Qwen2.5-0.5B (Qwen et al., 2025)) to evaluate their impact on the competence estimation performance of MAGELLAN. The experiments simulate agent learning using two goal spaces: OpenR1-Math-220k and BabyAI-Text (Figures C.9 and C.10). Results show that all tested LLMs achieve similar performance, suggesting that MAGELLAN's competence estimation remains robust across a range of lightweight language models.

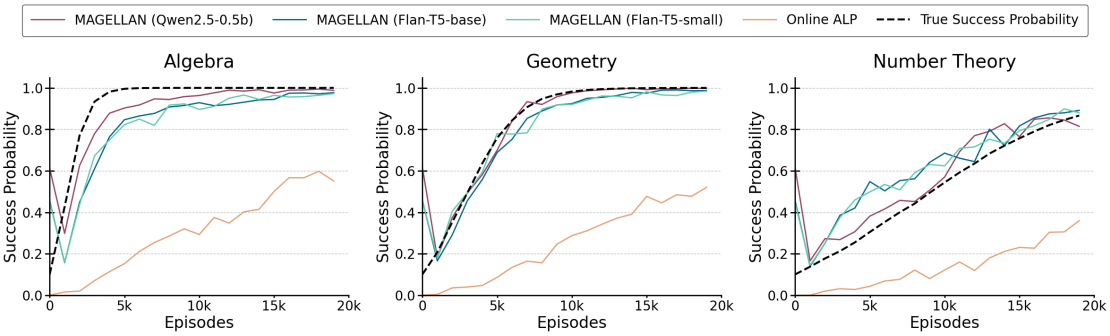


Figure C.9: Effect of LLM choice on MAGELLAN's competence estimation for OpenR1-Math-220k. All models yield similar performance.

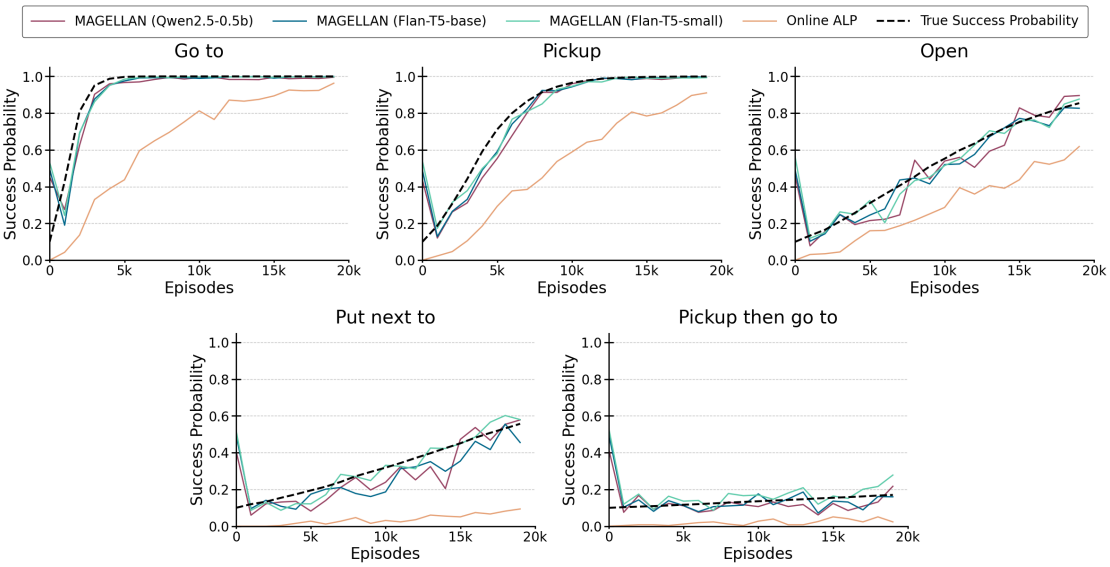


Figure C.10: Effect of LLM choice on MAGELLAN's competence estimation for BabyAI-Text. Performance remains consistent across models.

C.4.3 Q2. Training an LLM-based RL agent with MAGELLAN

Per-goal Success Rate

We detail in Figure C.11 the evolution of the success rate per method and per goal type. All methods perform similarly on the type "Grasp". However, MAGELLAN outperforms all the baselines that do not rely on expert knowledge (Uniform and Online-ALP) on all other types. Learning dynamics fostered by MAGELLAN are close to the ones generated with EK-Online-ALP which relies on expert knowledge. Such results hint towards a clustering of the goal space by MAGELLAN to efficiently and reliably estimate the SR of the LLM agent. This hypothesis is strengthened by the plot of the embedding space in Appendix C.4.4.

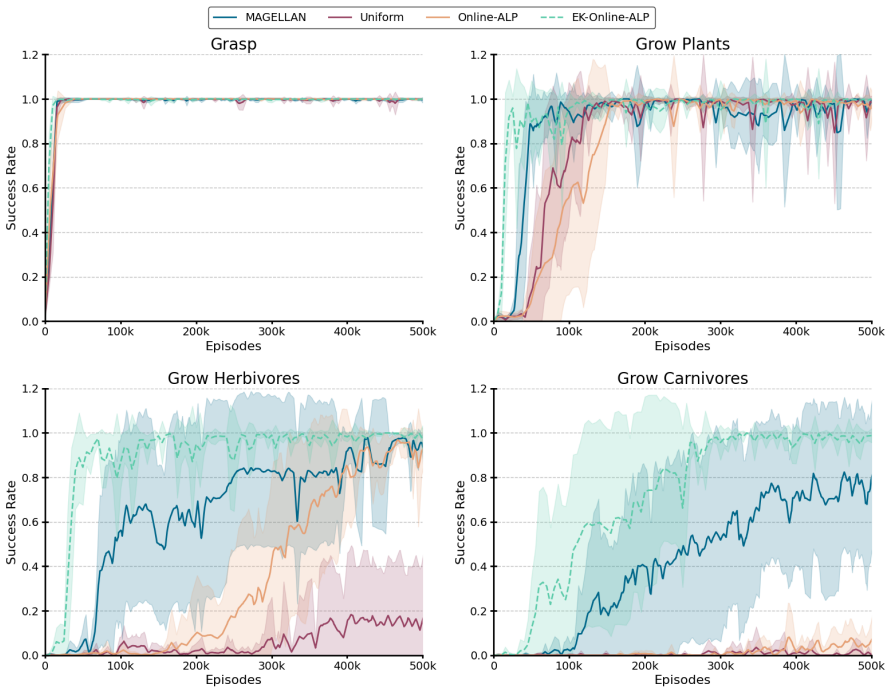
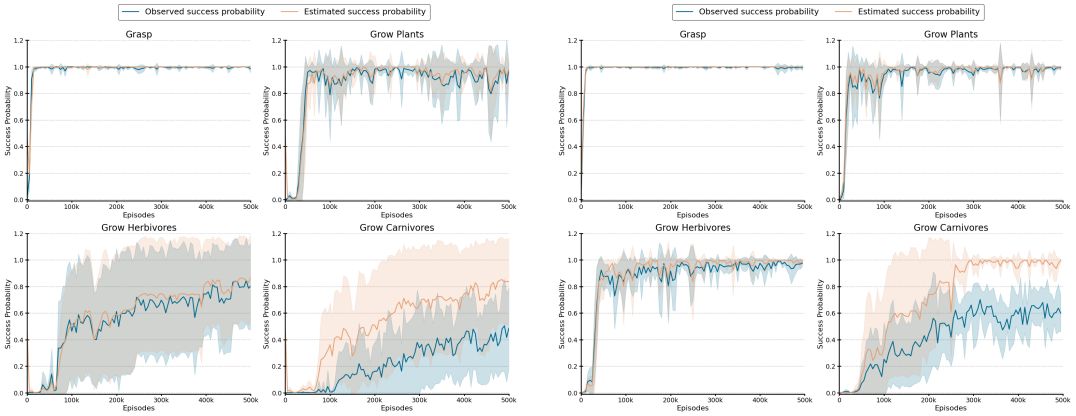


Figure C.11: Evolution of average SR for each ALP method for each goal category. To get the success rate within a category, we evaluate the policy on 64 goals of this category. We use 8 seeds to plot the mean and the standard deviation.

C.4.4 Q3. MAGELLAN's generalization abilities

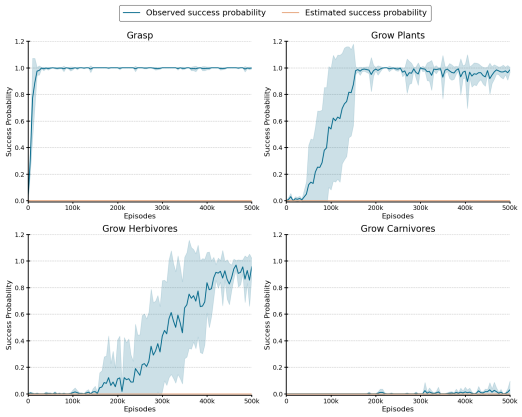
Per-goal Success Probability estimation on test set

In this section, we provide a detailed analysis of the results presented in Section 8.5.4, specifically examining the ability to generalize success probability estimations to test goals across different goal types (see Figure C.12). As expected, Online-ALP exhibits the largest errors, as it can only assign a success probability of 0 to unseen goals. In contrast, both MAGELLAN and EK-Online-ALP achieve highly accurate estimations for "Grasp", "Grow plant", and "Grow herbivore". However, both methods tend to overestimate the generalization abilities of the policy on "Grow carnivore" goals.



(a) MAGELLAN.

(b) EK-Online-ALP.



(c) Online-ALP.

Figure C.12: Per-goal estimation of the success probability for each method. The average result (over 8 seed) is the solid line and the shaded zone represents the standard deviation.

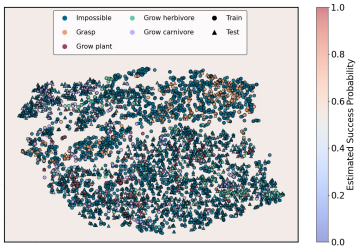
Evolution of the embeddings

In Section 8.5.4, we present the embedding at the beginning and the end of the training for the seed 0.[2] In this section, we explore how the goal space is dynamically modified throughout training with Figure C.13. It is a chronogram of the embedding space used by MAGELLAN projected using t-SNE van der Maaten & Hinton (2008) with the estimated success probability over the whole space calculated using linear interpolation with a Gaussian filter. Each embedding (except the first one that is the initial state) is plotted after the agent masters a goal category.

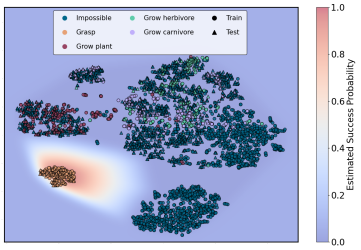
- At the beginning, Figure C.13a, no structure is discernible in the goal space and the estimated success probability is uniform around 0.5.
- After mastering the goals of type "Grasp", Figure C.13b shows several clustered appeared. All "Grasp" goals are in the same cluster with a high estimated success probability zone. The goals of type "Grow plant" are also clustered together and close to the zone of high estimated success probability. That is a hint that MAGELLAN has already picked them as the next candidate type for the curriculum. It also correctly places the "Grow plant" goals from the test set in the same cluster as the ones from the train set. However, it still mixes them with impossible goals, underlying that it does not fully master this type of goal. The "Grow herbivore" and "Grow carnivore" are mixed with other impossible goals.
- After achieving mastery of "Grow plant", in Figure C.13c, we see both "Grasp" and "Grow plant" are correctly clustered in the zone of high success probability, with the goals from the test set correctly placed into the two clusters. The goals from the type "Grow herbivore" and "Grow carnivore" are clustered together and almost separated from the impossible goals. Their cluster is close to the zone of high estimated probability.
- Once "Grow herbivore" has been solved, we see in Figure C.13d that the three achieved goal types are clustered in three groups, all in the zone of high estimated success probability. The "Grow carnivore" are far from this zone but still clustered together, apart from the impossible ones. It also appears that the model places "Grow carnivore" goals from the test close to the "Grow herbivore".
- Finally, when the type "Grow carnivore" is mastered, in Figure C.13e, all the goals from the possible goal types (from both train and test) are placed in the zone of high success probability. The goals of type "Grow X" form three clusters tightly packed. The "Grasp" type is separated from the other types.

What is clear from the chronogram is that MAGELLAN's learning modifies the embedding space in a way that facilitates prediction of the probability of success.

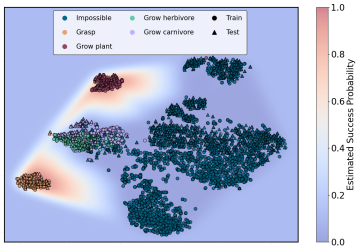
[2]The dynamic is the same for the other seeds.



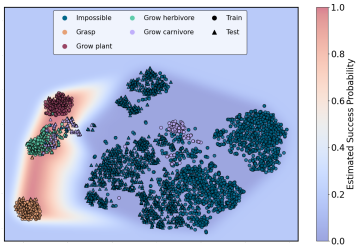
(a) Beginning of training, no goal mastered.



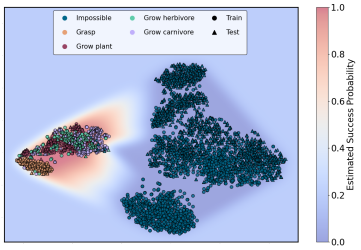
(b) "Grasp" type of goals mastered.



(c) "Grow plant" type of goals mastered.



(d) "Grow herbivore" type of goals mastered.



(e) "Grow carnivore" type of goals mastered.

Figure C.13: Chronogram of the embedding space of the seed 0, at the beginning and after mastering each type of goal.

C.4.5 Q4. Leveraging the generalization abilities when facing new goals

10 adaptation cases throughout training

In Section 8.5.5, we present adaptation tests where the goal space evolves by replacing goals with new unseen goals from similar categories. We conduct the test by training an agent using MAGELLAN for 500k steps. Then, every $50000 * n$ episodes (with $n \in \{1; 2; 3; 4; 5; 6; 7; 8; 9; 10\}$), we stop training and replace MAGELLAN by one of the four ALP methods of goal sampling (MAGELLAN, EK-Online-ALP, Online-ALP, and Uniform). Finally, we resume the training for 50k steps and measure the SR. For EK-Online-ALP, we make it track the agent's competence based on goals sampled by MAGELLAN during the first phase, allowing it to start with an estimation on the new unseen goals. Our test measures the ability of a method to estimate and quickly update competence on unseen goals. Figure C.14 details all the results obtained at the ten points we use in the experiments.

We can divide the different experiments in 2 typical scenarios:

- **Scenario zero LP (Figure C.14a):** the agent has mastered the "Grasp" and "Grow plant" goals and has 0 ALP across all goals. In this scenario, all ALP estimations are equivalent. EK-Online-LP manages to discover new ALP niches faster as all impossible goals are in the same group.
- **Scenario with LP**:
  - **High LP (figures C.14b, C.14c, C.14d):** the agent is getting a high ALP as it is learning some "Grow carnivore" goals. Here, MAGELLAN outperforms baselines by generalizing its ALP estimation and continuing training on these goals.
  - **Medium LP at the end of training (figures C.14e to C.14j):** the agent has almost learned all the goals, and it only remains few goals in the "Grow carnivore" category on which its performance is not stable. When changing the goal space, Online-ALP and Uniform take a lot of steps to find the remaining goals with LP in the new goal space, destabilizing the agent. As a result, the agent's SR decreases just after the transition and does not recover after 50k training epispdes. MAGELLAN and EK-Online-ALP maintain their high SR.

  We notice that the learning curve of the agents using MAGELLAN and EK-Online-ALP are almost identical, but our method does not rely on expert knowledge and naturally clustered the goal space as seen in Figure 8.7b.

Global Sample Efficiency assessment

To obtain a quantitative analysis of the results from Section 8.5.5, we plot in Figure C.15 the average sample efficiency (after κ episodes) of each method, averaged over the 10 tests, with $\kappa \in \{10000; 20000; 30000; 40000; 50000\}$. The sample efficiency after κ episodes is calculated as $\frac{1}{10} \sum_{n=1}^{10} \int_0^{\kappa} SR(50000 * n + t) - SR(50000 * n)\, dt$.

