Towards Vygotskian Autotelic Agents:

Learning Skills with Goals, Language and Intrinsically Motivated Deep Reinforcement Learning

---

By **Cédric COLAS**

Under the co-supervision of **Pierre-Yves OUDEYER** and **Olivier SIGAUD**

In partial fulfillment of the requirements

for the degree of Doctor of Philosophy

University of Bordeaux

Graduate school of Mathematics and Computer Science

Major in Computer Science

Submitted on May 1, 2021. Defended on June 30, 2021.

Composition of the jury:

Dr. Marc BELLEMARE	Research Scientist & Adjunct Pr.	Google Research & McGill Univ.	Reviewer
Pr. Sebastian RISI	Full Professor	IT Univ. of Copenhagen	Reviewer
Pr. Laura SCHULZ	Full Professor	MIT	Examinator
Dr. Harm VAN SEIJEN	Principal Research Manager	Microsoft Research	Examinator
Dr. Felix HILL	Research Scientist	DeepMind	Examinator
Dr. Mehdi KHAMASSI	Research Director	CNRS	Examinator
Dr. Pierre-Yves OUDEYER	Research Director	INRIA	Director
Pr. Olivier SIGAUD	Full Professor	Sorbonne Université	Director

Contents

Acknowledgments

I would like to thank the following people, without whom I would not have been able to complete this research.

First and foremost, my supervisors Olivier Sigaud and Pierre-Yves Oudeyer, whose insight and knowledge guided me throughout these three years. Always available, they could contribute ideas, perspectives, pointers to the literature from the old days or from yesterday. Thanks for reviewing all these pages, the good and the bad ones.

The Flowers lab at INRIA Bordeaux, probably the best environment for a PhD, and all its members. I am especially grateful for exciting scientific discussions with Tristan Karch, Mayalen Etcheverry, Rémy Portelas, Alexander Ten, Sébastien Forestier, Alexandre Péré, Laetitia Teodorescu, Eleni Nisioti and Adrien Laversanne-Finot, among others.

All my co-authors, who helped me publish interesting work: Vashisht Madhavan, Joost Huizinga, Jeff Clune, Rémy Portelas, Lilian Weng, Katja Hofmann, Tristan Karch, Nicolas Lair, Clément Moulin-Frier, Jean-Michel Dussoux, Peter Ford Dominey, Pierre Fournier, Ahmed Akakzia, Mohamed Chetouani, Boris Hejblum, Mélanie Prague, Rodolphe Thiebaut, Sébastien Rouillon, Olivier Sigaud and Pierre-Yves Oudeyer. Details about these collaborations can be found in the introduction.

Timothée Anne and Tianwei Lan, two master students whom I hope I advised well.

Other people at INRIA, without whom I could not have done my summer internship, including Vanessa Michel, Nathalie Robin and Nicolas Roussel.

Colleagues at Uber AI Labs, from whom I learned a lot, including Kenneth Stanley, Adrien Ecoffet, Andrea Madotto, Sumanth Dathathri and Patrick von Platen.

My biggest thanks go to my friends and family who supported me during these three years, had to put up with my complaints, especially these last months, and finally convinced me that this manuscript would probably not be the worst text in recorded history.

Towards Vygotskian Autotelic Agents: Learning Skills with Goals, Language and Intrinsically Motivated Deep Reinforcement Learning

Abstract: Building autonomous machines that can explore large environments, discover interesting interactions and learn open-ended repertoires of skills is a long-standing goal in artificial intelligence. Inspired by the remarkable lifelong learning of humans, the field of developmental machine learning aims at studying the mechanisms enabling autonomous machines to self-organize their own developmental trajectories and grow their own repertoires of skills. This research makes steps towards that goal. Reinforcement learning methods (RL) train learning agents to control their environment by maximizing future rewards and, thus, seem adapted to our purpose. Although it achieved impressive results in the last decade—beating humans at video games, chess, go or controlling robotic agents—it falls short of solving our goal. Indeed, RL agents demonstrate low autonomy and open-endedness because they usually target a (small) set of pre-defined tasks characterized by hand-defined reward functions. In this research, we transfer, adapt and extend ideas from a developmental framework called *intrinsically motivated goal exploration process* (IMGEP) to the RL setting. The resulting framework builds on goal-conditioned RL techniques to design *autotelic RL agents*: agents that are intrinsically motivated to represent, generate, pursue and master their own goals as a way to grow repertoires of skills. The efficient acquisition of open-ended repertoires of skills further requires agents to creatively generate novel goals out of the domain of known effects (creative exploration), to readily generalize their understanding of known skills to similar ones (systematic generalization), and to compose known skills to form new ones (composition). Inspired by developmental psychology, we propose to use *language as a cognitive tool* to support such properties.

We organize the manuscript around these two notions: goals and language. The first part focuses on goals. It covers foundational concepts and related work on intrinsic motivations, reinforcement learning and developmental robotics before introducing our framework, *rl-based intrinsically motivated goal exploration process* (RL-IMGEP), the intersection of RL and IMGEPs. Building on this framework, we present three computational studies of the properties of autotelic agents. We first show that we can use autotelic exploration to solve external hard-exploration tasks (study 1: GEP-PG and 2: ME-ES). We then move on to reward-free environments and propose CURIOUS, an autotelic agent that targets a diversity of goals, transfers knowledge across skills and organizes its own learning trajectory by pursuing goals associated with high learning progress (study 3). The second part focuses on language. Inspired by the pioneering work of Vygotsky and others, we first discuss existing communicative and cognitive uses of language for goal-directed artificial agents. Language facilitates human-agent communications, abstraction, systematic generalization, long-horizon control, but also creativity and mental simulations. In two subsequent computational studies, we propose to implement these two last cognitive uses of language. IMAGINE uses language both to learn goal representations from social interactions (communicative use) and to imagine out-of-distribution goals used to drive its creative exploration and enhance systematic generalization (cognitive use). In our last study, LGB trains a language-conditioned world model to generate a diversity of possible futures conditioned on linguistic descriptions. This leads to *behavioral diversity* and *strategy-switching behaviors*.

Keywords: intrinsic motivations, goals, language, autotelic agents, open-ended learning, exploration, skill acquisition, creative exploration, reinforcement learning, deep learning

---

Equipe Flowers, Inria, 33000 Bordeaux, France.

Agents Autotéliques Vygostkiens: Buts, Langage et Apprentissage Intrinsèquement Motivé

Résumé : Concevoir des machines autonomes qui explorent des environnements larges, découvrent des interactions pertinentes et développent des répertoires de comportements non-bornés est un des défis majeurs en intelligence artificielle. Inspiré par le remarquable apprentissage de l'humain, l'apprentissage machine développemental étudie les mécanismes permettant aux machines d'auto-organiser leurs trajectoires développementales et de développer des répertoires de comportements. Notre recherche progresse vers ce but. *L'apprentissage par renforcement* (RL) entraîne des agents à contrôler leur environnement de sorte à maximiser des récompenses et apparaît donc adapté à notre objectif. Malgré ses récent succès—battre l'humains à certains jeux vidéos, aux échecs, au go ou contrôler des robots—le RL ne saurait être suffisant : les agents RL sont peu autonomes et montrent des comportements bornés car ils s'attaquent à de (petits) sets de tâches pré-définies, caractérisées par des fonctions de récompenses pré-codées. Dans cette recherche, nous proposons de transférer, d'adapter et d'étendre des idées issues d'une approche de robotique développementale appelée *processus d'exploration de buts intrinsèquement motivés* (IMGEP) aux méthodes de RL. Notre nouveau cadre algorithmique étend les techniques de RL conditionné par des buts pour développer des *agents RL autotéliques*: des agents intrinsèquement motivés à représenter, générer, poursuivre et maîtriser leurs propres buts en vue de développer des répertoires de comportements. L'acquisition efficace de répertoires de comportements non-bornés nécessite une génération créative de buts en dehors de la distribution des effets connus (exploration créative), la généralisation de comportements connus à des comportements nouveaux (généralisation systématique) et la capacité à composer des comportements connus pour en former de nouveaux (composition). Inspiré par la psychologie développementale, nous proposons d'utiliser le langage comme un outil cognitif de sorte à soutenir ces propriétés.

Ce manuscrit est construit autour de deux notions: les buts et le langage. La première partie se concentre sur les buts. Elle couvre les concepts fondamentaux et la littérature associée traitant des motivations intrinsèques, de l'apprentissage par renforcement et de la robotique développementale avant d'introduire notre framework: *les processus d'exploration de buts intrinsèquement motivés basées sur l'apprentissage par renforcement* (RL-IMGEP). À partir de ce cadre, nous présentons trois études computationnelles des propriétés des agents autotéliques. Nous montrons d'abord que l'exploration autotélique peut être utilisée pour résoudre des tâches nécessitant une importante exploration (étude 1: GEP-PG et 2: ME-ES). Nous proposons ensuite CURIOUS dans un environnement sans récompense: un agent autotélique qui vise une diversité de buts, transfère de l'information entre compétences et organise sa trajectoire d'apprentissage en poursuivant les buts liés à de forts progrès (étude 3). La seconde partie se concentre sur le langage. Inspirés par les travaux de Vygostky et d'autres, nous discutons des utilisations des capacités communicatives et cognitives du langage dans le cadre d'agents dirigés par des buts. Le langage facilite les interactions humain-agent, l'abstraction, la généralisation systématique, le contrôle à long horizon temporel, mais aussi la créativité et la simulation mentale. Dans les deux études computationnelles qui suivent, nous implémentons ces deux dernières capacités. IMAGINE utilise le langage pour apprendre des représentations de buts (usage communicatif) et pour imaginer de nouveaux buts de sorte à diriger une exploration créative (usage cognitif). Dans notre dernière étude, LGB entraîne un modèle du monde à générer une diversité de futurs possibles à partir de descriptions linguistiques. Cela mène à une plus grande diversité comportementale et à des comportements de changement de stratégie.

Mots-clés : motivations intrinsèques, buts, langage, agents autotéliques, apprentissage non-borné, exploration, acquisition de comportements, exploration créative, apprentissage par renforcement, apprentissage profond

Equipe Flowers, Inria, 33000 Bordeaux, France.

Introduction

Building autonomous machines that can explore large environments, discover interesting interactions and learn open-ended repertoires of skills is a long-standing goal in artificial intelligence (AI). We, humans, are remarkable examples of this lifelong, open-ended learning. We learn to recognize objects and crawl as infants, learn to ask questions and interact with peers as toddlers, and some even go on to revolutionize arts, science or sports as adults. Across our lives, we build large repertoires of diverse skills from a virtually infinite set of possibilities. Perhaps, what is most striking is our ability to invent and solve *our own problems*, using internal feedback to assess progress. For each new problem we invent, we can develop a new skill in interaction with our current repertoire and our physico-socio-cultural environment. Thus, although our developmental trajectories share similarities (e.g. most people learn how to walk), our specific experience makes each developmental trajectory unique. We want to build artificial agents developing open-ended repertoires of skills autonomously with such lifelong abilities. The present research aims to make progress towards that goal.

What would be the properties of such agents? First, they need to be **embodied and situated learners**, i.e., they must learn skills from experience collected through the interactions between their physical body and environment, ideally learning a lot from little experience. Second, they need to be **autonomous**, i.e., literally, they need to *make their own laws*. By *autonomous*, we mean that agents do not receive task-specific learning signals from engineers but learn them from data. Indeed, humans, for each new problem they invent, also generate their own problem-specific learning signals. Third, our agents must continually learn new skills in an **open-ended** fashion.