We observe that the Uniform method is the only approach exhibiting a negative average sample efficiency, which is expected since it predominantly samples impossible goals, thereby

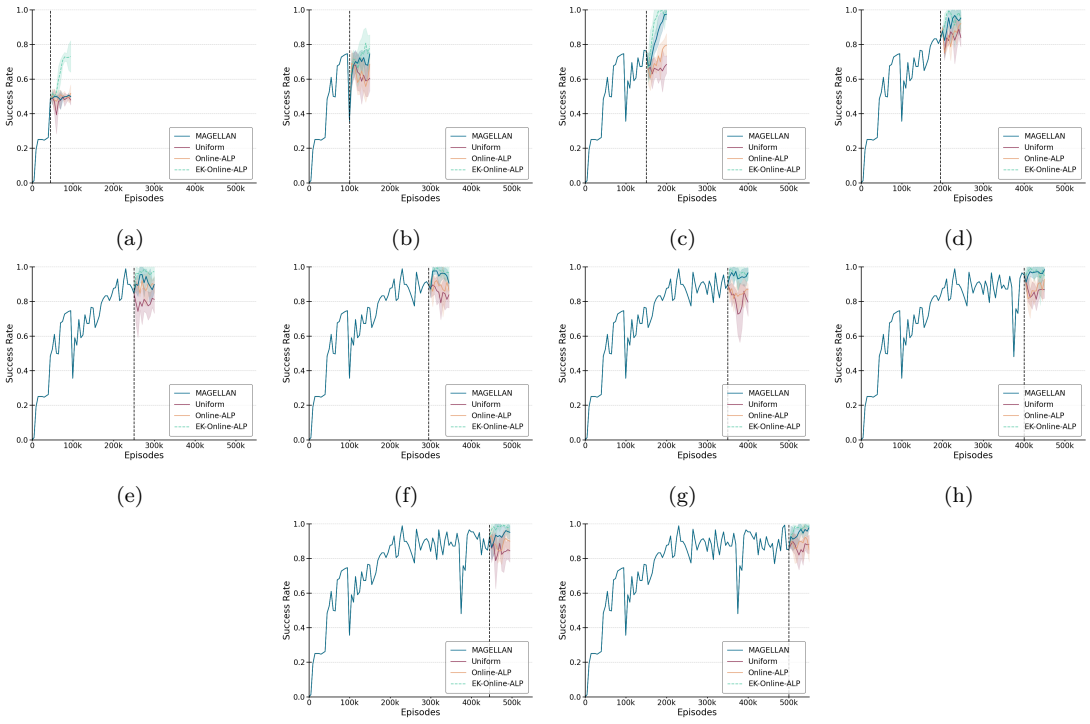


Figure C.14: **Adaptation tests**: Using a single's seed training of 500k episodes, we stop and replace all goals by new unseen goals every 50k episodes. We then resume training by sampling goals using each of our four methods' ALP estimation (MAGELLAN, EK-Online-ALP, Online-ALP, Uniform) and perform 50k training episodes. We report the evolution of observed competence (SR) when evaluating the policies on 64 goals per category from the new training set every 5000 episodes. We report the average competence over evaluated goals along with standard deviation.

destabilizing the agent. In contrast, the other methods demonstrate increasing sample efficiency as κ progresses, reflecting improved success probability estimation and, consequently, more effective goal sampling. However, Online-ALP exhibits only a marginal improvement, as it must continually re-estimate success probabilities across the entire goal space.

Both MAGELLAN and EK-Online-ALP, which leverage generalization to estimate success probabilities for novel goals, achieve higher sample efficiency. Notably, EK-Online-ALP benefits significantly from expert knowledge in clustering the goal space and identifying impossible goals. However, its strong performance is contingent on the assumption that new goals belong to the same categories as those in the training set, which may limit its generalization capacity.

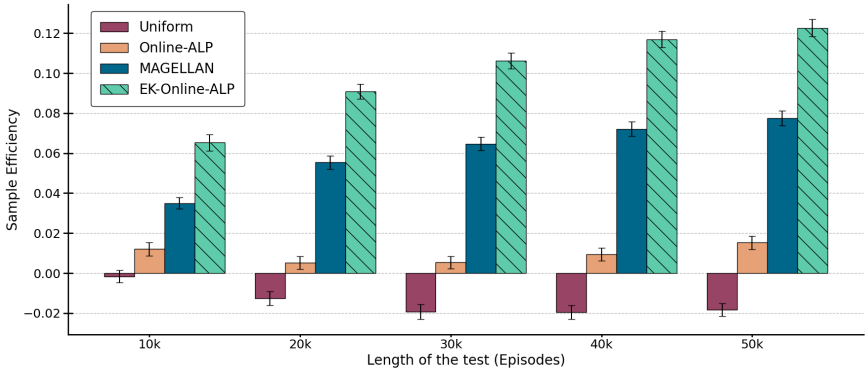


Figure C.15: Average sample efficiency (after κ, the length of the test) of each method average over the 10 tests.

Appendix D

Appendix of AutotelicLLM

This supplementary material provides:

- The detail fo some functions used in the AutotelicLLM algorithm Appendix D.1.
- The hyperparamters used in the experiments Appendix D.2.
- The details of the reward functions described in Section 9.5.1, Appendix D.3.

D.1 Helper functions for the algorithm

D.1.1 $\mathcal{F}_\Lambda$ the filter function

The Filter distribution $\mathcal{F}_\Lambda$ is defined by the following procedure:

1. Sample a category of goals among the C categories:
   $c \sim \text{Uniform}(C)$
2. Define the sets of learnable and not-learnable scores:
   $R_L = \{g_r | L(r) > 0.1 \text{ and } c_{g_r} = c(nm_{g_r}, z_{g_r}, r, L(g_r), D(g_r), c_{g_r}) \in \Lambda\}$,
   $R_N = \{g_r | L(r) \leq 0.1 \text{ and } c_{g_r} = c, (nm_{g_r}, z_{g_r}, r, L(g_r), D(g_r), c_{g_r}) \in \Lambda\}$.
3. Sample a goal among the one with learnable score$g \sim \text{Uniform}(R_L)$.
4. Select the goals with the two closest learnable scores to g in embedding space (excluding g) :
   $g_{r_{L1}},\ g_{r_{L2}} = \arg \min_{g_r \in R_L \setminus \{g\}} \text{distance}(\text{embedding}(g_r),\ \text{embedding}(g))$.
5. Select the two closest goals with not-learnable rewards to g in embedding space:
   $g_{r_{N1}},\ g_{r_{N2}} = \arg \min_{g_r \in R_N} \text{distance}(\text{embedding}(g_r),\ \text{embedding}(g))$.
6. Sample $g'_1,\ g'_2 \sim \text{Uniform}(R_L)$.
   This two goals are sampled for creative combinations with dissimilar goals.

   Then, we obtain the sample $(g,\ r_{L1},\ r_{L2},\ r_{N1},\ r_{N2},\ g'_1,\ g'_2) \sim \mathcal{F}_\Lambda$.

In the OMNI-EPIC-like baseline the Filtering distribution is similar except that the two sets R_L and R_N are replaced by:

$$R_D\{g_r \mid D(r) > 0.1 \text{ and } c_{g_r} = c(nm_{g_r}, z_{g_r}, r, L(g_r), D(g_r), c_{g_r}) \in \Lambda\}, \tag{D.1}$$

$$R_N = \{g_r \mid LP(r) \leq 0.1 \text{ and } c_{g_r} = c(nm_{g_r}, z_{g_r}, r, L(g_r), D(g_r), c_{g_r}) \in \Lambda\} \tag{D.2}$$

D.1.2 Goal and subgoals embedding

When generating a goal, the language model (LLM) is tasked with creating a name, denoted as nm, which includes both the name of the goal (nm_g) and the names of its subgoals ($(nm_{s_g}(1), nm_{s_g}(2), \ldots)$). These names are then embedded into a vector of size 512 using the OpenAI text-embedding-3-small embedder. This process results in the following embedding:

$$z_g = \left(\mathcal{E}(nm_g, nm_{s_g}(1), nm_{s_g}(2), \ldots)\right)$$

D.1.3 Filtering valid goals

Before adding a goal generated by the LLM to the archive we are testing its validity by:

- Checking if it's syntactically correct by executing it's code
- Checking if the name of the reward state, the goal and subgoal names and the reward function is consistent.
- Checking if the reward state and the reward function is jit compatible by trying to compile and execute on a random input.

D.2 Hyperparameters

D.2.1 PPO AGENT

We re-used the hyperparameters of the JAX implementation of PPO with the transformerXL architecture Hamon (2024)

Hyperparameter	Value
Learning Rate	2e-4
Batch Size	8
Number of Epochs	4
Clip Range	0.2
Discount Factor (Gamma)	0.999
GAE Lambda	0.8
Entropy Coefficient	2e-3
Value Function Coefficient	0.5
Max Gradient Norm	1.
Number of Layers	2
Number of Heads	8
Size hidden layers	256
QKV features	256
Window memory	128
Window gradient	64
Size embedding	256
max time step per rollout	100

Table D.1: PPO transformerXL Hyperparameters

D.2.2 Autotelic architecture hyperparameters

Hyperparameter	Value
Update per generation K	150
Samples goals for training $\#S$	1024
Number of steps per rollout p	128
Number of initial goal seed	33
Number of initial categories of goal	6
Max size of the archive N	200

Table D.2: Autotelic architecture hyperparameters

D.3 Example of reward functions generated by the LLM

We present here the code of the reward function displayed in Fig.9.5.

The structure is the following:

- the reward state is defined with a class

- the reward function is a function
- the names are stored in a list

  [<name goal>, <name sugboal 1>, <name subgoal 2>, ...]

Goal Craft and use wood swords:

```
@struct.dataclass
class RewardState_craft_and_use_wood_sword:
    counter_prompt: int = 0

def craft_and_use_wood_sword(state, action, rng, reward_state):
    """
    Check the sequenced goal that includes the sequence of
        subgoals: collect_wood, make_wood_sword, and defeat_monster
        .
    """

    # first subtask: collect wood
    subtask_1_achieved = state.inventory.wood >= 2
    counter_updated = jax.lax.select(jnp.logical_and(
        subtask_1_achieved, reward_state.counter_prompt == 0), 1,
        reward_state.counter_prompt)

    # second subtask: make wood sword
    subtask_2_achieved = state.inventory.wood_sword == 1
    counter_updated = jax.lax.select(jnp.logical_and(
        subtask_2_achieved, reward_state.counter_prompt == 1), 2,
        counter_updated)

    # third subtask: defeat monster
    monster_defeated = jnp.any(state.zombies.health <= 0)
    counter_updated = jax.lax.select(jnp.logical_and(
        monster_defeated, reward_state.counter_prompt == 2), 3,
        counter_updated)

    task_achieved = counter_updated == 3

    reward_state = reward_state.replace(counter_prompt=
        counter_updated)

    return task_achieved, reward_state, reward_state.
        counter_prompt

prompt_sequence_craft_and_use_wood_sword = ['collect wood', 'make
    wood sword', 'defeat monster']
```

Goal Build shelter:

```
@struct.dataclass
class RewardState_build_shelter:
    has_placed_enough_blocks: bool = False
    counter_prompt: int = 0

def build_shelter(state, action, rng, reward_state):
    """
    Check whether the agent builds a shelter using available
       resources (wood, stone) within a defined area.
    """
    def helper_in_bounds(state, position):
        in_bounds_x = jnp.logical_and(0 <= position[0], position
            [0] < state.map.shape[0])
        in_bounds_y = jnp.logical_and(0 <= position[1], position
            [1] < state.map.shape[1])
        return jnp.logical_and(in_bounds_x, in_bounds_y)

    # Define the shelter area (3x3 area around the player)
    shelter_area = jnp.array([
        [-1, -1], [-1, 0], [-1, 1],
        [0, -1], [0, 0], [0, 1],
        [1, -1], [1, 0], [1, 1]
    ], dtype=jnp.int32)

    # Check if the agent has placed blocks in the shelter area
    def check_placed_blocks(unused, loc_add):
        pos = state.player_position + loc_add
        is_in_bounds = helper_in_bounds(state, pos)
        is_placed = jnp.logical_and(is_in_bounds, state.map[pos
            [0], pos[1]] != BlockType.GRASS.value)
        return None, is_placed

    _, is_placed_blocks = jax.lax.scan(check_placed_blocks, None,
        shelter_area)

    # Check if the agent has placed at least 5 blocks in the
       shelter area
    has_placed_enough_blocks = is_placed_blocks.sum() >= 5

    # Update the reward state
    reward_state = reward_state.replace(has_placed_enough_blocks=
        has_placed_enough_blocks, counter_prompt=jax.lax.select(
        has_placed_enough_blocks, 1, 0))
```

```
    return has_placed_enough_blocks, reward_state, reward_state.
        counter_prompt

prompt_sequence_build_shelter = ['gather resources', 'place blocks
    ', 'form shelter']
```

Goal build fortified shelter

```
    @struct.dataclass
class RewardState_build_fortified_shelter:
    has_placed_enough_blocks: bool = False
    counter_prompt: int = 0

def build_fortified_shelter(state, action, rng, reward_state):
    """
    Check whether the agent builds a fortified shelter using stone
        and iron blocks, ensuring it is well-protected.
    """
    # Define the shelter area (5x5 around the player)
    shelter_area = jnp.array([
        [-2, -2], [-2, -1], [-2, 0], [-2, 1], [-2, 2],
        [-1, -2], [-1, -1], [-1, 0], [-1, 1], [-1, 2],
        [0, -2], [0, -1], [0, 0], [0, 1], [0, 2],
        [1, -2], [1, -1], [1, 0], [1, 1], [1, 2],
        [2, -2], [2, -1], [2, 0], [2, 1], [2, 2]
    ], dtype=jnp.int32)

    # Check if the agent has placed blocks in the shelter area
    def check_placed_blocks(unused, loc_add):
        pos = state.player_position + loc_add
        is_in_bounds = jnp.logical_and(0 <= pos[0], pos[0] < state
            .map.shape[0])
        is_in_bounds = jnp.logical_and(is_in_bounds, jnp.
            logical_and(0 <= pos[1], pos[1] < state.map.shape[1]))
        is_placed = jnp.logical_and(is_in_bounds, jnp.logical_or(
            state.map[pos[0], pos[1]] == BlockType.STONE.value,
            state.map[pos[0], pos[1]] == BlockType.IRON.value))
        return None, is_placed

    _, is_placed_blocks = jax.lax.scan(check_placed_blocks, None,
        shelter_area)

    # Check if the agent has placed at least 10 blocks in the
        shelter area
    has_placed_enough_blocks = is_placed_blocks.sum() >= 10

    # Update the reward state
```

```
    reward_state = reward_state.replace(has_placed_enough_blocks=
        has_placed_enough_blocks, counter_prompt=jax.lax.select(
        has_placed_enough_blocks, 1, 0))


    return has_placed_enough_blocks, reward_state, reward_state.
        counter_prompt

prompt_sequence_build_fortified_shelter = ['gather stone', 'gather
    iron', 'place stone blocks', 'place iron blocks', 'form a 5x5
   shelter']
```

Goal Build mob trap:

```
@struct.dataclass
class RewardState_build_mob_trap:
    task_achieved: bool = False
    counter_prompt: int = 0

def build_mob_trap(state, action, rng, reward_state):
    """
    Check whether the agent builds a trap to catch mobs by
        strategically placing blocks and using resources.
    """

    # Define the trap area (3x3 around the player)
    trap_area = jnp.array([
        [-1, -1], [-1, 0], [-1, 1],
        [0, -1], [0, 0], [0, 1],
        [1, -1], [1, 0], [1, 1]
    ], dtype=jnp.int32)

    # Check if the agent has placed blocks in the trap area
    def check_placed_blocks(unused, loc_add):
        pos = state.player_position + loc_add
        is_in_bounds = jnp.logical_and(0 <= pos[0], pos[0] < state
            .map.shape[0])
        is_in_bounds = jnp.logical_and(is_in_bounds, jnp.
            logical_and(0 <= pos[1], pos[1] < state.map.shape[1]))
        is_placed = jnp.logical_and(is_in_bounds, state.map[pos
            [0], pos[1]] != BlockType.GRASS.value)
        return None, is_placed

    _, is_placed_blocks = jax.lax.scan(check_placed_blocks, None,
        trap_area)

    # Check if the agent has placed at least 5 blocks in the trap
        area
```

```
    has_placed_enough_blocks = is_placed_blocks.sum() >= 5

    # Check if any mobs are trapped in the area
    def check_trapped_mobs(unused, loc_add):
        pos = state.player_position + loc_add
        is_in_bounds = jnp.logical_and(0 <= pos[0], pos[0] < state
            .map.shape[0])
        is_in_bounds = jnp.logical_and(is_in_bounds, jnp.
            logical_and(0 <= pos[1], pos[1] < state.map.shape[1]))
        is_mob = jnp.logical_and(is_in_bounds, state.mob_map[pos
            [0], pos[1]])
        return None, is_mob

    _, is_trapped_mobs = jax.lax.scan(check_trapped_mobs, None,
        trap_area)

    # Check if there is at least one mob trapped
    has_trapped_mobs = is_trapped_mobs.sum() > 0

    # Task achievement is based on having placed enough blocks and
        trapping mobs
    task_achieved = jnp.logical_and(has_placed_enough_blocks,
        has_trapped_mobs)

    # Update the reward state
    reward_state = reward_state.replace(task_achieved=
        task_achieved, counter_prompt=jax.lax.select(task_achieved,
        1, 0))

    # Reward is given when the task is achieved
    reward = jax.lax.select(task_achieved, 1., 0.)

    return task_achieved, reward_state, reward_state.
        counter_prompt

prompt_sequence_build_mob_trap = ['gather resources', 'place
    blocks strategically', 'trap mobs']
```