For these properties to emerge—autonomous, embodied, and situated learning of open-ended repertoires of skills—we need others to support them. Removing external task-specific reward functions makes agents autonomous but also passive and reactive. To demonstrate the curious, playful, exploratory behavior of children, artificial agents must be intrinsically motivated just like them. Intrinsic motivations can indeed drive agents to explore the world for their own sake, without any task-related objective. With these intrinsic learning signals, artificial agents can become *good explorers*; they can learn to gather the experience that is relevant for future learning and skill development. But this is not all. To learn skills, agents need to represent them cognitively and select which ones to practice here and now. We call this the **autotelic principle**—i.e., literally, the ability to generate one's own goals (Steels, 2004). Autotelic agents generate and pursue

their own goals, thereby selecting the skills they want to develop and master. Finally, the learning of open-ended sets of skills requires agents to be **creative**, **generalize** and **compose**. Agents can learn new skills if they can represent novel, creative goals out of the distribution of known ones and pursue them. Skill learning is itself more efficient when agents can transfer knowledge from one skill to another (generalization) and can compose known skills to form new ones (composition). In conclusion, to tackle the problem of the autonomous acquisition of open-ended repertoires of skills, we need to design *embodied, situated, intrinsically motivated, and autotelic agents that generate creative goals, generalize and compose skills.*

The present research builds on two existing frameworks: 1) *reinforcement learning* (RL), a set of methods training agents to maximize future rewards, and 2) *intrinsically motivated goal exploration processes* (IMGEP), a set of methods training intrinsically motivated, autotelic agents to pursue their own goals. We introduce a new family of architectures called *rl-based IMGEP* (RL-IMGEP) at their intersection—a set of methods training goal-conditioned policies to represent, generate, pursue and master their own goals with RL. These new algorithms overcome the limitations of existing implementations of IMGEP and RL to get us closer to the autonomous acquisition of repertoires of skills. The present research focuses on two aspects of the self-organization of learning trajectories. First, RL agents must represent, generate, pursue and master their own goals to grow repertoires of skills. Second, RL agents can use language as a cognitive tool to imagine creative goals and foster behavioral diversity. The remaining of this introduction unpacks these different concepts to motivate the present research and closes on a summary of its contributions. Let us first take a quick detour through the history of AI and discuss how past approaches fall short of addressing our problem.

Symbolic and Connectionist Machines

In *Thinking Fast and Slow*, the psychologist Kahneman pictures two types of human reasoning: the fast, distributed, and intuitive thinking of System 1 and the slow, sequential, and symbolic reasoning of System 2 (Kahneman, 2011). We can roughly map this decomposition to the two main currents in the history of AI: *symbolic AI* implements System 2 and *connectionist AI* implements System 1.

Following his intuition that "*intellectual activity consists mainly of various kinds of search*," Turing saw *machine intelligence* as the use of computing machines to solve problems by the *generate-and-test* method: first, generate candidate solutions, then test them (Turing, 1948). In line with these intuitions, the symbolic approach to AI builds on the *physical symbol system hypothesis*:

> "*A physical symbol system has the necessary and sufficient means for general intelligent action.*" (Newell & Simon, 1976)

According to this hypothesis, human intelligence is based on a symbolic system (since it is *necessary*) and implementing it in machines would make them intelligent (since it is *sufficient*). Researchers thus proceeded to hand-code models of the world with rule-based symbolic systems. This approach led to numerous early successes in checker

programs (Copeland & Proudfoot, 2007), theorem proving (Newell & Simon, 1956), puzzles resolution (Newell et al., 1959), simulated communication (Weizenbaum, 1966), and instruction-following (Winograd, 1972), among others.

Connectionist approaches to AI, on the other hand, aim to learn as much as possible from data while staying away from expert knowledge. Instead of searching for solutions in a symbolic solution space, connectionist approaches adapt the parameters of a sub-symbolic model from experience to improve on a set of target tasks. They aim to model perception and cognition by borrowing mechanisms thought to occur in natural agents' brains. Inspired by the works of psychologists such as Thorndike and Hebbs (Thorndike, 1933; Hebb, 1949), connectionist pioneers such as McCulloch, Pitts, or Rosenblatt built the first artificial neural networks: webs of interconnected computing units called *neurons* (McCulloch & Pitts, 1943; Rosenblatt, 1957). Connectionism, however, stayed in the shadow of symbolic AI until Rumelhart and colleagues extended the now-famous *backpropagation* algorithm (Rosenblatt, 1961) to train multi-layer neural networks on continuous data (Rumelhart et al., 1986). This seminal paper presented a connectionist method to solve what everyone thought to be a symbolic problem: learning the past tenses of English verbs. Connectionism, however, had to wait for the recent hardware developments to unveil its potential. Larger computational resources and the invention of convolutional neural networks (LeCun et al., 1995, 1998) laid the ground for the new connectionist wave called *deep learning*. In the last two decades, deep learning was scaled to effectively train highly expressive functions composed of up to billions of parameters and became the method of choice in most machine learning applications (Salakhutdinov, 2014; Schmidhuber, 2015).

Connectionist and symbolic approaches, like System 1 and System 2, are complementary. Let us start by symbolic AI. Like System 2, symbolic AI implements explicit and abstract reasoning but lacks the intuitive, non-symbolic properties of System 1. Learning in spaces of explicit symbols and rules offers interpretability, systematic generalization, and systematicity but fails to represent the implicit knowledge required for perception, continuous control or common sense. Identifying and implementing these rules is time-consuming, requires extensive expert knowledge, and is mostly task-specific. Once rules are hard-coded, the machine cannot unlearn them, thus cannot adapt in the light of new information nor easily transfer to new tasks. Finally, large models lead to a *combinatorial explosion* of the solution space and require the definition of search heuristics.

Connectionist approaches demonstrate complementary properties. They lack interpretability, systematic generalization, systematicity and abstract reasoning because they do not rely on symbolic, rule-based models. However, they can represent implicit knowledge within the networks' activations and, for this reason, can easily handle low-level sensorimotor tasks involving continuous perception and control. Although deep learning methods often require large quantities of data, they do not require extensive domain knowledge and can quickly be adapted to new problems. Recent research attempts to bridge the gap between symbolic and connectionist AI by using a combination of continuous symbolic representations and soft operations to retain the differentiability of the models. These *neuro-symbolic* approaches have proven successful in a variety of domains such as theorem proving (Minervini et al., 2018), visual question answering (Andreas et al., 2016; Mao et al., 2019), program synthesis (Parisotto et al., 2017) or control (León et al., 2020).

Connectionist approaches are better suited to our objective. Indeed, symbolic AI requires experts to define a set of symbols and symbolic rules for each new task. In situated, embodied settings, they need to hand-code modules to translate raw perceptions to symbols and symbols to actions. These hand-coded modules drastically limit the autonomy and the open-endedness of the agents. On the other hand, recent deep learning methods better handle the low-level sensorimotor aspects of situated, embodied settings and require less expert knowledge. They can handle high-dimensional continuous inputs and, when trained on diverse data, demonstrate reasonable generalization capacities. However, although deep learning can be compatible with our requirements, it does not guarantee any of them.

Embodied, Situated Machines

Building on the early developments of *cybernetics* from the 1940s[1] (Wiener, 1948), *nouvelle AI* emerged as a new approach to design embodied, situated agents that react and adapt quickly in unknown environments (Brooks, 1991b; Rosch et al., 1991; Steels, 1993; Meyer, 1996; Chiel & Beer, 1997; Arkin et al., 1998; Pfeifer & Bongard, 2006). Cybernetics and nouvelle AI take the bottom-up approach and build adaptive systems from simple interconnected modules without resorting to internal world models, symbolic representations or planning.

Nouvelle AI emphasizes two concepts: *embodiment* and *situatedness* (Brooks, 1991b; Rosch et al., 1991; Steels, 1993). Natural agents are *embodied*; they perceive the world from their own sensory apparatus and act on it—thus on their perception of it—through their own motor system. They are also *situated* and must react and adapt to their environment in real-time by interacting with it directly—in the words of Brooks, "*the world is its own best model*" (Brooks, 1990). Following these principles, Brooks and colleagues built a series of robots interacting with the physical world in real-time via simple mechanisms, non-symbolic distributed representations and control similar to those found in animals (Brooks, 1990, 1991a,b). These approaches were however criticized for their lack of cognition. Indeed, Brooks' machines are primarily reactive and can only adapt to situations immediately accessible to their sensors. Because they cannot build cognitive representations of what is not there—no world models—and have limited or nonexistent memory, they cannot solve long-term tasks (Clark & Grush, 1999). It seems that nouvelle AI might be lacking the cognitive reasoning of System 1, just like symbolic AI lacked the intuitive reasoning of System 2.

Reinforcement learning (RL) is another set of methods that emphasizes embodiment and situatedness. Inspired by research on animal learning (e.g. Thorndike, 1898), RL trains embodied agents to perform sequences of actions in an environment to maximize some measure of rewards (Sutton & Barto, 1998). When engineers want to train an artificial agent on a particular task, they design a corresponding reward function that generates positive (respectively negative) rewards when the agent demonstrates behaviors

[1] Cybernetics takes its roots in the first study of embodied machines: the life-like behavior of the light-seeking *electric dog* of Hammond and Miessner (Miessner, 1916; Cordeschi, 2002) further rediscovered and extended with the turtles of Bristol (Walter, 1950) and the autonomous vehicles of Braitenberg (Braitenberg, 1986).

compatible (respectively incompatible) with the task. Learning agents interact with their environment and, by trial and error, leverage their experience to improve on the task. Although standard RL problems often involve a single agent learning to solve a unique task, RL researchers recently extended RL problems to *multi-goal RL problems*. We can now train agents to tackle entire goal distributions (Kaelbling, 1993; Sutton et al., 2011; Schaul et al., 2015). As the field progresses, new goal representations emerge: from the specific goal states (e.g. Schaul et al., 2015) to the high-dimensional visual goals (e.g. Nair et al., 2018b) or the abstract linguistic goals (Luketina et al., 2019).

Building on deep learning methods, *deep reinforcement learning* (DRL) has recently emerged as a set of RL algorithms able to train embodied agents to solve ever-harder tasks. Like any deep learning algorithm, DRL benefits from large computational clusters. With a combination of algorithmic advances and extensive computing resources, DRL now achieves impressive results. One can mention, for example, beating the best humans at video games (Mnih et al., 2015; Bellemare et al., 2016; Vinyals et al., 2019; Berner et al., 2019; Badia et al., 2020a), chess and go (Silver et al., 2017), controlling complex robotics systems (OpenAI et al., 2019), or piloting stratospheric balloons in the real world (Bellemare et al., 2020). These successes seem to designate DRL as a good candidate to train artificial embodied agents to learn skills.

Existing DRL approaches, however, fall short of satisfying our requirements. Indeed, standard approaches train agents to solve a restricted set of pre-defined tasks using hand-defined learning signals. Externally-rewarded agents are not autonomous. The pre-definition of the task set also hinders the ability to acquire ever-growing repertoires of skills. Although DRL approaches are more flexible than symbolic AI approaches, they require many interactions to learn and still have trouble generalizing to new goals and transferring knowledge across tasks.

Intrinsically Motivated Machines

In 1950, Turing already reflected on the importance of *development* in the design of intelligent machines:

> "*Instead of trying to produce a programme 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)

Children, however, do not become adults by a simple conditioning process. Although caretakers sometimes use treats and punishments, children mostly learn in autonomy. Indeed, if we look at children—or humans in general—we find that most of the time, they are not motivated by external rewards but spontaneously explore their environment to discover and learn about what is around them. Children play, explore, ask questions about the world, make up pretend scenarios, etc. This behavior seems to be driven by *intrinsic motivations* (IMs), a set of brain processes that motivate humans to explore for the mere purpose of experiencing novelty, surprise or learning progress (Berlyne, 1966; Gopnik et al., 1999; Kidd & Hayden, 2015; Oudeyer & Smith, 2016; Gottlieb & Oudeyer, 2018).