D.4 Prompt for the LLM

```
You are a player in an open-world game trying to write reward
    functions to train an Deep-Reinforcement learning AI agent.You
    want the AI agent to achieve as many different interesting and
    complex tasks as possible. For this you want to design a
    curriculum of tasks tailored to the AI agent capabilities. The
```

```
AI agent is acting in the crafter environement. Crafter is an
open-world game designed to evaluate the general abilities of
intelligent agents within a single environment. Developed for
reinforcement learning research, Crafter provides a
procedurally generated 2D world where agents must perform tasks
 such as foraging for food, finding water, building tools and
constructions, and defending against monsters. Crafter is in 2D
.


Instructions:
- The next task should be learnable:
    - Not too difficult given the difficulty of the previous tasks
      . Don't create tasks that are not learnable.
    - Realistic for the agent to achieve in the current
      environment

- The next task should be interesting:
    - Not too similar to the current tasks. Do not copy the
      existing tasks.
    - Not too easy for the agent to achieve. The agent should
      learn something new.
    - Useful according to humans, making it worth learning.
    - Creative or surprising.
    - Optionally, the task can be fun and engaging to watch.

- The task should be well-defined:
    - The code should reflect the task description (in the
      docstring). If it's not the case, create a new reward
      function (with a different name). Tasks that are are not
      learnable usually are not well defined and can be corrected
       by changing how the code reflects the task description.
    - If you want to re-use a helper function, you have to re-
      write it in the reward function.
    - Pay attention to the state representation (EnvState and
      Inventory) and the game logic. The reward function should
      be compatible with the game logic and the EnvState.

- Jax and jit compatible:
    - don't use python conditionals and boolean operations.
        - example:
            - python (not jax and jit compatible): has_resources =
                state.inventory.stone >= 3 and state.inventory.
              wood >= 2
            - jax and jit compatible: has_resources = jnp.
              logical_and(state.inventory.stone >= 3, state.
              inventory.wood >= 2)
```

```
    - jnp.logical_or() and jnp.logical_and() takes only 2
      positional arguments, so you can't use it with more than 2
      arguments. Alternatively you can use & and | but add
      parentheses around the condition.
    - if you use array in the reward state, you should use the
      following syntax using field:
class RewardState_move_far_away:
    start_position: jnp.ndarray = field(default_factory=lambda:
      jnp.array([0, 0], dtype=jnp.int32))
    counter_prompt: int = 0


The world-state is updated based on a state representation ("state
    ") that is updated after each action of the agent by the
    following game logic. The state has the following structure:


# Inventory class to track player's collected resources and
    crafted tools
class Inventory:
    wood: int = 0
    stone: int = 0
    coal: int = 0
    iron: int = 0
    diamond: int = 0
    sapling: int = 0
    wood_pickaxe: int = 0
    stone_pickaxe: int = 0
    iron_pickaxe: int = 0
    wood_sword: int = 0
    stone_sword: int = 0
    iron_sword: int = 0

# Mobs class to represent creatures (zombies, cows, skeletons) in
    the game
class Mobs:
    position: jnp.ndarray
    health: int
    mask: bool
    attack_cooldown: int

# EnvState class to represent the entire game state
class EnvState:
    map: jnp.ndarray= map # 64x64 grid representing the world
    mob_map: jnp.ndarra= jnp.zeros((64, 64), dtype=bool)

    player_position: jnp.ndarray = player_position
```

```
    player_direction: int = Action.UP.value

    # Player's vital stats (health, food, water, energy)
    player_health: int = 9
    player_food: int = 9
    player_drink: int = 9
    player_energy: int = 9
    is_sleeping: bool = False

    # Rates of change for player's vital stats
    player_recover: float = 0.0
    player_hunger: float = 0.0
    player_thirst: float = 0.0
    player_fatigue: float = 0.0

    inventory: Inventory
    zombies: Mobs
    cows: Mobs
    skeletons: Mobs
    arrows: Mobs
    arrow_directions: jnp.ndarray

    # Tracking planted crops
    growing_plants_positions: jnp.ndarray
    growing_plants_age: jnp.ndarray
    growing_plants_mask: jnp.ndarray

# Enum for different block types in the game world
class BlockType(Enum):
    INVALID = 0
    OUT_OF_BOUNDS = 1
    GRASS = 2
    WATER = 3
    STONE = 4
    TREE = 5
    WOOD = 6
    PATH = 7
    COAL = 8
    IRON = 9
    DIAMOND = 10
    CRAFTING_TABLE = 11
    FURNACE = 12
    SAND = 13
    LAVA = 14
    PLANT = 15
    RIPE_PLANT = 16
```

```
# Enum for possible actions the player can take
class Action(Enum):
    NOOP = 0
    LEFT = 1
    RIGHT = 2
    UP = 3
    DOWN = 4
    DO = 5
    SLEEP = 6
    PLACE_STONE = 7
    PLACE_TABLE = 8
    PLACE_FURNACE = 9
    PLACE_PLANT = 10
    MAKE_WOOD_PICKAXE = 11
    MAKE_STONE_PICKAXE = 12
    MAKE_IRON_PICKAXE = 13
    MAKE_WOOD_SWORD = 14
    MAKE_STONE_SWORD = 15
    MAKE_IRON_SWORD = 16

# Blocks that the player cannot walk through
SOLID_BLOCKS = jnp.array(
    [
        BlockType.WATER.value,
        BlockType.STONE.value,
        BlockType.TREE.value,
        BlockType.COAL.value,
        BlockType.IRON.value,
        BlockType.DIAMOND.value,
        BlockType.CRAFTING_TABLE.value,
        BlockType.FURNACE.value,
        BlockType.PLANT.value,
        BlockType.RIPE_PLANT.value,
    ],
    dtype=jnp.int32,
)

# Directions for movement and interaction
DIRECTIONS = jnp.concatenate(
    (
        jnp.array([[0, 0], [0, -1], [0, 1], [-1, 0], [1, 0]],
            dtype=jnp.int32),
        jnp.zeros((11, 2), dtype=jnp.int32),
    ),
    axis=0,
)
```

```
# Adjacent blocks for interaction
CLOSE_BLOCKS = jnp.array(
    [
        [0, -1],
        [0, 1],
        [-1, 0],
        [1, 0],
        [-1, -1],
        [-1, 1],
        [1, -1],
        [1, 1],
    ],
    dtype=jnp.int32,
)


# Example usage
view = extract_agent_view(state).

The code you create needs to be compatible with jax and jit. To
    create a new task you should write a function with the same
    input and output as the previous examples. Also add a reward
    class like the previous example. The reward class is a way to
    store information across different step updates and generate
    reward based on information contained in a sequence of states
    of the world. Also add a list of brief description of the
    sequential subtasks (less than 10 words per subtasks and less
    than 10 subtasks)to guide the AI agent in the learning process.
     Create one new task by writing the code for its reward
    function with its associated reward state and prompt sequence.
    Take into account the agent achievements to make a novel
    interesting task that is not too easy and not too hard for the
    AI agent. Answer based on the syntax of the following format:


My proposed task is:
Reasoning for what the next reward should be:
<brief reasoning>

Next The code for the reward function and its reward state and
    prompt sequence is:
'''python
'''
<reward function>

<reward state>
```

```
<prompt_sequence = [<description of the task>, <description of
    subtask 1>, <description of subtask 2>, ...]>
'''


Please create a new reward function. Don't re-use the same name of
    function. You should take inspiration from the reward
    functions below to create a new reward function that is
    interesting, novel and achievable.For each reward functions a
    score between 0 and 100 is given to describe the difficuly and
    learnability based on the AI-agent experience in craftax.

Below are some rewards that are learnable:

The learnability of the following goal is: <score learnability>
The difficulty of the following goals is: <score difficulty>
<g, main goal sampled uniformly among the learnable ones>

The learnability of the following goal is: <score learnability>
The difficulty of the following goals is: <score difficulty>
<r_{L1}, closest learnable goal in embedding space >

The learnability of the following goal is: <score learnability>
The difficulty of the following goals is: <score difficulty>
<r_{L2}, second closest learnable goal in embedding space >

The learnability of the following goal is: <score learnability>
The difficulty of the following goals is: <score difficulty>
<g_1', creative goal sampled uniformly among the learnable ones>

The learnability of the following goal is: <score learnability>
The difficulty of the following goals is: <score difficulty>
<g_2', creative goal sampled uniformly among the learnable ones>


Below are some rewards that are not learnable:

The learnability of the following goal is: <score learnability>
The difficulty of the following goals is: <score difficulty>
<r_{N1}, closest non-learnable goal in embedding space >

The learnability of the following goal is: <score learnability>
The difficulty of the following goals is: <score difficulty>
<r_{N2}, second closest non-learnable goal in embedding space >

My proposed task is:
```

Appendix E

Appendix of EAGER

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

- Section E.1 summarises the different tasks and the assumption used in RIDE, EAGER, ELLA.
- Section E.2 gives more details about training of the QA module and the agent. It also includes explanations of how we built the training data set for the QA module.
- Section E.3 gathers several results on EAGER: comparison with behavioural cloning, generalisation capacity of QA, robustness results of EAGER...
- Section E.4 contains a commented version of the EAGER algorithm.
- Section E.5 summarises hyperparameters.

E.1 Tasks description and assumptions used for the different method of reward shaping

Table E.1 describes the tasks used in the experiments with an example and if it has been used to train the QA module or the agent. The Unlock and Open tasks have the same type of instructions, the agent can nevertheless see the difference because in the Unlock task, the door is a solid square where in the Open task, the door is just a border.

Table E.1: Tasks description, ✓means that the task is used for training the QA resp. the Agent

Task	**Explanation**	**Example**	**Train QA**	**Train Agent**
PutNextTo	put an object next to another	*put the purple ball next to the blue key*	✓	✓
PickUp	pick up an object	*pick up a red box*	✓	
Open	open a door (does not require a key)	*open the green door*	✓	
Unlock	open a door using the key of the same colour	*open the green door*		✓
Sequence	sequence of two of the previous tasks: task 1 before/after task 2	*put the blue key next to the red box before you open the grey door*	✓	✓

E.2 Training details

In this section we explain how we create the QA training data set, the architecture of the agent used for RL training, and give more details on the baselines.

E.2.1 QA architecture and pre-training

QA architecture The QA architecture is based on the Episodic Transformer architecture Pashevich et al. (2021) depicted in Figure E.1. Using multimodal transformers, the QA can direct its attention over certain observations in the trajectory about the words used for the question and the previously taken action. Thus we can train the QA over the full trajectory and use it on partial trajectories (up to time step t) at test time.

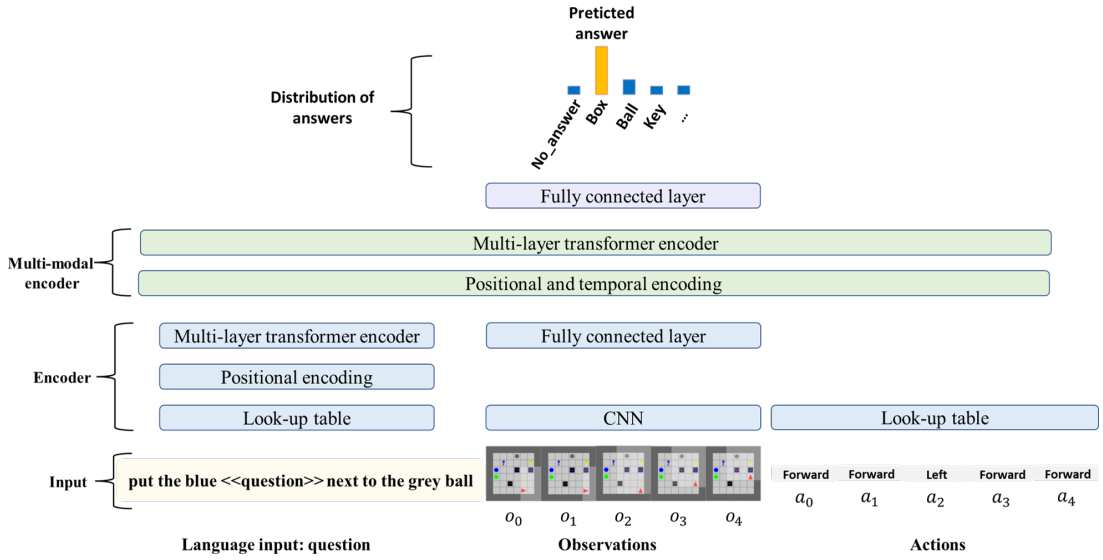


Figure E.1: The QA architecture is based on the architecture of the Episodic Transformer Pashevich et al. (2021). To predict the correct answer the QA model received a natural language question, visual observations, and actions generated by the agent from the beginning of an episode (see Figure E.3). Here we show an example that corresponds to the 5th time step of an episode. After processing the question with a transformer-based language encoder, embedding the observation with a convolutional neural network CNN and passing actions through a look-up table, the agent outputs the correct answer. During training, we use full trajectories. At test time, we ask all the questions in the active set of questions using the current trajectory.

QA training data set We train the QA with a mix of 4 tasks: Open-Large, PickUp-Large, PutNextTo-Local, and Sequence-Medium. We use a bot provided with the BabyAI platform that procedurally solves the environment. Using it, we generate 7500 example trajectories for each task. We use a mix of tasks to push the QA to leverage the compositionality of language. Indeed, the Sequence task is created by putting in sequence two tasks from Open, PickUp, and PutNextTo. Compared to QA trained on an individual task, our QA needs fewer examples per task. Thus the QA can use the environment with the goal *put the blue*

key next to the red box then open the grey door to ground the instruction both for PutNextTo and Open.

Adding "no_answer" questions To train the QA to respond: "no_answer" and prevent it from hazard-guessing the answer, we randomly associate certain paths and questions from unrelated objectives. To generate these questions, for each new trajectory generated by the bot, we take a goal among the last three used. If this goal differs from the goal used for the trajectory, we use it to create questions that are associated with "no_answer". To avoid biasing the dataset with too much of these questions, we randomly keep some of the generated "no_answer" questions. The probability to keep a question depends on the number of words in common between the goal corresponding to the trajectories and the random goal. If these two goals share a lot of words in common, it is harder for the QA to see that it cannot answer the question generated from the random goal, thus we want to favour these negative examples in our dataset. We define $\mathbb{P}(\text{keep question}) = \frac{0.325}{1+\exp(6.75-3 \text{ number words in common})} + 0.095$. The hyperparameters are chosen empirically to obtain a number of "no_answer" questions similar to the number of questions generated for other words.

Wide distribution of trajectories As explained in the experimental settings and empirically demonstrated in the experiments, a broad distribution of trajectories improves the QA and the efficiency of EAGER overall. To obtain such distribution from the procedural bot, we replace at each time step with a probability p the action of the bot by a random action in: *turn right*, *turn left*, *go forward*, *pick up*, *drop* as it is shown in Figure E.2 . For each new example trajectory, we randomly select p, with probability 0.45 for $p = 0$ (no random action) and with a probability of $[0.35, 0.1, 0.1]$ for $p \in [0.1, 0.4, 0.8]$ respectively. We use such a distribution to sample p to ensure enough good quality trajectories in the dataset. If the bot cannot complete the task due to the added noise, we discard the trajectory from the set of example trajectories.

We train the QA using the cross entropy loss with a batch size of 10 and a learning rate of 10^{-4} using Adam Kingma & Ba (2014). We multiply the learning rate by 0.1 every 5 epochs. Table E.2 summarises the hyperparameters used.

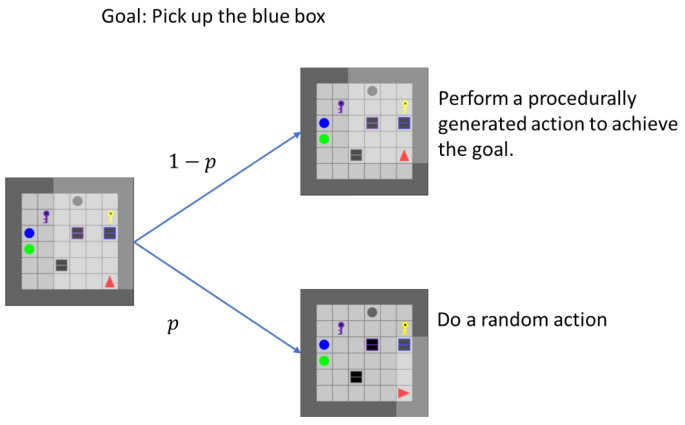


Figure E.2: Choice of a new action during the generation of the wide distribution of trajectories. At each time step, the agent can perform a random action with probability p.

E.2.2 Agent architecture and training

We use the actor-critic architecture and the PPO implementation proposed in BabyAI Chevalier-Boisvert et al. (2019) (Figure E.3). This PPO implementation uses the default hyperparameters (see Table E.3). The output of the actor is the distribution of actions. There are six possible actions: turn left, turn right, forward, pick up, drop, toggle and done (to signify the completion of the task). The output of the critic is the value of the current state.