Developmental robotics takes inspiration from artificial intelligence, developmental psychology and neuroscience to model cognitive processes in natural and artificial systems (Asada et al., 2009; Cangelosi & Schlesinger, 2015). While the field of RL is organized around an algorithmic framework, developmental robotics gathers around specific scientific questions dealing with sensorimotor, cognitive and social development (e.g. how can we model language acquisition?). Because modeling child learning requires modeling their exploratory and play behaviors, researchers quickly saw the necessity to integrate IMs in the design of artificial agents (Schmidhuber, 1991b; Kaplan & Oudeyer, 2007). In computational models of intrinsic motivations, agents rely on an internal reward function to generate their own intrinsic rewards. Oudeyer and Kaplan organize these different models of IMs in two prominent families (Oudeyer & Kaplan, 2009):

- **Knowledge-based IMs** (KB-IMs) compare the situations experienced by the agent to its current knowledge and expectations and reward it for experiencing dissonance or resonance. This family includes IMs rewarding prediction errors, novelty, surprise, negative surprise, progress in forward predictions or information gains.
- **Competence-based IMs** (CB-IMs), on the other hand, reward agents to solve self-generated problems; to achieve self-generated goals. CB-IMs agents learn to represent, select and master their own goals. They can bias goal selection towards intrinsic proxies such as novelty or progress in goal-reaching. See a review of intrinsic motivations in Section 1.3.

Intrinsic motivations can organize the behavior and learning of artificial agents, thus provide them with autonomy. In the pursuit of the autonomous acquisition of repertoires of skills, agents must represent goals. Only agents that represent, generate and pursue their own goals can take control of their learning trajectories and decide which goal to target—i.e. which skill to practice—at any time. As we just saw, competence-based IMs can power such systems.

Autotelic Machines

Autonomous agents act independently and do not rely on engineers to hand-code task-specific reward functions in their design. *Intrinsically motivated* agents are *active autonomous agents*; they learn to generate their own learning signals to orient learning and behavior. *Autotelic* agents—from the Greek *auto* (self) and *telos* (end, goal)—generate their own goals and learning signals. Thus, one can see *autotelic* as a special case of *intrinsically motivated*, itself a special case of *autonomous*. As such, *autotelic* stands for *goal-directed and intrinsically motivated*. Note that these characteristics do not prevent agents from leveraging the help of social partners, as long as it occurs through natural interactions, not through the explicit hand-coding of representations or reward functions.

Autotelic agents leverage competence-based intrinsic motivations to organize exploration around self-generated goals. Emerging from the field of developmental robotics, *intrinsically motivated goal exploration processes* (IMGEP) is the family of autotelic algorithms that bake competence-based IMs into learning agents (Rolf et al., 2010; Baranes & Oudeyer, 2010, 2013; Forestier et al., 2017). IMGEP agents generate and pursue their own

goals to explore their environment, discover possible interactions and build repertoires of skills. Existing implementations rely on *population-based* optimization algorithms (POP-IMGEP), non-parametric models trained on datasets of (policy, outcome) pairs see a review in Section 1.4). As we will argue later, this limits the practicality of existing implementations. Indeed, population-based algorithms do not benefit from the higher sample efficiency of DRL methods, have difficulties handling high-dimensional inputs (e.g. images, language) and often require hand-defined representations. Moreover, existing implementations rely on pre-defined and bounded goal spaces, which prevents the emergence of open-endedness.

Linguistic Machines

Humans, and children in particular, learn a lot from social interactions, descriptions, instructions, most of which take the form of spoken or written language. Most of the time, language learning is seen as a disembodied task. This is the case of chatbots, visual question answering systems (e.g. Antol et al., 2015; Andreas et al., 2016) or language models trained from text corpora (e.g. Devlin et al., 2019; Brown et al., 2020). Some researchers, however, have argued that human-level linguistic abilities and common sense can only be acquired via multi-modal interactions—language needs to be grounded in the physical world (Cangelosi et al., 2010; McClelland et al., 2019; Bisk et al., 2020; Lake & Murphy, 2020). Symbolic and connectionist AI can both train such embodied linguistic machines. Symbolic approaches require modules to parse instructions into actions or ground them in symbolic world models (e.g. Winograd, 1972; MacMahon et al., 2006; Kollar et al., 2010), whereas recent DRL approaches can teach artificial agents to follow instructions end-to-end (Luketina et al., 2019). Existing embodied linguistic machines, however, always receive external instructions and rewards.

We believe our autotelic agents must also be linguistic. To this end, they must learn to represent and ground language in experience autonomously, from natural social interactions. Going further, autotelic agents could leverage the *cognitive functions* of language. The view of *language as a cognitive tool* was pioneered by the developmental psychologist Vygotsky (Vygotsky, 1934) and further developed by many others (e.g. Berk, 1994; Clark, 1998a; Clark et al., 1998; Dautenhahn & Billard, 1999; Lindblom & Ziemke, 2003; Mirolli & Parisi, 2011; Gentner & Hoyos, 2017). Language, more than a communication tool, is a cognitive tool that co-develops with the higher cognitive functions of humans: abstract representations, generalization, planning and creativity. It thus seems natural to model these supra-communicative functions of language into linguistic autotelic machines.

Objectives and Contributions

The previous paragraphs described different approaches to AI and discussed how our artificial agents could benefit from them to solve the autonomous acquisition of repertoires of skills. Agents must be situated, embodied and learn efficiently (like DRL agents), be autotelic (like IMGEPs) and learn from social interactions. However, neither RL nor POP-IMGEP solves the problem alone. Reinforcement learning algorithms lack

autonomy and open-endedness because they often require engineers to pre-define the set of tasks and their associated learning signals. POP-IMGEP implementations, on the other hand, generate their own goals but lack generalization abilities and are not robust to perturbations. Existing linguistic machines bring the communicative functions of language to machines, but so far lack autonomy and could be further developed to leverage its supra-communicative functions.

The general objective of the present research is to make progress towards the autonomous acquisition of repertoires of skills. To this end, we formalize a new set of methods that transpose, adapt and extend concepts from the IMGEP family into the technical framework of goal-conditioned deep reinforcement learning. We call this new family RL-IMGEP for RL-based IMGEP.

Our first contribution is to formalize the RL-IMGEP framework as the intersection of developmental robotics and reinforcement learning. We propose a general definition of the terms *goals* and *skills* for RL and review the existing literature under this lens. We observe that many of the components introduced for goal-conditioned RL can be directly transposed to build autotelic agents. We develop this contribution in Chapter 1.

This first conceptual contribution leads to two algorithmic ones. The second contribution, presented in Chapter 2, investigates the use of autotelic exploration to facilitate the resolution of external, pre-defined tasks characterized by sparse and/or deceptive rewards. Our autotelic agents generate their own goals and pursue them as a way to explore an outcome space—a space of abstract representations. This exploration is decoupled from the exploitation phase where agents leverage the experience acquired during exploration to maximize an external reward function, i.e. to solve an external task. In our two first studies, we demonstrate that pursuing one's own goals is an efficient way to self-organize exploration.

Our third contribution moves away from external tasks and defines autotelic RL agents targeting diverse sets of self-generated goals (Chapter 3). We introduce mechanisms to support two essential functions of autotelic agents. First, we propose a general policy architecture enabling agents to target a diversity of goals involving different affordances while maintaining efficient transfer across skills. Second, we let agents organize their own learning trajectories by focusing on goals where they progress the most, avoiding easy and impossible ones.

Our fourth contribution is conceptual and advocates for the use of language as a cognitive tool to structure learning trajectories and skill acquisition. In Chapter 4, we discuss various uses of language as a communicative and cognitive tool in humans and machines and identify potential gains from the use of language in RL-IMGEP approaches. Inspired by the pioneering work of the developmental psychologist Vygotsky (Vygotsky, 1934), we advocate for the design of *Vygotskian autotelic agents*: a developmental approach to RL that follows our RL-IMGEP framework and integrates the use of language as a cognitive tool.

This conceptual contribution leads to two additional algorithmic ones. Our fifth contribution overcomes two limitations of previous approaches: hard-coded goal representations and bounded repertoires of skills. IMAGINE learns goal representations from experience and generates out-of-distribution goals as a way to drive exploration and skill acquisition (Chapter 5). It does so by using language both as a *communicative* and

cognitive tool. It first learns goal representations by interacting with a social partner that provides descriptions of interesting interactions (communication tool). After having grounded language, the agent can use it as a *cognitive tool* to imagine new goals by composing known ones. This creative use of language powers a creative exploration, enhances systematic generalization and offers a path towards open-endedness in artificial agents.

We develop our last contribution in Chapter 6. We propose to use descriptive language as a cognitive tool to prompt the simulation of concrete, possible futures resulting from behaving according to the description. When the agent considers the description "*put the red block on top of the green one*," it can generate a set of concrete goals matching the description. Here, goals are expressed as a set of binary spatial relations that objects in the scene should verify (e.g. blue block close to the green one; red block above the green one). We use semantic representations as a pivot between sensorimotor learning—learning to achieve goal configurations—and language grounding—learning to map language to configurations. This decoupling better models the goal-directed behavior of pre-verbal infants and fosters behavioral diversity such that agents can perform several behaviors for any given instruction.

As a whole, this research uses state-of-the-art techniques from deep reinforcement learning to tackle questions from developmental robotics—*how do children grow repertoires of skills autonomously?* Across several experiments, we move on from task-oriented RL to fully autonomous RL by introducing goals and language as cognitive tools to self-organize exploration and sculpt open-ended repertoires of skills.

Document Structure

We organize this manuscript into three parts. In *Autotelic Reinforcement Learning Agents*, we cover inspirations, concepts, and related works before introducing our computational framework: goal-conditioned intrinsically motivated goal exploration processes (contribution 1). We train autotelic agents to represent, generate, pursue and master their own goals with deep reinforcement learning methods. Following the presentation of these foundational concepts, we cover contributions 2 and 3 in two empirical studies. In *Vygotskian Autotelic Reinforcement Learning Agents*, we propose to endow our agents with linguistic abilities. After a first chapter introducing the Vygotskian idea of *language as a cognitive tool* in humans and machines (contribution 4), we present two empirical studies evaluating linguistic RL-IMGEPs (contributions 5 and 6). Finally, *Discussions* takes a step back. We integrate this research into the ecosystem of related AI approaches, highlight its limits and discuss perspectives for the future.

Collaborations

This research is also the result of several collaborations and discussions with other researchers. During an internship at UberAI Labs, I have worked with Vashisht Madhavan (Element Inc.), Joost Huizinga (OpenAI), Jeff Clune (OpenAI) and Kenneth Stanley (OpenAI) on the second study presented in Chapter 2. There, I learned about evolution strategies and quality-diversity algorithms, among many other subjects. I developed my

understanding of automatic curriculum approaches in collaboration with Rémy Portelas (INRIA), Lilian Weng (OpenAI) and Katja Hofmann (Microsoft) while working on a survey of these methods. During the study presented in Chapter 5, I have worked with Tristan Karch (INRIA), Nicolas Lair (INSERM, Cloud Temple), Clément Moulin-Frier (INRIA), Jean-Michel Dussoux (Cloud Temple) and Peter Ford Dominey (INSERM). Their expertise in linguistic agents helped me refine the idea of seeing language as a cognitive tool to structure learning and exploration in artificial agents. I have developed intrinsically motivated goal-conditioned agents that use learning progress to structure their learning trajectories with Pierre Fournier, Ahmed Akakzia and Mohamed Chetouani (ISIR) while working on the projects presented in Chapters 3 and 6. I had the chance of co-supervising Timothée Anne and Tianwei Lan, two master students working on model-based curiosity algorithms and captioning systems for RL, respectively. Although not presented in this research, I have also applied reinforcement learning methods—including goal-conditioned RL—to the automatic design of intervention strategies in the context of epidemics. I have learned a lot about epidemiology from my co-authors: Boris Hejblum, Mélanie Prague and Rodolphe Thiebaut (INRIA, INSERM). In all of these projects, I have been excellently advised by Olivier Sigaud (ISIR) and Pierre-Yves Oudeyer (INRIA). They could always find the time to discuss ideas, good or bad, and helped me improve both the technical and communication aspects of these projects. Finally, most of this research has been supported by the French Ministère des Armées–Direction Générale de l'Armement.