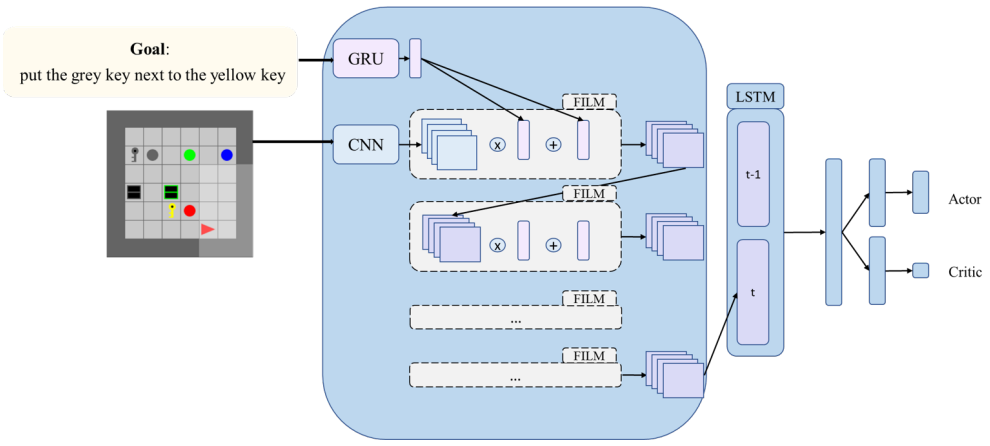


Figure E.3: The actor-critic architecture uses a multimodal encoder that mixes language and image using a feature-wise affine transformation FiLM Perez et al. (2017a). Then the encoded representation passes through an LSTM to process the history of observations. Finally, the output is used in an actor and critic heads.

E.2.3 RIDE

RIDE is a method that does not use language and gives an intrinsic reward encouraging agents to explore actions that significantly alter the state of the environment, as measured in a learned latent space. To use it as a baseline, we reimplemented the RIDE algorithm from the open-source implementation.[1]. We kept the same architecture as the original work and adapted RIDE to the on-policy setting of our PPO algorithm by updating the dynamics models once per batch of on-policy rollouts. We used the hyperparameters values published in the code repository for the coefficients on forward dynamics loss and inverse dynamics loss (10 and 0.1 respectively), as well as the published value for learning rate of 10^{-4}. We took the intrinsic reward coefficient $\lambda = 0.5$ as it was empirically shown in ELLA to be the best coefficient.

[1] https://github.com/facebookresearch/impact-driven-exploration

E.2.4 ELLA

ELLA is a reward shaping approach for guiding exploration based on the principle of abstraction in language decomposing high-level goals into low-level auxiliary objectives. Two classifiers are learned: a termination classifier that determines when a certain auxiliary objective has ended and a relevance classifier that determines which auxiliary objective is relevant for the high-level goal. For instance, *pick up yellow key* is relevant for the goal *open yellow door*. The termination classifier is trained from labelled trajectories and the relevance classifier is learned online. Expert knowledge is required to label trajectories and to determine which set of auxiliary objectives is interesting such as: *go to object* or *pick up object*. For this baseline, we rerun the open-source code.[2]

E.2.5 EAGER and RIDE

EAGER and RIDE reward different aspects of the exploration, so their combination has the potential to outperform the two methods taken separately. In our experiments we simply add the intrinsic reward returned by EAGER and RIDE, weighted with a shape reward weight λ_{RIDE} and λ_{EAGER}. Nonetheless, we cannot use the technique explained in Section 4 to find the optimal value of λ. Indeed, this technique is based on the sparsity of the reward, and adding the intrinsic reward from RIDE invalidates this condition. Thus we have to resort to ad hoc methods for tuning λ_{RIDE} and λ_{EAGER} in the combined version.

We observe that the values of λ used by ELLA and EAGER are of the same order of magnitude, so we use the value found in Mirchandani et al. (2021) for the combination of ELLA et RIDE. Thus we take $\lambda_{RIDE} = 0.05$ and $\lambda_{EAGER} = 0.1$, we test this combination on PutNext-local, PutNext-medium and Unlock-Medium tasks.

We obtain better results for the task Unlock-Medium Figure E.4. For the PutNext-local and PutNext-medium tasks the combination of RIDE and EAGER performs worse. Thus this combination is highly sensitive to the value of λ_{RIDE} and λ_{EAGER}.

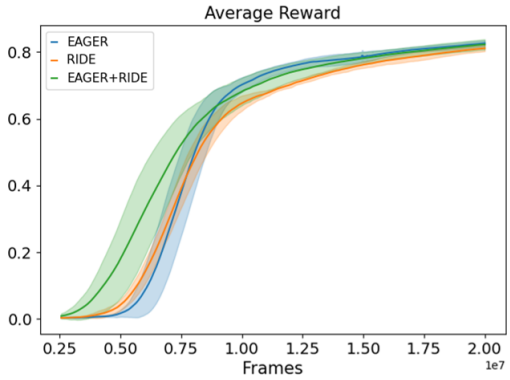


Figure E.4: Average Reward for the Unlock-Medium task for EAGER, RIDE and the combination of EAGER and RIDE.

[2] https://github.com/Stanford-ILIAD/ELLA/tree/22884a3da33da2534754693280a47bb0d99eb8c5

E.3 More results on EAGER

Statistical significance of the results for Unlock-Medium For Unlock-Medium the last value for the average return of EAGER is 0.8903, higher than the one of ELLA (0.8765), RIDE (0.8761) and PPO without reshaping (0.8747), see Figure E.5. The standard deviation of EAGER is higher (0.011) than the one of ELLA (6.10^{-4}), RIDE (2.10^{-4}) and PPO without reshaping (0.0021). Indeed, EAGER is still learning and exploring, whereas the baselines are stuck in a local minimum. Moreover, we use a Welch's t-test to test the null hypothesis: equality between the mean of EAGER and ELLA at the end of the curve. The test rejects this hypothesis with $p_{\text{value}} = 1.9\,10^{-31}$. Thus EAGER significantly outperforms ELLA in this task.

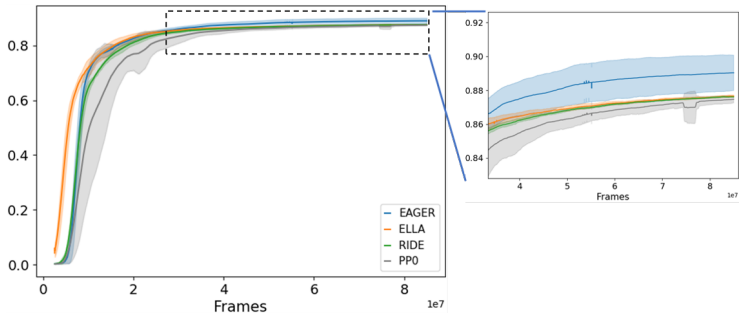


Figure E.5: Average Reward for EAGER and baselines.

Comparison with behavioural cloning and offline RL To train the QA, we used a data set of 7.500 example trajectories per task. One can ask if these trajectories can be used to train an agent through behavioural cloning to obtain results similar to the ones obtained with EAGER. However, experiments performed in Chevalier-Boisvert et al. (2019) show that tasks like PutNextTo-Local require at least 244.000 example trajectories to be learned successfully. Nonetheless, we trained an agent with behavioural cloning using our data set. For the PutNextTo-Local task, our trained agent failed to complete any goal. But when we examine generated trajectories, it seems to display relevant behaviour, going around the objects of interest. This behaviour is coherent with the results given in Mirchandani et al. (2021) with 50.000 example trajectories needed for learning goals like *Go to red ball*. Thus EAGER requires much less demonstration trajectories than behavioural cloning, as it is more efficient to train a QA module than a BC policy. To further this analysis, we have implemented IQL, a recent offline RL algorithm Kostrikov et al. (2021). IQL never needs to evaluate actions outside the data set, but still maintains some generalization capability. Nonetheless, this method also fails due to the small number of trajectories, with a success rate of 0 for PutNextTo-local tasks. To ensure that the implementation is correct, we tested IQL with a simple GoTo task. It obtains a non zero result on the test set with 6% of the test trajectories being successful. Thus the data set used for training the QA cannot be used to warm start an agent. This result is not too surprising, it is indeed easier to learn from a data set to recognise a pattern (as the QA module does) rather than learning a policy Bahdanau et al. (2019).

Generalisation properties of the QA module We also verified that the QA can generalise by correctly answering questions formulated from goals never seen in training. To

check this, we trained the QA with questions and trajectories generated from the Sequence-Medium task. To test generalisation, we only used in the test data set trajectories and questions generated from goals that are not present in the train data set. On the test set, we obtained a success rate of 0.67 to be compared with 0.74 when we tested with trajectories with already seen goals. Thus the QA module can generalise to new goals, making EAGER efficient even for goals not present in the QA training data set.

Ablation of the policy invariance To ensure the policy invariance requirement (see Section 12.4.2) we subtracted the shaped reward at the final step of successful trajectories by neutralizing the intrinsic reward. In Figure E.6, the curve in orange is the learning curve when we do not apply neutralization. The agent learns faster that the one on which neutralization is applied because it receives more indication through reward. However, it ends up stuck in a local minimum because the final policy is influenced by the intrinsic reward.

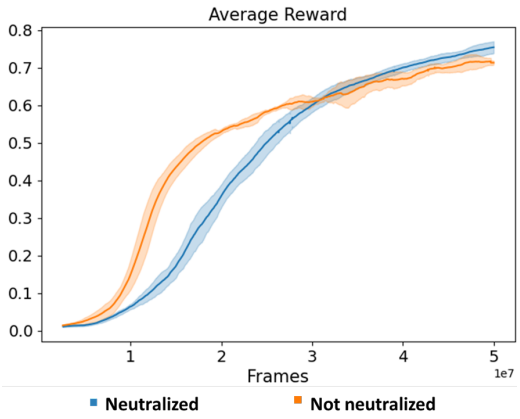


Figure E.6: Ablation of the neutralization of the intrinsic reward on the task PutNextTo-local.

QA performance along the trajectory In order to minimise human intervention, we only use whole trajectories to train the QA. It is legitimate to ask whether the performance of the QA module changes along the trajectory. To determine this, we used the number of different attempts before successfully answering a question as a proxy to measure the performance of the QA module on short trajectories.

We used the PutNext-local task which has the following structure: "Put the adjective_1 noun_1 next to the adjective_2 noun_2". Measuring the number of different attempts before finding the correct answer, we began to count after the first answer different from "no_answer" (before that, the QA estimates that it cannot answer). For this task, after 100 trajectories, we found:

It appears that the QA module needs on average twice as many attempts to guess the answer at the beginning (when the trajectory is short and partial) rather than at the end of the trajectory. Nonetheless, in practice, this does not seems to impact the performance of the methods, which underlines the robustness of EAGER.

Token	Average number of attempts
adjective_1	4.37
noun_1	2.55
adjective_2	1.69
noun_2	1.25

Results for the task Sequence with an extended time budget In order to understand why EAGER suddenly does better than ELLA after 6.5e8 frames Figure E.7, we empirically observed which tasks were successful before and after this threshold. The goals in Sequence task are a combination of two tasks among 'Pickup', 'Go to','Open', and 'Put Next to'. Nonetheless, the later is already a combination of the tasks 'Pickup' and 'Go to', thus more difficult than any of them alone. In Figure E.8 we observe the reward obtains by ELLA and EAGER for goals without 'Put Next to' tasks in it, with 1 'Put Next to' and 2 'Put Next to'. The former type represents around 50% of the goals, goals with 1 'Put Next to' represents 40% and the latter accounts for 10% of the goals. Thus, after 6.5e8 frames, the improvement of the EAGER agent for the goals with 1 and 2 'Put Next to' allows for an important increased in the total reward.

Up to 6.5e8 frames EAGER and ELLA only mostly succeed on Sequence tasks that does not contains a 'Put Next to' goal. However, when the agent starts to have a trajectory that brings it closer to success for tasks with 'Put Next to' in them, it seems that EAGER recognises it and rewards it. In contrast ELLA is much slower as it must first learn to break down complex goals (with one or two 'Put Next to' tasks in them) into auxiliary objectives (using its relevance classifier) before it can use them for reward shaping.

If we look at the curves of ELLA and EAGER we can try to interpret all the different phases:

- At the beginning (0 to $1,5.10^8$ frames) ELLA does better than EAGER because the relevance classifier breaks down the goal into auxiliary objectives quite easily, whereas EAGER is hampered by the fact that the trajectories are shorter and performs poorly (as shown in the paragraph: *QA performance along the trajectory*)
- After ($1,5.10^8$ to $3,5.10^8$) the agent's trajectories improve and become longer, EAGER gains in performance and overtakes ELLA
- From $3,5.10^8$ to $6,5.10^8$ EAGER and ELLA manage to achieve almost the same goals (goals with no 'Put Next to tasks in it)
- From $6,5.10^8$ frames the agent's trajectories for complex goals are longer, EAGER is performing well while ELLA is training the relevance classifier to decompose complex goals into auxiliary objectives (this is learned online from the agent's successful trajectories which explains the slow performance)

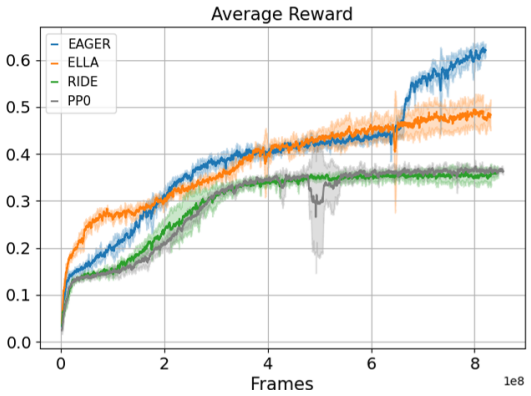


Figure E.7: Average reward for the task Sequence.

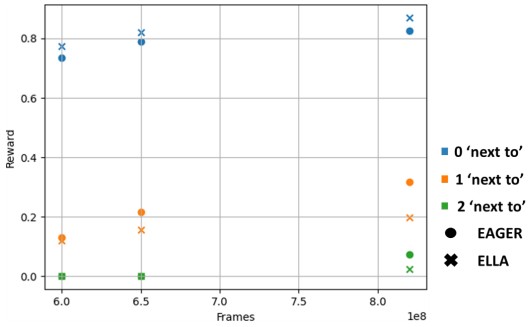


Figure E.8: Average reward over 500 goals and 4 seeds, in Sequence, distributed in three categories: goals without 'Put next to' task in them and goals with 1 or 2 'Put next to' tasks.

Relation between sample efficiency and success rate of the QA In the experiments on EAGER robustness, we looked at the robustness of EAGER relatively to the success rate of the QA during pre-training. However, we only used some values of SR — by sampling the QA at different training epochs— to train the RL agent. To strengthen our point, we used a Gaussian Process model Rasmussen & Williams (2005) to fit the relation between the success rate of the QA and the sample efficiency (SE) of EAGER. The SE is defined as $SE = \frac{1}{n_{frames}} \sum_i^{n_{frames}} r_i)$, where r_i is the extrinsic reward and n_{frames} the number of frames seen during training. Figure E.9 shows once again that EAGER is robust to the quality of the QA module with an almost constant sample efficiency when $SR > 0.52$

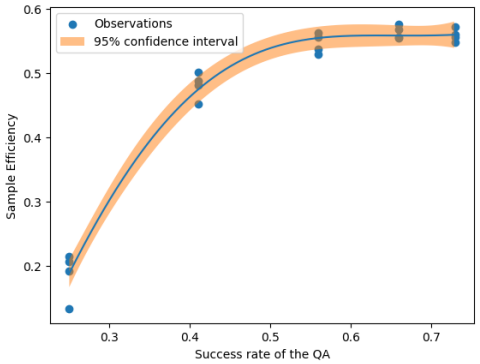


Figure E.9: Gaussian process model to fit Sample Efficiency (SE) of the EAGER algorithm function of Success Rate (SR) of the QA. For each value of SR, the different points correspond to the different seeds used to train the RL agent on the PutNextTo task.

E.4 Commented EAGER algorithm

6 outlines the pseudo-code of our learning architecture. At the beginning of an episode, the EAGER algorithm uses the goal to generate an active set of questions $\mathcal{Q}$ that is updated at each time step by removing correctly answered questions. Each question can be understood as an auxiliary objective guiding the agent towards the completion of the main goal. When an auxiliary objective is completed (i.e. the corresponding question is correctly answered), the agent receives an intrinsic reward λ. To avoid modifying the optimal policy with these additional rewards, we substract the sum of discounted intrinsic rewards from the reward at the last step of successful trajectories with the NEUTRALISE function. In ELLA Mirchandani et al. (2021), the authors prove that shaping the reward does not change the optimal policy when using this neutralisation procedure.

Algorithm 6 Automatic auxiliary goal generation and reward shaping using EAGER

Input: θ_0 initial policy parameters, λ bonus scaling factor, ENV the environment and OPTIMISE an RL optimiser

for k=0,..., n_{step} **do**
  $\omega^g, o_0, \text{done}_0 \leftarrow$ENV.$reset()$
  $\mathcal{Q}_0 = \{q_1, ..., q_k\} \leftarrow QG_k(\omega^g)$ ▷ Generate questions by masking words
  $t \leftarrow 0$
  while done_t not True **do**
    $a_t \leftarrow \pi^{\theta_0}(o_t)$
    $o_{t+1}, r_t, \text{done}_{t+1} \leftarrow$ENV$(a_t)$
    $r'_t, \mathcal{Q}_{t+1} \leftarrow$ QA_SHAPE$(\mathcal{Q}_t, \tau, r_t)$ ▷ Shape reward and update active set of question $\mathcal{Q}$
    if done_{t+1} is True **then**
      $N \leftarrow t$
      $r'_N \leftarrow$ NEUTRALISE$(r'_{1:N})$
  Update $\theta_{k+1} \leftarrow$ OPTIMISE$(r'_{1:N})$

function QA_SHAPE$(\mathcal{Q}_t, \tau, r_t)$
  for q in $\mathcal{Q}_t$ **do**
    $\tilde{a}^* \leftarrow QA(q, \tau)$ ▷ Answer q using the trajectory, $\tau = (o_i, a_i)_{i \in [|0,t|]}$
    if $\tilde{a}^*$ is correct answer to q **then**
      $r'_t = r_t + \lambda\, QA(\tilde{a}^* \,|\, q, \tau)$
      $\mathcal{Q} = \mathcal{Q} \setminus \{q\}$ ▷ Update the active set of questions
  return $r'_t, \mathcal{Q}$

function NEUTRALISE$(r'_{1:N})$
  $r'_N \leftarrow r'_N - \sum_{t \in T_\mathbb{S}} \gamma^{t-N} r'_t$ ▷ $T_\mathbb{S}$ time steps where the agent has received a shape reward
  return r'_N

E.5 Hyperparameters tables

This section contains three tables: the hyperparameters used for training the QA in Table E.2, the hyperparameters for the PPO algorithm in Table E.3, and the shape reward value λ depending on the task in Table E.4.

Table E.2: QA training hyperparameters

Variable	**Value**
batch size	10
learning rate (lr) at the beginning	10^{-4}
number of steps before decreasing lr	5
factor decrease	0.1

In Table E.4 we give all the elements we use to compute λ, which is the value of the intrinsic reward: $\lambda = \frac{\gamma^N r_N}{k}$, where $\gamma = 0.99$ is the discount factor. $r_t = 20(1 - 0.9\,\frac{t}{H})$ is the reward obtained for completing the goal at step t, with H the maximum number of steps for a given task. In the calculation of λ, we assume that once trained, the agent completes the goal in N steps in the worst case scenario. k represents the maximum number of questions that can be generated from goals of a certain task.

Table E.3: PPO hyperparameters

Variable	**Value**
batch size	2560
mini-batch size	1280
discount factor	0.99
lr	7×10^{-4}
entropy coefficient	0.01
loss coefficient	0.5
clipping-ϵ	0.2
generalised advantage estimation parameter	0.99

Table E.4: Value of λ depending on the task.

Task	**k**	**H**	**N**	λ
PutNextTo-Local	4	128	40	2.4
PutNextTo-Medium	4	256	80	1.6
Unlock-Medium	2	128	40	4.8
Sequence-Medium	9	512	185	0.23

Appendix F

Appendix of HERAKLES

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

- Section F.1 details our environment.
  - Section F.1.1 provides our environment's achievement tree with the list of all possible goals and their requirements.
  - Section F.1.2 details the modifications we performed to Crafter.
  - Section F.1.3 explains our approach to quantifying goals' difficulty.
  - Section F.1.4 details the observation space used by our agents.
- Section F.2 provides complementary results.
  - Section F.2.1 shows the per-task success rate for each method across training.
  - Section F.2.2 analyzes the generalization abilities of each method. We study both generalization to synonyms and n-compositionality.
  - Section F.2.3 provides insights on HERAKLES' high-level sampling strategy. We notably study the evolution fo sampling strategy for multiple goals.
- Section F.3 details HERAKLES' interactions between the high-level and low-level policy.
  - Section F.3.1 gives details about our implementation of HERAKLES. We also provide the link to our code.
  - Section F.3.2 provides and analyzes multiple ablation studies on HERAKLES.
- Section F.4 details the implementation of our high-level policy.
  - Section F.4.1 explains how our LLM backbone is used and trained to select skills.
  - Section F.4.2 details our prompting strategy.
- Section F.5 details the implementation of our low-level policy.
  - Section F.5.1 explains the architecture of our lightweight neural networks used for the low-level policy.

  - Section F.5.2 details the low-level policy's training strategy based on AWR Peng et al. (2019).
  - Section F.5.3 proposes an ablation study on AWR's buffer size impact on our low-level policy.

- Section F.6 gives details about our low-level success rate estimator.
  - Section F.6.1 shows the prompt given to our estimator.
  - Section F.6.2 provides details on the training strategy for the low-level success rate estimator.
  - Section F.6.3 explains how the estimation is used to generate the set of skills the high-level policy samples from.

- Section F.7 describes how our autotelic goal sampler reimplements MAGELLAN Gaven et al. (2025).
  - Section F.4.2 shows the prompt given to MAGELLAN's success rate estimator.

- Section F.8 provides implementation details of our baselines.
  - Section F.8.1 describe our implementation of FuN Vezhnevets et al. (2017).
  - Section F.8.2 details our implementation of POAD's Wen et al. (2024b) training strategy.

- Section F.9 details the hyper-parameters we used.

F.1 Environment detail

F.1.1 Achievement tree

We build upon the Crafter environment introduced by Hafner (2022), introducing several modifications to emphasize learning in a heterogeneous and compositional goal space. Specifically, our environment features an achievement tree structure that explicitly defines the prerequisite relationships among goals, thereby encouraging agents to discover and exploit goal dependencies. The environment comprises 11 distinct goals, categorized into three types: (i) 5 movement-related goals requiring navigation to specific locations; (ii) 3 collection-based goals that require prior construction of the appropriate tools —e.g., collecting stone and coal is only feasible after crafting the corresponding equipment; and (iii) 3 crafting goals, which are achievable only when the agent possesses a sufficient quantity of the required resources in its inventory. This design introduces structured complexity that challenges agents to plan and act over extended temporal horizons.

Here is the list of all the goals and their requirements:

- go to tree: facing a tree
- collect wood: facing a tree, no tool needed
- place table: facing grass, sand, path and having 2 woods in inventory
- go to table: facing a table
- make wood pickaxe: facing a table and having 1 wood in inventory
- go to stone: facing a stone
- collect stone: facing a stone, having a wood pickaxe
- go to coal: facing a coal
- collect coal: facing a coal, having a wood pickaxe
- place furnace: facing grass, sand, path and having 4 stones in inventory
- go to furnace: facing a furnace

F.1.2 Modifications

We modify the original Crafter environment to isolate and emphasize the compositional aspects of goal-directed behaviour, removing elements primarily associated with survival dynamics. In particular, we eliminate hostile entities such as zombies and skeletons, as well as the need for the agent to eat or drink, thereby reducing perturbation-related distractions. To improve the agent's navigational capabilities, we augment the goal space with *"go to X"* objectives. These are always feasible for $X \in$ tree, stone, coal, but may not be realizable for $X \in$ table, furnace depending on the current world state, introducing ambiguity that

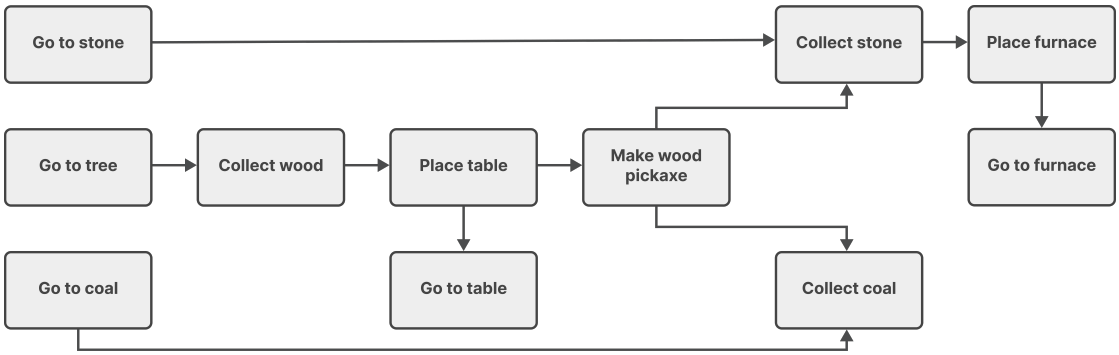


Figure F.1: Achievement tree used in our experiments

poses a challenge for the low-level success rate estimator (see Appendix F.6). To maintain computational efficiency, we limit the goal space to include up to the *"collect coal"* objective, which suffices to capture the dynamic nature of compositional task learning. Furthermore, we enrich the observation space by integrating a captioning module (see Appendix F.1.4). The hierarchical policy operates under temporal constraints: the high-level controller is allowed up to 64 steps (i.e., skill invocations), each of which may be executed by the low-level controller in up to 128 steps. To enable multi-goal episodes while ensuring bounded episode length, we cap the environment at 155 high-level steps before a reset occurs.

F.1.3 Goal difficulty

We introduce a principled method to quantify the difficulty of individual goals within our environment. The difficulty is defined with respect to an environment in its initial (reset) state, ensuring consistency across evaluations. To compute this metric, we decompose each successful trajectory into a sequence of intermediate subgoals, categorized as *"collect X"*, *"make X"*, and *"place X"*. Each *"collect"* operation is assigned a difficulty score of 1, reflecting the need to navigate to the target and execute a sequence of primitive actions. In contrast, *"make"* and *"place"* operations are assigned a lower difficulty of 0.5, as they typically require a single, correct atomic action given the appropriate preconditions. Importantly, goals such as *"go to wood"* and *"collect wood"* are considered equivalent in difficulty, since collecting wood necessitates no equipment—only proximity to a tree and execution of the elementary *"chop"* action. Similarly, *"go to X"* goals for $X \in$ wood, stone, coal are treated as having comparable difficulty, due to the widespread availability of these resources in the environment. In contrast, *"go to table"* and *"go to furnace"* are assigned the same difficulty scores as *"place table"* and *"place furnace"* respectively, since these locations do not exist by default and must be created through agent actions. Table F.1 gives the difficulty of each task and its decomposition.

F.1.4 Observations

The modified Crafter environment used in our experiments generates both a standard visual observation and a structured textual description. The visual observation consists of an RGB image of dimension $3 \times 32 \times 32$. The textual description acts as a sparse captioning of the

Table F.1: Table of difficulty scores

	Go to	Collect	Place	Make	Difficulty score
go to tree	1				0.5
go to stone	1				0.5
go to coal	1				0.5
collect wood		1			0.5
place table		2	1		2.5
go to table		2	1		2.5
make wood pickaxe		3	1	1	4
collect stone		4	1	1	5
collect coal		4	1	1	5
place furnace		7	2	1	8.5
go to furnace		7	2	1	8.5

visual input and is composed of multiple informative components, each providing structured context relevant to the agent's decision-making. Specifically, the textual description includes:

- A general sentence describing the overall nature of the game.
- The agent's current goal, along with the number of remaining high-level steps to achieve it.
- The player's current position in the environment.
- A description of all visible elements in the field of view, each identified by type and relative position using cardinal directions (e.g., "a tree is three steps north"). Each type of element is mentioned only once, even if multiple instances are visible.
- A description of the block directly in front of the player.
- A list of items currently in the player's inventory.
- The positions of environment elements that have been placed by the player.
- The available elementary actions.
- If applicable, the list of admissible skills that can be invoked by the high-level policy.
- The last action executed by the agent.

This detailed textual observation allows the high-level policy to operate with a semantically rich input space, facilitating more informed and context-aware decision-making. An example of such textual observation is given in Figure F.8 F.9.

F.2 Complementary Results

F.2.1 Evolution of success rate

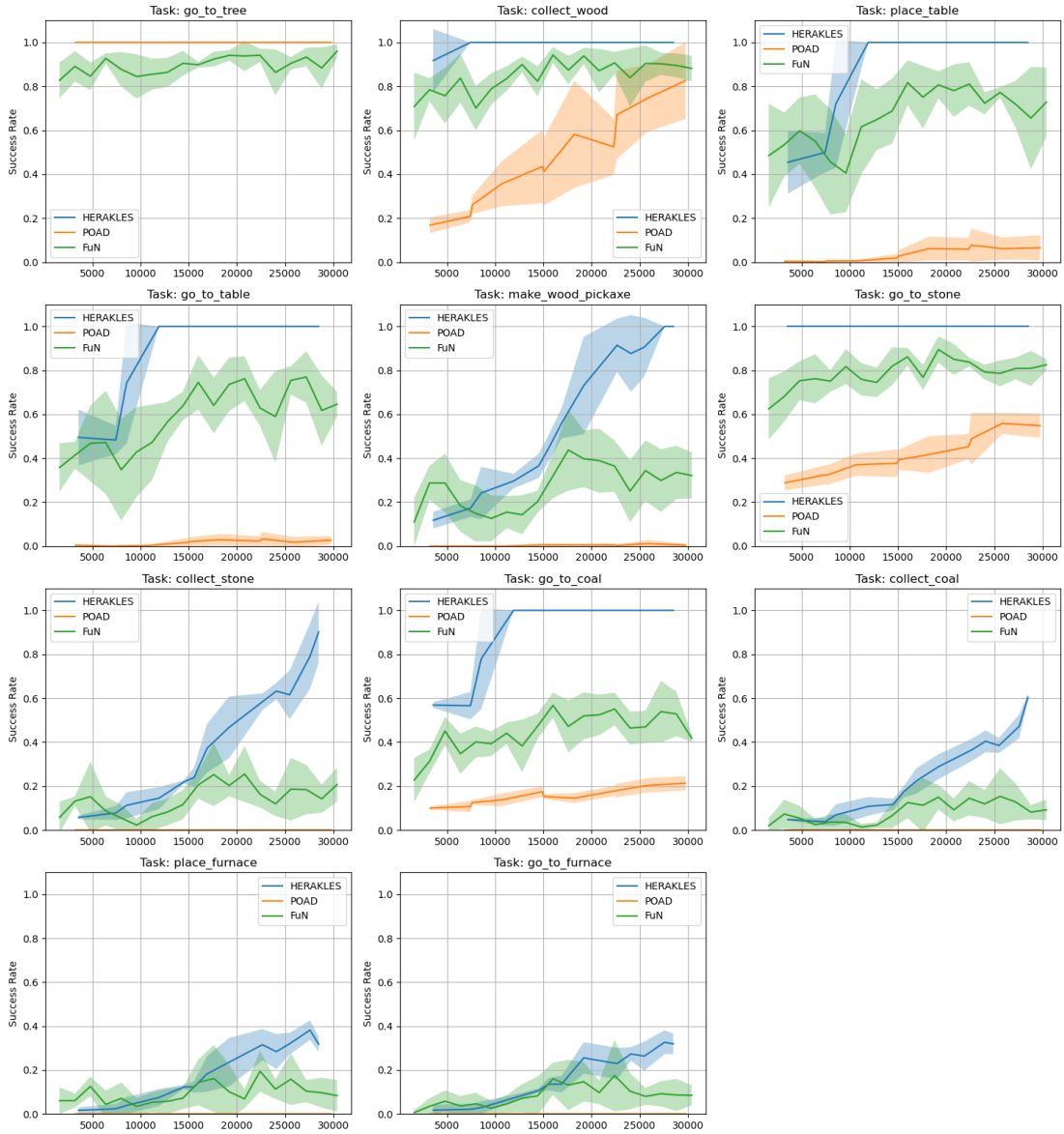
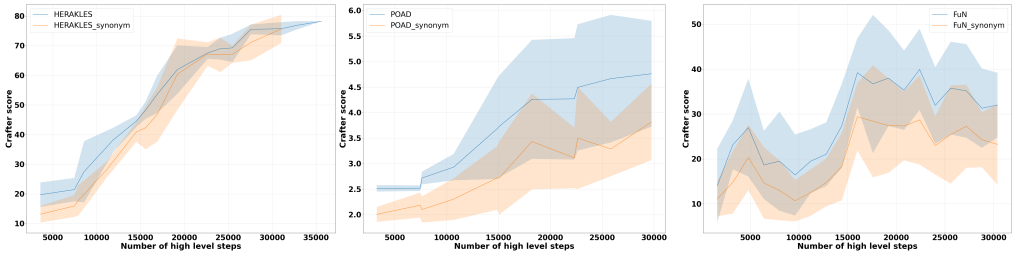


Figure F.2: Evolution of success rate per task

We analyze the evolution of the success rate (SR) for each method across training steps and observe that HERAKLES consistently outperforms the baselines POAD and FUN across all tasks and at every training stage. For simple navigation tasks such as *"go to tree"*, both POAD and HERAKLES achieve perfect performance from step = 0. This suggests that HERAKLES' hierarchical agent initially relies on its high-level policy, which is identical to that of POAD before the first update. Notably, HERAKLES exhibits rapid improvement on more complex compositional tasks. For instance, the SR for *"place table"* increases from 0.5

to 1.0 in fewer than 5000 steps once the prerequisite goal *"collect wood"* is mastered. Similar patterns are observed for other tasks such as *"make wood pickaxe"* and *"collect stone"*, where mastery of foundational subgoals accelerates progress. In contrast, when a baseline method successfully learns an individual task, this progress does not translate to improvements on related downstream tasks, indicating a lack of effective compositional generalization.