Publications

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

- GEP-PG: Decoupling Exploration and Exploitation in Deep Reinforcement Learning Algorithms (ICML 2018), Chapter 2 (Colas et al., 2018a).
- CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning (ICML 2019), Chapter 3 (Colas et al., 2019a).
- How Many Random Seeds? Statistical Power Analysis in Deep Reinforcement Learning Experiments (pre-print), Appendix A (Colas et al., 2018b).
- A Hitchhiker's Guide to Statistical Comparisons of Reinforcement Learning Algorithms (pre-print), Appendix A (Colas et al., 2019b).
- Language as a Cognitive Tool to Imagine Goals in Curiosity-Driven Exploration (NeurIPS 2020), Chapter 5 (Colas et al., 2020b).
- Automatic Curriculum Learning For Deep RL: A Short Survey (IJCAI 2020), Chapter 1 (Portelas et al., 2020b).
- Deep Sets for Generalization in RL (BeTRL Workshop, NeurIPS 2020), Chapter 5 (Karch et al., 2020).
- Scaling Map-Elites to Deep Neuroevolution (GECCO 2020), Chapter 2 (Colas et al., 2020d).
- Intrinsically Motivated Goal-Conditioned Reinforcement Learning: a Short Survey (under review), Chapter 1 (Colas et al., 2020c).

- Grounding Language to Autonomously-Acquired Skills via Goal Generation (ICLR 2021), Chapter 6 (Akakzia et al., 2021).
- Language as a Cognitive Tool: Dall-E, Humans and Vygotskian RL Agents (blog post),[2] Chapter 4 (Colas et al., 2021).
- EpidemiOptim: A Toolbox for the Optimization of Control Policies in Epidemiological Models (JAIR 2020), not discussed here (Colas et al., 2020a).

[2] https://developmentalsystems.org/language_as_cognitive_tool_vygotskian_rl

Part I

Autotelic Reinforcement Learning Agents

The first part of this manuscript introduces a computational framework called *rl-based intrinsically motivated goal exploration processes* (RL-IMGEP) to tackle the *autonomous acquisition of open-ended repertoires of skills in unknown environments*. *Autotelicity* is the central concept developed in this part. Formed from the Greek *autos* (self) and *telos* (end, goal), it characterizes intrinsically motivated agents that represent, generate, pursue and master their own goals (Csikszentmihalyi, 1997; Steels, 2004). We develop these ideas in three chapters:

- Chapter 1 presents the foundations of the RL-IMGEP framework. It covers the inspirations and fundamental concepts, introduces the computational frameworks on which ARL builds and surveys related work. We first discuss autotelic behaviors in humans and their source in *intrinsic motivations*: processes that drive humans to be curious and playful, experience interesting situations and explore the world (Berlyne, 1966; Ryan & Deci, 2000a; Oudeyer & Kaplan, 2009). We pursue by covering the basics of two algorithmic frameworks implementing such autonomous and exploratory behaviors in artificial agents: *intrinsically motivated goal exploration processes* (IMGEP) and *reinforcement learning* (RL). After discussing their strengths and limitations, we propose a novel framework at their intersection: RL-IMGEPs. RL-IMGEP agents are *autotelic*; they represent, generate, pursue and target their own goals in an intrinsically motivated goal-directed exploration. They are also goal-conditioned RL agents; they target multiple goals and leverage state-of-the-art RL optimization methods. Finally, we review existing RL algorithms targeting similar problems and discuss their relations to the RL-IMGEP framework.

- Chapter 2 draws the first connections between the IMGEP and RL frameworks. In two empirical studies, we combine an autotelic exploration—agents target their own goals—and a standard RL setting—agents try to maximize an externally defined reward function. In both cases, we use autotelic exploration as an auxiliary task to facilitate the resolution of external tasks that require strong exploration capabilities. As agents grow their repertoires of skills autonomously, they organize their exploration of the environment and transfer knowledge between skills. The collected experience and skills can then be used to solve external tasks.

- Chapter 3 directly targets the problem of the autonomous acquisition of repertoires of skills. Here, agents generate their own goals and learning signals. We mainly focus on two properties of efficient autotelic RL agents. First, we propose a new architecture to enable agents to target a larger diversity of goals involving different types of affordances. The agent shares representations and experiences across different types of goals, which results in a positive transfer that speeds up learning. Second, we propose to use learning progress as an intrinsic motivation to shape the learning trajectories of agents evolving in rich environments. At any point during learning, agents focus on goals where they progress the most. This simple mechanism leads to the emergence of developmental trajectories: agents switch their focus on different parts of the goal space as they learn new skills.

Chapter 1

Foundations: Computational Models of Goals, Intrinsic Motivations and Autotelic Exploration

Artificial autotelic agents aim to autonomously acquire repertoires of skills from experience. In this chapter, we decompose this problem into simpler ones and, in doing so, introduce the building blocks of our proposed autotelic reinforcement learning framework. First, we introduce the reinforcement learning framework (RL) and explain how artificial agents can learn a *single* (Section 1.1) and *multiple* (Section 1.2) pre-defined skills from experience. Such agents lack *autonomy* because skills are pre-defined and learned from signals generated by hard-coded reward functions. Autonomous learning agents need to craft their own skills by generating their own learning signals. To this end, we introduce the concept of *intrinsic motivations* and present a typology of computational models (Section 1.3). Among these models, *competence-based intrinsic motivations* specifically endow agents with the drive the represent, generate, pursue and master their own goals; to shape their own learning trajectories. Within the field of developmental robotics, *intrinsically motivated goal exploration processes* (IMGEP) emerged as a computational framework integrating competence-based intrinsic motivations to define autotelic agents (Section 1.4). After discussing the limitations of current implementations based on population-based optimization methods (POP-IMGEP), we go on to introduce our novel computational framework: RL-IMGEP (Section 1.5). The RL-IMGEP framework builds on the concepts introduced in the previous sections and lies at the intersection of IMGEP and RL. It transposes, adapts and extends insights from IMGEPs to train goal-conditioned policies to craft their own repertoires of skills via state-of-the-art RL methods. In the last section, we discuss how the frameworks above handle challenging *exploration* settings (Section 1.6).