F.2.2 Generalization results



(a) HERAKLES (b) POAD (c) FuN

Figure F.3: Generalization synonym Crafter score

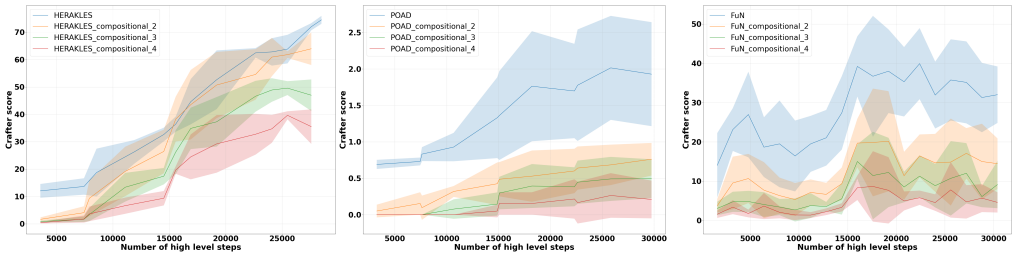
Figure F.3 shows the values of $S_c^{\text{syn}}(t)$ recorded every 2,496 training steps, compared to the $S_c(t)$ which uses original goals from G only.

As illustrated in Figure F.3, HERAKLES preserves a high Crafter score in the synonym goal space, demonstrating robust linguistic generalization.

Table F.2: Synonyms used for the synonym generalization experiment.

Original names	Synonyms		
collect	gather	acquire	procure
	harvest	amass	
make	craft	construct	build
	acquire	create	
place	put	putdown	install
	deploy	position	
go	move	walk	proceed
	travel	run	

n-Compositionality Figure F.4 illustrates that the Crafter score decreases for all methods as n increases, reflecting the growing complexity of the environment.



(a) HERAKLES (b) POAD (c) FuN

Figure F.4: Generalization compositional Crafter score

F.2.3 Evolution of high-level sampling strategy

HERAKLES leverages the continuous compilation of mastered goals into low-level skills, gradually transferring control from the hierarchical agent to the low-level controller. In this section, we analyze how this mechanism shapes the high-level policy's sampling strategy throughout training. According to the learning dynamics described in Section 13.4, we hypothesize a three-phase progression: initially, the high-level policy broadly samples elementary actions to guide the undertrained low-level policy; subsequently, as the low-level improves, skill calls become more frequent; and finally, the high-level converges to invoking the compiled skill corresponding to the goal directly, relying fully on the low-level policy for execution. For instance, for the goal "collect wood", the high level would sample the *"collect wood"* skill.

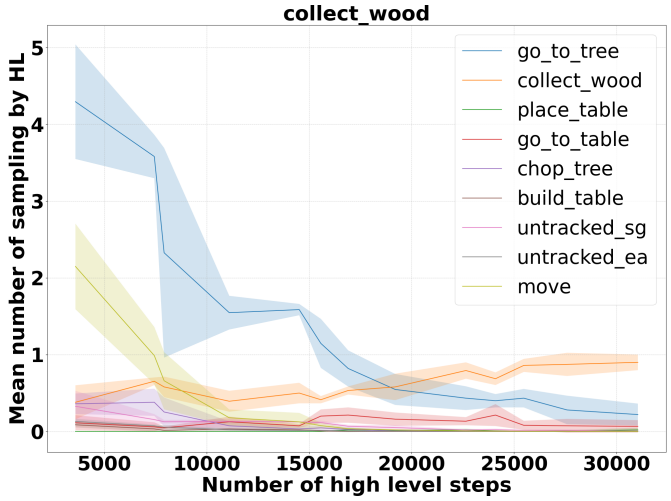
In Figure F.5, we report the mean number of skill or action calls per trajectory. For clarity, we aggregate all untracked compiled skills and untracked elementary actions into *untracked_sg* and *untracked_ea*, respectively, and average all *"move"* actions under a single *"move"* category.

Across the three subplots of Figure F.5, we observe that before 10,000 high-level steps, the policy predominantly samples elementary actions of type *"move"* and the skill *"go to tree"*. Since gathering wood is a prerequisite for nearly all goals, this early focus is expected. Notably, the skill *"go to tree"* is oversampled—approximately 10 times per trajectory—because the low-level policy has yet to master it, despite the skill being highly accessible (random agents succeed 90% of the time in 128 steps).

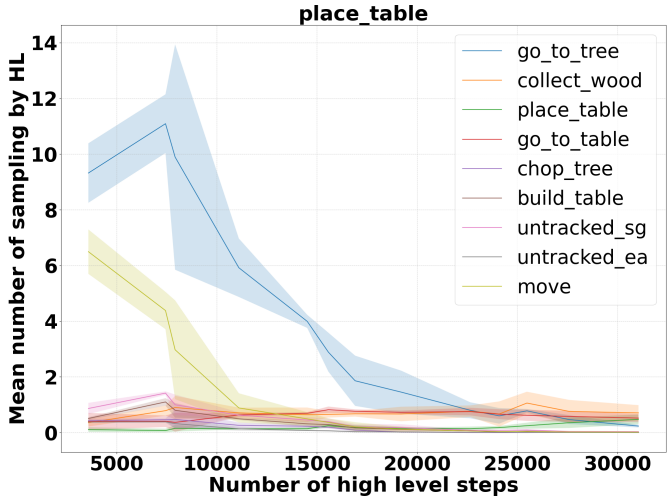
As shown in Figure F.5a, the elementary action *"chop tree"* is also sampled about 0.5 times per trajectory in the same early phase. Between 10,000 and 20,000 steps, elementary action calls decrease and the high-level policy increasingly delegates to compiled skills. This shift is evident in Figure F.5c, where *"collect wood"* and *"go to table"* are more frequently invoked.

By 25,000 steps, the high-level policy either directly selects compiled skills, as in Figure F.5a with *"collect wood"*, or replaces simpler skills with more complex ones, as seen in Figures F.5b and F.5c, where *"collect wood"* replaces the earlier calls to *"go to tree"*.

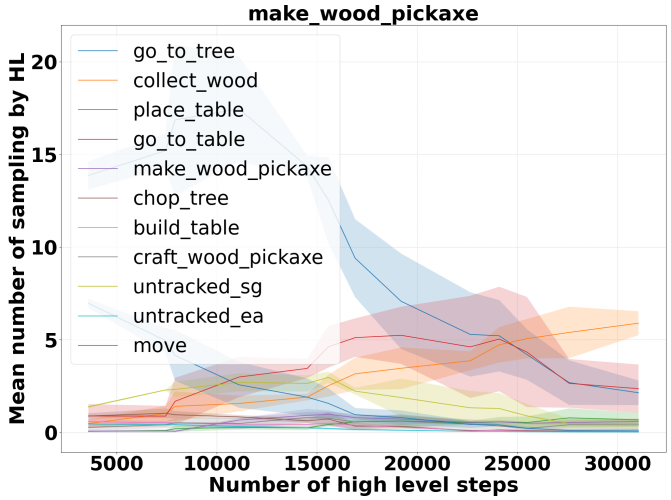
These observations confirm that HERAKLES adheres to the hypothesized learning dynamics, effectively leveraging the hierarchy to bootstrap new skills from previously acquired ones, thereby enabling efficient goal acquisition over time.



(a) Collect wood.



(b) Place table.



(c) Make wood pickaxe.

Figure F.5: Evolution of sampling strategy of high level.

F.3 Hierarchical Agent

F.3.1 HERAKLES implementation

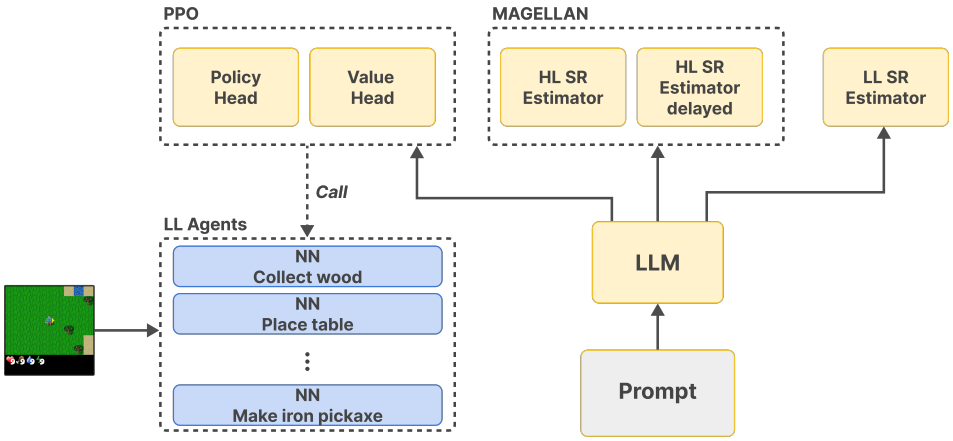


Figure F.6: HERAKLES architecture.

To instantiate the hierarchical agent described in Section 13.4, we employ a large language model (LLM)-backboned agent as the high-level policy (detailed in Appendix F.4) and a collection of lightweight neural networks as the low-level policy (see Appendix F.5). Each low-level network is specialized in executing a single skill, effectively mitigating catastrophic forgetting across the skill set. The Figure F.6 gives an overview of the overall architecture.

In accordance with the formulation in Section 13.4, we assume that $\forall(a, s) \in A \times S, ; \pi^{LL}(a|s, a) = 1$. Consequently, when the high-level policy selects a skill that corresponds to an elementary action—e.g., *"move right"*—we bypass the low-level controller and directly execute the action in the environment.

HERAKLES further incorporates a low-level success rate estimator to manage skill inclusion in the filtered skill set $\tilde{G}$, and leverages MAGELLAN to guide goal sampling. These components are described in Sections F.6 and F.7, respectively.

Each module—the high-level policy, the individual low-level networks, the success rate estimator, and MAGELLAN—is trained independently using separate Adam optimizers. Detailed hyper-parameter configurations for each optimizer are provided in Appendix F.9.

To help reproducibility, we make our code accessible at: `https://github.com/AnonymousUser530/HERAKLES`.

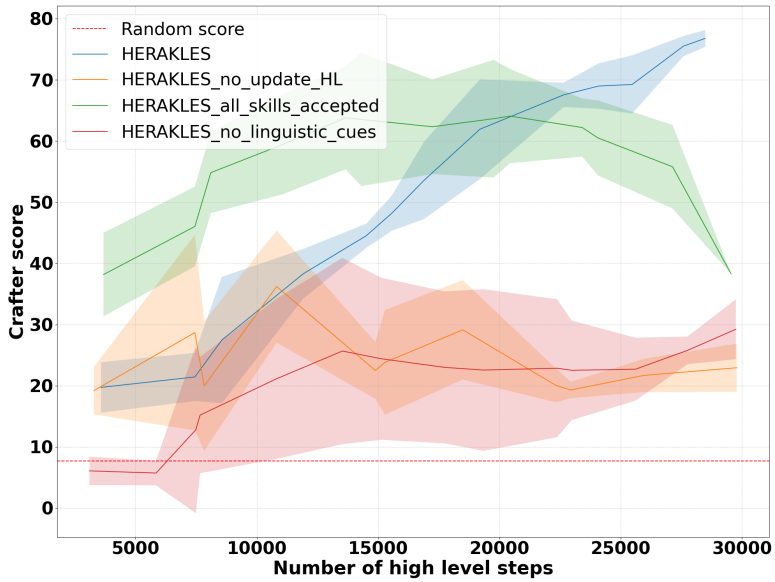


Figure F.7: Evolution of Crafter score for several ablations of HERAKLES.

F.3.2 HERAKLES ablations

We analyze the role of key architectural choices in HERAKLES by comparing its performance to several ablated versions. First, we evaluate the impact of updating the high-level policy by introducing HERAKLES_NO_UPDATE_HL, where the LLM-based high-level policy remains fixed throughout training. Next, we assess the necessity of the low-level success rate estimator by comparing to HERAKLE_ALL_SKILLS_ACCEPTED, where all skills are perpetually available to the high-level policy regardless of the low-level's competence. Finally, we test the role of linguistic grounding by substituting skill names in the high-level policy prompt with arbitrary identifiers of the form *skill* n, producing the HERAKLES_NO_LINGUISTIC_CUES variant.

In Figure F.7, we observe that HERAKLES_NO_UPDATE_HL mirrors the Crafter score trajectory of the full model during the initial 10,000 steps, indicating that even an non-finetuned LLM can benefit from improved low-level skills. However, its performance plateaus shortly thereafter, stabilizing between scores of 20 and 30. This result highlights the necessity of updating the high-level policy to maintain alignment with the evolving capabilities of the low-level controller.

HERAKLES_NO_LINGUISTIC_CUES begins with performance comparable to a random agent, unable to leverage textual information to guide skill selection. Over time, it improves moderately, reaching a Crafter score of approximately 25 as it empirically learns the use of certain skills. This underscores the importance of semantic cues in the skill space for sample-efficient learning.

Interestingly, HERAKLES_ALL_SKILLS_ACCEPTED starts with a higher Crafter score (40) than the full model (20). Manual inspection reveals that the high-level policy exploits the availability of unmastered skills by effectively executing long, random trajectories—up to 64 high level skill calls of 128 steps each (by the low level)—thereby occasionally achieving

incidental goals. Nevertheless, this approach saturates quickly, plateauing at a Crafter score of 65. The performances even decrease after 20,000 steps as the low level policy improves and makes shorter trajectories, then rending this "long random exploration" strategy useless. In contrast, HERAKLES, by selectively invoking truly mastered skills, continues to improve and surpasses a score of 80 without signs of convergence. This confirms the value of dynamic skill selection and the structured exploitation of learned behaviours.

F.4 High-level policy

F.4.1 High-level policy architecture

The high-level policy architecture in HERAKLES builds upon GLAM Carta et al. (2023), which employs an actor-critic framework grounded in a large language model (LLM) backbone. The value function is implemented as a two-layer multilayer perceptron (MLP) with 1024 sigmoid-activated units, applied to the final hidden state of the decoder. Both the policy and the value head utilize shared LoRA adapters Hu et al. (2022), enabling parameter-efficient fine-tuning of the LLM.

In contrast to GLAM, skill selection in HERAKLES is achieved through constrained decoding over the set of admissible skills $\tilde{G}$. Instead of sampling tokens from the entire vocabulary, token generation is restricted to sequences that correspond to valid skills in $\tilde{G}$, ensuring both syntactic validity and semantic precision.

Training of the high-level policy is conducted using POAD Wen et al. (2024b), an algorithm that adapts PPO to operate at the token level for language agents. The policy is updated every 2496 high-level transitions, using POAD with 4 training epochs per update cycle. In addition to the POAD loss we add a penalty term based on Kullback-Liebler divergence:

$$\beta_{KL} KL(\pi^{HL}, \pi^{HL}_{original}),$$

with β_{KL} an hyper-parameter controlling for the importance of this penalty term. It ensures that the high-level policy does not drift too much from the original policy and retains some plasticity, allowing it to adapt to the apparition of new usable skills during training.

F.4.2 Prompts for the High-Level policy

In this section, we provide an illustrative example of a prompt given to the high-level policy in HERAKLES. In the depicted scenario, the environment has not been reset since the previous interaction, a new goal, *"make furnace"* has been sampled and the agent has already done one high level step. The prompt includes a description of the current state, the previously executed skill, and the list of admissible skills $\tilde{G}$. This setup allows the high-level policy to condition its next action on both the recent interaction history and the updated goal. Such prompt structure ensures that the LLM-based policy can contextualize its decision-making process in a temporally coherent and semantically grounded manner.

Example Prompt for the high level policy (1/2)

You are playing a Minecraft like game. You can use elementary actions or, if available, more efficient low-level policies.
Your task: Make to furnace in 64.0 steps
You have already done 1 step.
Your coordinates: (29,10)
You see:

- grass 1 step to your west
- path 5 steps to your south-west
- sand 4 steps to your south-east
- tree 3 steps to your east
- table 1 step to your north
- plant 6 steps to your south-west

You face table at your front.
Your inventory:

- sapling: 9
- wood: 4
- coal: 2
- wood pickaxe: 4

You placed table at (4,3) You placed table at (23,13) You placed table at (29,9) You placed plant at (0,12) You placed plant at (0,0) You placed stone at (13,0)
Elementary actions you can take:

- move left
- move right
- move up
- move down
- sleep
- consume cow
- consume plant
- attack zombie
- attack skeleton
- attack cow
- chop tree (require facing tree)
- chop bush (require facing bush)

Figure F.8: Example of a prompt given to the agent high level policy (1/2).

F.5 Low-level policy

Example Prompt for the high level policy (2/2)

- chop grass (require facing grass)
- extract stone (require 1 wood pickaxe in your inventory while facing stone)
- extract coal (require 1 wood pickaxe in your inventory while facing coal)
- extract iron (require 1 stone pickaxe in your inventory while facing iron)
- extract diamond (require 1 iron pickaxe in your inventory while facing diamond)
- drink water
- put stone (require 1 stone in your inventory)
- build table (require 2 woods in your inventory)
- build furnace (require 4 stones in your inventory)
- put plant (require 1 sapling in your inventory)
- craft wood pickaxe (require 1 wood in your inventory while facing a table)
- craft stone pickaxe (require 1 wood, 1 stone in your inventory while facing a table)
- craft iron pickaxe (require 1 wood, 1 coal, 1 iron in your inventory while facing a furnace)
- craft wood sword (require 1 wood in your inventory while facing a table)
- craft stone sword (require 1 wood, 1 stone in your inventory while facing a table)
- craft iron sword (require 1 wood, 1 coal, 1 iron in your inventory while facing a furnace)

Low-level policies you can call:

- go to tree
- collect wood
- place table
- go to table

The last action you took: collect wood
Your action:

Figure F.9: Example of a prompt given to the agent high level policy (2/2) .

F.5.1 Low-level policy architecture

The low-level policy in HERAKLES is designed to be lightweight compared to the high-level language model and operates directly on raw sensory input, specifically RGB observations from the environment. To extract features and generate elementary actions, we adopt the

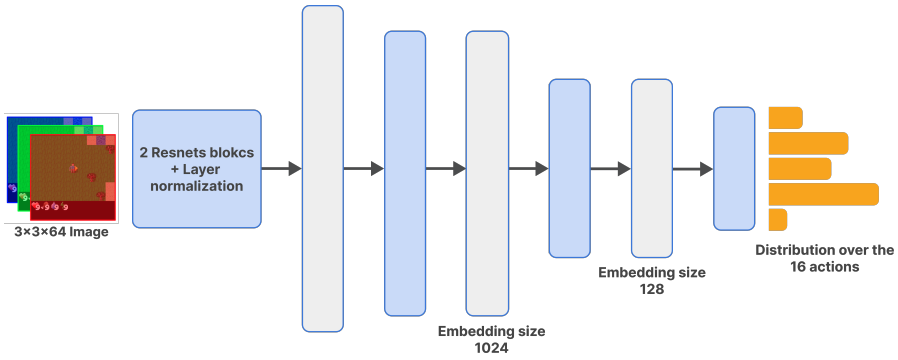


Figure F.10: Low-level policy architecture.

architecture proposed by Moon et al. (2023). This model employs a two-block ResNet He et al. (2015) with channel dimensions $[64, 128, 128]$ to encode visual information. The resulting feature map is flattened and passed through two fully connected layers of sizes 1024 and 128, respectively. The final embedding is then fed into a sigmoid-activated output layer of dimension 16 to produce a probability distribution over the discrete space of elementary actions. A low-level policy architecture diagram is given in Figure F.10. Each low-level agent contains approximately 2 million parameters, which constitutes only 0.03% of the total parameter count of the high-level policy, ensuring computational efficiency.

F.5.2 Low-level training

We reimplement the Advantage-Weighted Regression (AWR) algorithm Peng et al. (2019) to train the low-level policy in HERAKLES. AWR is selected for its simplicity and robustness, as well as its flexibility in integrating heterogeneous data sources into a unified replay buffer. Specifically, we leverage two types of transitions: (i) those collected from the low-level policy executing a given skill to enable continual skill refinement, and (ii) transitions sampled from successful trajectories of the hierarchical agent, used to compile successful behaviour into low-level policies.

This dual use of transitions allows for multiplicative relabeling, akin to the principle of hindsight experience replay (HER) Andrychowicz et al. (2017). For instance, a transition such as *"facing a tree and chopping wood"* can simultaneously serve to reinforce the skill *"collect wood"* and contribute to the higher-level objective *"place table"*. This mechanism enables the low-level agent to acquire new skills by imitating composite hierarchic behaviours, while also refining its competence on previously known skills.

We employ sparse binary rewards to signal the success of skill execution and goal achievement. In the case of failed trajectories—i.e., when a skill is not completed within the 128-step budget—we do not assign a reward of zero. Instead, we use the value predicted by the critic as a proxy reward, effectively smoothing the discontinuity introduced by the fixed time limit and providing a more informative learning signal. We update a low-level policy for the skill g every time 2496 new transitions have been added to the corresponding buffer.

F.5.3 Ablation of the AWR buffer size

We perform an ablation study on the size of the replay buffer used to train the low-level policy with the AWR algorithm. As illustrated in Figure F.11, a larger buffer capacity, accommodating up to 100,000 transitions, consistently yields slightly improved performance compared to smaller buffers limited to 25,000 or 50,000 transitions. This improvement can be attributed to the increased diversity of experiences available during training, which enhances generalization across skills. Based on this observation, we adopt a buffer size of 100,000 transitions in all our experiments.

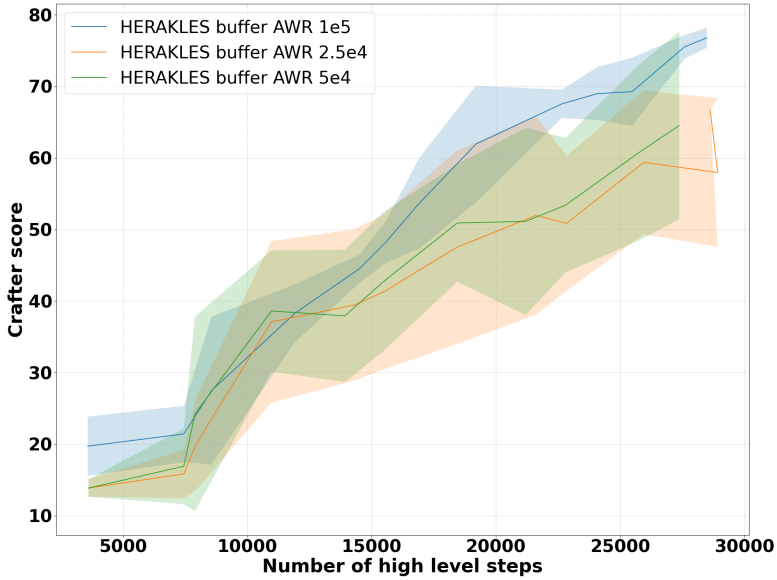


Figure F.11: Evolution of Crafter score for low-level policy updated with AWR using several buffer size.

F.6 Low-level success rate estimator

F.6.1 Prompt of the low-level success rate estimator

In this section, we give an example of prompts given to the low-level success rate estimator.

Example of prompt of the low-level policy success rate estimator

```
You are playing a Minecraft like game.  Your task is to evaluate your
success rate for the goal:  goal
Your coordinates:  (13,31)
You see:
    • water 7 steps to your north-west
    • grass 1 step to your west
    • stone 3 steps to your north-east
    • sand 3 steps to your north
    • table 1 step to your south
You face table at your front.
Your inventory:
    • sapling:  8
    • stone:  9
    • coal:  6
    • wood pickaxe:  3
    • wood sword:  1
You placed table at (39,40) You placed table at (3,32) You placed table
at (12,34) You placed table at (13,32) You placed table at (10,30) You
placed table at (37,18) You placed table at (40,17) You placed table at
(31,16)
Your success rate is:
```

Figure F.12: Example of a prompt given to low-level policy success rate estimator.

F.6.2 Training of the low-level success rate estimator

The low-level success rate estimator predicts the probability of successful skill execution given an observation. It is built upon an LLM backbone, from which we extract the final hidden state of the decoder. This representation is passed through two SiLU-activated hidden layers of size 1024 Elfwing et al. (2017), followed by a sigmoid-activated output layer of dimension 1, yielding a scalar success probability.

During execution, whenever the high-level policy selects a skill, we record a tuple $(\text{obs}_{\text{ll_sr_estimator}}, \text{skill}, \text{outcome})$, where *outcome* is a binary indicator of whether the skill was successfully executed by the low-level policy. To augment training data and enhance the estimator's ability to interpret

rich observations, we assume that starting from any of the initial 10% of states visited in a trajectory would not alter the success outcome. Accordingly, we collect $12 = \lfloor 0.1 \times 128 \rfloor$ such tuples per skill execution.

These transitions are aggregated during each high-level data collection cycle (consisting of 2496 high-level steps; see Appendix F.4) and stored in a buffer of size 3. The estimator is updated every 256 new transitions using a binary cross-entropy loss and trained for a single update epoch per cycle. These two hyper-parameters were selected based on empirical performance.

F.6.3 Composition of the set of skills

In Section 13.4, we define the probability of sampling a skill g into the admissible skill set $\tilde{G}$ at step k as:

$$p_g = \max\left(\mathbb{E}_{\pi^{LL}(\cdot|g),\, s_k \sim D_{\pi^{LL}}}\left[r^g \mid O(s_k)\right], \varepsilon_k\right),$$

where ε_k is an exploration term. We empirically found that ε_k should reflect the update frequency of the low-level policy corresponding to skill g. The rationale is that a frequently updated low-level policy is more likely to have improved, thus increasing the probability that the associated skill will succeed and should be selected.

We therefore define the exploration term as:

$$\varepsilon_k = \min(\text{update_frequency_g}, 0.1),$$

where `update_frequency_g` denotes the number of times the low-level policy for skill g has been updated in the past five high-level data collection cycles. This dynamic exploration term outperforms a fixed value —such as the constant $\varepsilon_k = 0.1$— in empirical evaluations.

F.7 Implementation of MAGELLAN

We reimplemented MAGELLAN from Gaven et al. (2025) to make our autotelic goal sampler prioritize goals with maximum Learning Progress (LP) from G. Our implementation slightly differs from the original one to match our framework.

First, the setup in MAGELLAN involved a single goal per episode. As a result, their goal space was the combination of all possible initial states and instructions (i.e., the Multi-Armed Bandit had one arm per initial state-instruction pair). Here, multiple goals can be solved within a single episode. Therefore, whenever a goal has to be sampled, MAGELLAN estimates the LP of each goal from G only for the current state. While this does not change much from a practical point of view, our bandit is now a contextual Multi-Armed Bandit (i.e., the set of arms never changes but the current state conditions the LP estimation).

Second, instead of storing a single entry per goal (i.e. the goal, the state it was sampled from, and its associated outcome) in MAGELLAN's dataset ($\mathcal{D}$ in their paper), we store multiple copies of the same goal-outcome pair with different states seen during the trajectory. In particular, we store the first $N^{HL} * 0.1 = 6$ states observed by the high-level policy during the trajectory. This provides richer information for a goal and helps foster generalization.

Then, as a possibly varying number of entries are added to the dataset for each goal (i.e. up to 6), MAGELLAN's update frequency is now calculated based on the number of entries added instead of goals sampled. We perform this update every 128 new entries. Moreover, instead of keeping a dataset of fixed size, we use a dataset of varying size storing the entries from the last 3 data collection phases of the high-level policy (i.e. 2496 high-level steps, see Appendix F.4).

For the buffer of weights (named $\mathcal{B}$ in MAGELLAN), we only keep the last 3 success rate estimator copies. Finally, we use an exponential decay for ϵ with a rate of 3.34. All the other hyper-parameters not mentioned in the section are kept the same as in the original implementation of MAGELLAN.

F.7.1 Prompt of the high-level success rate estimator

We show in F.12 the prompt given to the LLM when estimating the competence, i.e. the success rate.

Example Prompt high-level policy success rate estimator

```
You are playing a Minecraft like game.  Your task is to evaluate your
success rate for the goal:  <goal>
Your coordinates:  (24,8)
You see:
```

- grass 1 step to your west
- stone 2 steps to your west
- path 2 steps to your north-east
- coal 2 steps to your south-west
- iron 7 steps to your south-west
- table 3 steps to your south-east
- furnace 1 step to your south

```
You face a furnace at your front.
Your inventory:
```

- sapling: 6
- stone: 5
- coal: 1
- wood pickaxe: 3
- wood sword: 1

```
You placed table at (26,36) You placed table at (16,24) You placed table
at (26,9) You placed table at (24,0)
You placed plant at (1,0) You placed plant at (4,0)
You placed stone at (5,17) You placed stone at (3,17) You placed stone at
(2,18)
You placed furnace at (24,9)
Your success rate is:
```

Figure F.13: Example of a prompt given to high-level policy success rate estimator. This prompt is given for each goal in $g \in G$ with <goal> replaced g.

F.8 Baselines details

F.8.1 FuN

We implemented FeUdal Networks (FuN) a hierarchical RL architecture with two components: a Manager and a Worker, both modeled as recurrent networks. The Manager operates at a lower temporal resolution, producing directional goals $g_t \in \mathbb{R}^d$ in a learned latent state space. The Worker receives these goals and produces elementary actions at every time step.

Observations x_t are encoded into latent states z_t via a shared perceptual module. The Manager computes a latent embedding s_t, and generates g_t using a dilated LSTM that supports long-range dependencies. The Worker pools recent goals, maps them into a low-dimensional space via a projection ϕ, and combines them with action embeddings U_t to

produce the policy:

$$\pi_t = \text{SoftMax}(U_t \, \phi \left(\sum_{i=0}^{c-1} g_{t-i} \right))$$

The Manager is trained using a *transition policy gradient* to align g_t with advantageous latent transitions $s_{t+c} - s_t$, while the Worker is trained via intrinsic reward based on goal-following:

$$r_t^I = \frac{1}{c} \sum_{i=1}^{c} \cos(s_t - s_{t-i}, g_{t-i})$$

and optimizes a weighted sum of intrinsic and extrinsic rewards.

In the original FuN implementation, only environment observations were passed to the perceptual module, as the evaluated environments were not goal-conditioned. In our version, we concatenate an embedding of the goal to the output of the perceptual module. These goal embeddings are generated using the same language model employed for HERAKLES (Mistral 7B v0.3), enabling the agent to generalize to goals it has not encountered during training.

Table F.4 summarizes the key hyper-parameters used in our implementation of FuN.

Table F.3: Hyperparameters used in the FuN implementation.

Parameter	**Value**
Worker goal horizon	10
Hidden dimension of Manager	256
Hidden dimension of Worker	16
Discount factor for Worker	0.99
Discount factor for Manager	0.999
Intrinsic reward coefficient	0.5
Dilation factor for Manager's LSTM	10

F.8.2 POAD training

The POAD baseline Wen et al. (2024b) corresponds to a simplified version of HERAKLES in which the hierarchical structure is removed by discarding the low-level policy. In this configuration, all elementary actions are directly selected by the high-level policy. The training procedure strictly follows the methodology and hyper-parameters detailed in Appendix F.4.

F.9 Hyper-parameters

Table F.4: Hyper-parameters used in HERAKLES.

Parameter	**Value**
high level policy max steps	64
low level policy max steps	128
number of environment in parallel	48
number of high level step of data collection for POAD	2496
lr_{POAD}	10^{-5}
λ_{POAD}	0.9
γ_{POAD}	0.95
entropy coefficient POAD	0.01
β_{KL}	0.1
update epoch POAD	4
AWR buffer size	10^5
lr_{AWR}	10^{-4}
$lr_{\text{low level success rate estimator}}$	10^{-4}
udpate epoch low level success rate estimator	2

List of Figures

1.1 The reinforcement learning framework from Sutton & Barto (2018b). 5
1.2 Classical illustration of the Transformer architecture from Vaswani et al. (2017). The original design follows an encoder–decoder structure, as shown here, whereas the GPT series retains only the (causal) decoder component. 14

4.1 **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). 34
4.2 **Q1. Sample efficiency**: Evolution over 4 seeds of the average success rate and standard deviation on all Q1 tasks. The details of average success rate calculation over the goals is given in Appendix A.2.2 41
4.3 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. . . 42
4.4 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. 42

4.5 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. 43
4.6 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. 43
4.7 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. . 43
4.8 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. 44
4.9 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. 45
4.10 Impact of the aciton space size and number of distractors on the sample efficiency measure (Equation (4.3)). We report results averaged over 2 seeds for training on the *Go To* task. 45
4.11 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. 47
4.12 **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). The details of average success rate calculation over the goals is given in Appendix A.2.2. 48

5.1 Our framework aims at improving an LLM's ability to predict the outcome of state-action pairs by augmenting it with natural language hypotheses about the world (*Statistician*). WorldLLM alternates between generating hypotheses (*Scientist*) and collecting evidence to update these hypotheses (*Experimenter*). The former uses Bayesian inference with an LLM as the proposal distribution. The latter uses curiosity-driven RL to train an agent to collect experiences that have a low likelihood under the *Statistician* with the current set of hypotheses. . 54
5.2 *Our experimental setup.* We use the Playground-Text environment which features objects the agent can interact with along with textual observations and actions (a). We focus on 4 object types from Playground-Text: *Water*, *Plants*, *Small Herbivores*, and *Big Herbivores*. We show in (b) the technology tree for combining objects. 59
5.3 Evolution of log-likelihodds computed by the Statistician (using the Scientist's last accepted hypotheses) on the test set throughout WorldLLM's iterations when using different oracles as Experimenters. We average per transition type over 8 random seeds with the standard deviation represented by the error bars. We show at iteration 0 the log-likelihood from the Statistician without any hypotheses given. 60
5.4 Proportion of collected transitions for RL-ALPEXP (left), RL-LogP (middle) and RL-ALP (right). The colored dashed lines correspond to the amount of transitions collected by O-Hardest. 62
5.5 Log-likelihoods (averaged by transition type) produced by the Statistician when facing an instance of a *Grow Plant* transition. We show how using the generalization environment from Section 5.5.2 affects the log-likelihoods (b). 65

8.1 **Navigating large goal spaces with MAGELLAN**: During training, our LLM agent uses MAGELLAN to estimate its past and current competence to compute Absolute Learning Progress (ALP) on each goal. Given the per-goal ALP, the LLM agent's goal selector chooses the next goal to practice proportionally to their ALP. The LLM agent then performs a trajectory to achieve this goal and the outcome is used to update both the LLM agent with online RL and MAGELLAN's competence estimation. 73
8.2 A) Little-Zoo's tech tree. B) Little-Zoo's goal space is composed of all the possible combinations between instructions and objects that can be in the scene. Most object configurations make an instruction infeasible (e.g. "grow lion" is impossible with the second configuration, as water, needed to obtain plants, is missing). C) Little-Zoo provides a textual description that is given in our LLM agent's prompt. 81
8.3 Scaling of competence estimation error and competence estimation cost (i.e. total number of additional evaluation episodes) when increasing the goal space size. . 82
8.4 Competence estimation on OpenR1-Math-220k. MAGELLAN (blue) accurately tracks competence across Algebra, Geometry, and Number Theory, closely matching true success probabilities and outperforming Online-ALP (orange). 83

8.5 Evolution of the observed competence (SR) when evaluating policies on 64 training goals per category every 5000 episodes. We report the average SR over evaluated goals along with standard deviation (8 seeds). Icons indicate the average time step at which a method mastered a goal (i.e. SR $> 90\%$). We add stars to MAGELLAN, denoting significantly earlier mastery of a category compared to the method with the star's color (p-value $< 8 \times 10^{-4}$). The dotted line (EK-Online-ALP) indicates that the method relies on expert knowledge. 84
8.6 Goal sampling strategies of MAGELLAN, EK-Online-ALP, Online-ALP. We do not take into account the 20% uniformly sampled goals from the ε-greedy exploration. 84
8.7 MAGELLAN's LLM embedding space displayed using t-SNE with goals used in Q2 (Train) and Q3 (Test), along with the estimated success probability and linear interpolation between goals. We show the embedding space for a single seed (a) before training and (b) at the end of the 500k training steps. We see that impossible goals have been left aside, and that the other goals with a high estimated success probability are clustered consistently. 86
8.8 **Adaptation tests**: Using a single's seed training of 500k episodes, we stop and replace all goals with unseen ones every 50k episodes. We then resume training and sample goals using each method for 50k training episodes. We show two isolated and representative points of goal replacement: (a) there is no ALP on any goal (after 50k training episodes), and (b) some goals (here, "Grow carnivores" after 150k training episodes) have a high ALP. We report the evolution of SR when evaluating the policies on 64 goals per category from the new training set every 5000 episodes. Results show the average competence over evaluated goals along with standard deviation (8 seeds). 86
8.9 Plot of the t-SNE-projected embeddings of impossible goals, where each goal is color-coded based (a) on the type of impossible goal (b) on the specific missing element required to make it possible. 87

9.1 The autotelic agent architecture consists of two main modules: the goal generator and the goal-conditioned learner, along with a goal archive. In the goal generator, goals are sampled based on their learnability. These goals, their learnability (L) and difficulty (D) estimates, are given to LLM to generate new goals: reward functions (R) and names (nm). Names are embedded for goal conditioning of the RL agent. In the goal-conditioned learner, goals for training the RL agent are sampled based on Learning Progress (LP), and their learnability (L) and difficulty (D) are estimated. 92
9.2 Evolution of cumulative learnability, diversity and difficulty with the number of data collection turns. 97
9.3 Cumulative learnability of all goals generated by our method compared to the OMNI-EPIC-like baseline. 98
9.4 t-SNE of the goals embeddings contained in the archive at each generation. . . . 98
9.5 Examples of generated reward functions, pseudocode, and visualizations of the observations at the task completion state. Complexity increases from a)-c), but implementations semantically limited (shelters are not enclosed). In d), a more standard (less abstract) minecraft-like goal is implemented well, despite being multi-step. 99

9.6 Ablation between guided generation (using examples from similar predefined categories) and free generation (where the examples in the prompt are not group by categories). 100

12.1 During training, the agent uses the goal to generate relevant questions using its question-generation module **QG**. Then, it attempts at answering them from current trajectories at each step with its question-answering module **QA**, by looking at the trajectory. When it succeeds, it obtains an intrinsic reward proportional to its confidence in its answer. Then it removes the answered questions from the list of questions. This incentivizes the agent to produce trajectories that enable to reconstruct unambiguously partial information about the general language goal, enabling to shape rewards and guide learning. 108
12.2 Average reward for EAGER and baselines for four tasks, with error regions to indicate standard deviation over four random seeds. For the PutNext-Local and Sequence-Medium tasks, we give an example of possible tasks with environment at time step 0. 116
12.3 Average trajectory for PutNextTo-Local tasks for EAGER and ELLA. The colour of the tiles depends on the average number of passes of the agent on the tile. The 5 environments presented here have been randomly selected. The position of the agent is its position at the beginning of the episode. 119
12.4 Average reward (left) and success rate of the QA (right) for the PutNextTo-Local tasks. The agent are trained with QA having different success rate after pre-training. 120
12.5 Average reward in tasks PutNextTo and Sequence for two distributions of bot trajectories used to train QA. A narrow and a wide distribution where noise is added to bot trajectories 120
12.6 Average reward in PutNextTo with various ablations of the QA module. 120
12.7 Four examples and possible question at time step $t = 0$ where the QA could guess the answer either linguistically or by elimination process. In all of these examples the QA returns no_answer because the trajectory does not correspond to a possible answer 121

13.1 **HERAKLES** is an autotelic hierarchical RL agent which relies on two policies. An LLM-based high-level policy iteratively selects goals for a low-level policy which interacts with the environment through elementary actions to solve them. In particular, the high-level policy only selects goals among the ones with high success probability by learning to estimate the probability of success of the low-level policy on each goal. As HERAKLES masters a goal, the full trajectory (i.e. all elementary actions used to accomplish the sequence of goals produced by the high-level policy) is compiled and learned as a new single skill by the low-level policy. This new skill can then be directly used by the high-level policy. 128

13.2 **Skill learning and compilation in HERAKLES**: Given a goal g and a state s_0, the high-level policy generates its skill space $\tilde{\mathcal{G}}_0$ and samples one skill $\tilde{g}_0$. This skill is then sent to the low-level policy which performs k elementary actions in the environment to achieve $\tilde{g}_0$, leading to state s_k. Given s_k and g, the high-level policy then generates a new skill space, samples a new skill g_k and sends it to the low-level policy. This process repeats until g has been achieved or the limited number of high-level steps have been reached. This process produces both multiple low-level trajectories (i.e. one per skill sampled by the high-level) and a single high-level trajectory (i.e. the sequence of skills sampled). The high-level policy is trained on the high-level trajectory to improve skill selection. The low-level policy is trained on all the low-level trajectories, conditioned to reach the associated skill. Crucially, we perform **skill compilation** by also training the low-level policy on a trajectory that is the concatenation of all the low-level trajectories and that is conditioned to reach the high-level policy's goal g. Doing so, our low-level policy progressively internalizes the sequence of elementary actions required to directly solve g. 130
13.3 Average number of low level steps to master a goal as a function of the difficulty of that goal. Error bars account for standard deviation over 4 seeds. HERAKLES achieves harder goals much faster. 134
13.4 Number of goals reached as a function of the number of high level steps. Shaded area standard deviation based on 4 seeds. HERAKLES is the only method whose number of goals grows linearly with time. 134

A.1 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. 211
A.2 Per-task success rate for the 1000 evaluation trajectories performed in Figure 4.12. 213
A.3 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. 214
A.4 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. 215
A.5 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. 216
A.6 Evolution of actions' probability over training for test prompts listed in Table A.1. 223

B.1 An example of a trajectory in Playground-Text. 233
B.2 Prompt used for the Statistician. The <hypothesis> corresponds to the hypotheses to test. The <state>, <action> and <change> represent the current state, the action taken, and the resulting change, respectively, for the given transition. 234
B.3 Prompt used for the Scientist. <trajectories> correspond to the trajectories collected by the Experimenter, <previous hypothesis> corresponds to the set of hypotheses that have been accepted at the last iteration and <worst trajectories> to the trajectories where the previous hypotheses obtained the worst log-likelihood. 235
B.4 Proportion of collected transition types for each oracle. We show on the left the distribution of our stationary oracles and on the right the evolving distribution of O-Curriculum. 235

B.5 Changes in the observations for the generalization environment. 237
B.6 Evolution of log-likelihoods on the test set for all methods. 238
B.7 Evolution of the hypotheses retention rate for the different Experimenters. . . . 239
B.8 T-SNE embeddings obtained using the *all-mpnet-base-v2* sentence transformer on the hypotheses generated by different Experimenters. 240
B.9 Examples of accepted hypotheses. 242
B.10 The evolution of rewards obtained by the RL agent when using RL-ALP for a single seed. We plot the sum of rewards obtained for each PPO update. 243
B.11 Transitions collected by the Experimenter when using RL-ALPEXP for $\alpha = \{0.6, 0.75, 0.9, 0.95\}$ respectively. 243

C.1 We first generate the full goal space whom the distribution is given Figure C.1a then for computational reasons we sample a smaller goal space with the following distribution Figure C.1b. 248
C.2 Different architectural choices in MAGELLAN: (a) we learn separate LoRA adapters between the policy and MAGELLAN (used in the paper); (b) we share adapters and update them using both the policy and MAGELLAN gradient; (c) we share adapters but they are only updated by the policy gradient; (d) MAGELLAN directly uses the latent representation produced by the pretrained LLM. 255
C.3 Training curves of the four different possible architecture for MAGELLAN. We use 8 seeds to plot the mean and the standard deviation (shadow area around the solid line). 256
C.4 The LLM embedding space of MAGELLAN displayed using t-SNE with goals used in Q2 (Train) and Q3 (Test), along with MAGELLAN's estimated success probability and linear interpolation between goals. We show the embedding space for a single seed for the four architectures described in Appendix C.4.1 at the end of the 500k training episodes. 256
C.5 Evolution of competence estimation for each ALP method on each goal category for 25k goals. We show the average competence and its standard deviation across 8 seeds that use EK-Eval-ALP to sample goals. 257
C.6 Evolution of competence estimation for each ALP method on each goal category for 50k goals. We show the average competence and its standard deviation across 8 seeds that use EK-Eval-ALP to sample goals. 257
C.7 Evolution of competence estimation for each ALP method on each goal category for 100k goals. We show the average competence and its standard deviation across 8 seeds that use EK-Eval-ALP to sample goals. 258
C.8 Competence estimation on BabyAI-Text. MAGELLAN accurately tracks competence across five goal types of increasing difficulty and consistently outperforms Online-ALP. 258
C.9 Effect of LLM choice on MAGELLAN's competence estimation for OpenR1-Math-220k. All models yield similar performance. 259
C.10 Effect of LLM choice on MAGELLAN's competence estimation for BabyAI-Text. Performance remains consistent across models. 259
C.11 Evolution of average SR for each ALP method for each goal category. To get the success rate within a category, we evaluate the policy on 64 goals of this category. We use 8 seeds to plot the mean and the standard deviation. 260

C.12 Per-goal estimation of the success probability for each method. The average result (over 8 seed) is the solid line and the shaded zone represents the standard deviation. . . . 261
C.13 Chronogram of the embedding space of the seed 0, at the beginning and after mastering each type of goal. . . . 263
C.14 **Adaptation tests**: Using a single's seed training of 500k episodes, we stop and replace all goals by new unseen goals every 50k episodes. We then resume training by sampling goals using each of our four methods' ALP estimation (MAGELLAN, EK-Online-ALP, Online-ALP, Uniform) and perform 50k training episodes. We report the evolution of observed competence (SR) when evaluating the policies on 64 goals per category from the new training set every 5000 episodes. We report the average competence over evaluated goals along with standard deviation. . . . 265
C.15 Average sample efficiency (after κ, the length of the test) of each method average over the 10 tests. . . . 266

E.1 The QA architecture is based on the architecture of the Episodic Transformer Pashevich et al. (2021). To predict the correct answer the QA model received a natural language question, visual observations, and actions generated by the agent from the beginning of an episode (see Figure E.3). Here we show an example that corresponds to the 5th time step of an episode. After processing the question with a transformer-based language encoder, embedding the observation with a convolutional neural network CNN and passing actions through a look-up table, the agent outputs the correct answer. During training, we use full trajectories. At test time, we ask all the questions in the active set of questions using the current trajectory. . . . 282
E.2 Choice of a new action during the generation of the wide distribution of trajectories. At each time step, the agent can perform a random action with probability p. . . . 283
E.3 The actor-critic architecture uses a multimodal encoder that mixes language and image using a feature-wise affine transformation FiLM Perez et al. (2017a). Then the encoded representation passes through an LSTM to process the history of observations. Finally, the output is used in an actor and critic heads. . . . 284
E.4 Average Reward for the Unlock-Medium task for EAGER, RIDE and the combination of EAGER and RIDE. . . . 285
E.5 Average Reward for EAGER and baselines. . . . 286
E.6 Ablation of the neutralization of the intrinsic reward on the task PutNextTo-local. 287
E.7 Average reward for Sequence . . . 289
E.8 Average reward over goals . . . 289
E.9 Gaussian process model to fit Sample Efficiency (SE) of the EAGER algorithm function of Success Rate (SR) of the QA. For each value of SR, the different points correspond to the different seeds used to train the RL agent on the PutNextTo task. 289

F.1 Achievement tree used in our experiments . . . 295
F.2 Evolution of success rate per task . . . 297
F.3 Generalization synonym Crafter score . . . 298
F.4 Generalization compositional Crafter score . . . 299
F.5 Evolution of sampling strategy of high level. . . . 300
F.6 HERAKLES architecture. . . . 301

F.7 Evolution of Crafter score for several ablations of HERAKLES. 302
F.8 Example of a prompt given to the agent high level policy (1/2). 305
F.9 Example of a prompt given to the agent high level policy (2/2) 306
F.10 Low-level policy architecture. 307
F.11 Evolution of Crafter score for low-level policy updated with AWR using several buffer size. 308
F.12 Example of a prompt given to low-level policy success rate estimator. 309
F.13 Example of a prompt given to high-level policy success rate estimator. This prompt is given for each goal in $g \in G$ with <goal> replaced g. 312

List of Tables

1 Abbreviations xix
2 Symbols xxi

4.1 Generalization tests for Behavioral Cloning 49
4.2 Test on tasks with temporal components 49

5.1 This table presents the normalized area under the training curve for each algorithm and transition type. The normalization is performed by dividing the computed area by the area formed by the initial log-likelihoods, i.e. obtained without using hypotheses. 61
5.2 Top 3 performance of Constrained Decoding for each algorithm by transition type. We also report the performance without any hypotheses given (No Hypo.). 63
5.3 Top 3 performance on Constrained Decoding for each algorithm by transition type on the test set using the generalization environment. We also report the performance without any hypotheses given (No Hypo.). 64

8.1 Comparison of ALP estimation methods. We use the following dimensions: computational Efficiency, competence Transfer tracking, and required Expert Knowledge. 80
8.2 We evaluate the policies trained with each ALP method on a held-out test set. We show the difference between the observed and predicted competence. Online-ALP's performance on "Grow carnivore" is simply explained by the fact that its policies never mastered this goal category. 85

12.1 All the assumptions and expert knowledge required for RIDE, EAGER, and ELLA 117

13.1 Generalization for n-compositional goals. 136
13.2 Generalization for synonyms goals. 137

A.1 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 219
A.2 Generalization tests 224
A.3 Complementary tests for Q2 224
A.4 Complementary tests for Q3 225
A.5 PPO hyperparameters 227

A.6 Adam hyperparameters . . . 227
A.7 Out-of-vocabulary substitutions for Nouns and adjectives . . . 230
A.8 Invented substitutions for Nouns and adjectives . . . 230
A.9 Synonym actions . . . 231
A.10 French translation . . . 231

B.1 PPO hyperparameters used for our RL agents . . . 236
B.2 Mean of the inertia score on the T-SNE embeddings across the different seeds for each Experimenter. . . . 240

C.1 The optimal trajectories per categories. . . . 247
C.2 Comparison of prior work w.r.t. their LP estimation approach. We use the following dimensions: computational efficiency, dynamical competence transfer assumptions and no required expert knowledge. . . . 250
C.3 SAC hyperparameters . . . 251
C.4 MAGELLAN hyperparameters . . . 252
C.5 Online methods hyperparameters . . . 252
C.6 Eval methods hyperparameters . . . 253

D.1 PPO transformerXL Hyperparameters . . . 269
D.2 Autotelic architecture hyperparameters . . . 269

E.1 Tasks description, ✓means that the task is used for training the QA resp. the Agent . . . 281
E.2 QA training hyperparameters . . . 291
E.3 PPO hyperparameters . . . 291
E.4 Value of λ depending on the task. . . . 291

F.1 Table of difficulty scores . . . 296
F.2 Synonyms used for the synonym generalization experiment. . . . 298
F.3 Hyperparameters used in the FuN implementation. . . . 313
F.4 Hyper-parameters used in Herakles. . . . 314