1.1 Learning One Pre-Defined Skill with Reinforcement Learning

How can an artificial agent learn a pre-defined skill? As argued in the introduction, artificial agents need to be embodied and situated, i.e. they need to learn from interactions with their environment. Inspired by the operant and drive reduction theories from

psychology, *reinforcement learning* offers a computational framework to train agents to acquire skills. This section formalizes the standard RL problem, presents the basics of RL optimization methods and briefly discusses other methods to solve RL problems.

Inspirations from Psychology

At the end of the ninetieth century, the psychologist Thorndike plotted the first learning curve: cats find the exit of a maze quicker with the number of attempts (Thorndike, 1898). From these experiments, he hypothesized the *law of effect*—also called *operant conditioning*—behaviors followed by positive consequences tend to repeat, the ones followed by negative consequences tend not to. In the behaviorist *drive reduction theory of motivations*, natural agents experience *drives* resulting from physical or psychological needs—hunger, thirst, reproduction, etc. (Hull, 1943). The appearance of a need leads to discomfort (costs) that drive the agent to retrieve the stable and rewarding state of *homeostasis*. Building on these ideas, Skinner further developed the *operant theory* (Skinner, 1953). In analogy with Darwinian evolution, he posited that new behaviors appear as variations of existing ones and can then be either reinforced or extinguished depending on the rewards or punishments they lead to. In line with behaviorist accounts of psychology, all these theories picture learning agents as passive entities modeled by rewards and punishments.

Reinforcement learning (RL) offers a mathematical and computational framework to implement these ideas. Following the operant theory, it trains embodied agents to perform sequences of actions to maximize future rewards. RL agents learn by trial-and-error: they interact with their environment, observe and learn from the consequences of their actions in terms of costs and rewards. We distinguish *RL problems*—a kind of optimization problem—from *RL algorithms*—a set of solutions to tackle RL problems.

Reinforcement Learning Problems

In an RL problem, the agent learns to perform sequences of actions in an environment to maximize some measure of cumulative reward (Sutton & Barto, 1998). Let us consider a learning agent evolving in an environment for a given amount of time steps $t = 1, 2, \ldots, T$. The interaction can be *episodic* with fixed or variable length T or can consist in a single (potentially infinite) lifetime where T represents the agent's death. The initial *state* of the world s_0—agent and environment—is sampled from an *initial state distribution* $s_0 \sim \rho_0(\mathcal{S})$ where s_t is the state of the world at time step t, $\mathcal{S}$ is the state space and ρ_0 the initial state distribution. The agent then repetitively performs actions in the environment $a_t \in \mathcal{A}$ and observes consequences: a novel state of the world s_{t+1} and a reward $r_{t+1} = R(a_t, s_{t+1})$—see Figure 1.1.

We often frame RL problems as *Markov decision processes* (MDPs): $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0, R\}$ (Sutton & Barto, 1998), where $\mathcal{T}$ is the *transition function* dictating the distribution of the following state s' from the current state and action $\mathcal{T}^a_{ss'} = p(s' \mid s, a)$. The MDP framework makes the *Markov assumption*: given the present, the future does not depend on the past. This means that knowing s_t, previous states s_i, $\forall i \in [\![0, t-1]\!]$ does not

bring new information to help predict the following state s_{t+1}:

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

The objective of the agent is to maximize a cumulative measure of reward called *return*. This return is often modulated by the discounting function Γ so that the *discounted return* G_t from time step t onward is:

$$G_t = \sum_{i=t}^{T} \Gamma(s_i, a_i, i) \times R(a_i, s_i).$$

Γ is usually an exponentially-decreasing function of time: $\Gamma(t) = \gamma^t$ with a constant discount factor $\gamma \in [0, 1]$. Each instance of an MDP is called a *task*. An RL problem is a collection of tasks, also called a *task set*. In the episodic setting, the agent faces a new task sampled from the task set in each new episode.

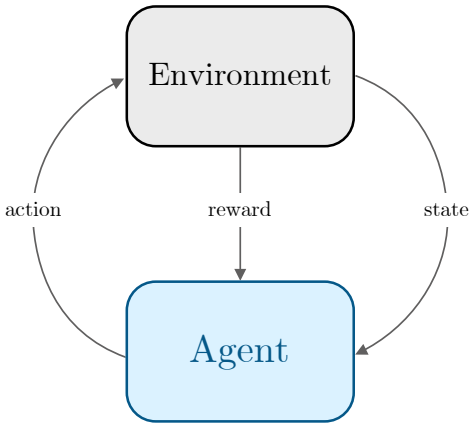


Figure 1.1: Agent-environment interaction loop in reinforcement learning.

Reinforcement Learning Algorithms

RL methods use transitions $(s_t, a_t, s_{t+1}, r_{t+1})$ collected via interactions between the agent and its environment to train a *policy* π: a function that generates the next action a_{t+1} based on the current state s_t to maximize the expected return. Most algorithms rely on the definition of *value* and/or *action-value functions*. The value function $V_\pi(s)$ of a policy π is the expected return from state s when following policy π. The action-value function $Q_\pi(s, a)$ of a policy π is the expected return from state s when the agent executes action a and follows π after that. Value (and action-value) functions thus evaluate the quality of being in a given state (and performing a given action) conditioned on a policy. The *optimal* value function $V^*(s)$ and optimal action-value function $Q^*(s, a)$ are the value and action-value functions of an optimal policy, a policy that maximizes the expected return.

We can decompose values into two components: the reward we expect for the next transition $r(s', a)$ and the discounted expected value of the next state. This decomposition

unveils a recursive definition: values are functions of other values. Precisely, value and action-value functions obey the *Bellman equations* (Sutton & Barto, 1998):

$$V_\pi(s) = \int_a \int_{s'} (r(s', a) + \gamma V_\pi(s'))\, \mathcal{T}^a_{ss'}\, ds'\, \pi(a \mid s)\, da, \tag{1.1}$$

$$Q_\pi(s, a) = \int_{s'} \left(r(s', a) + \gamma \int_{a'} Q_\pi(s', a')\, \pi(a' \mid s')\, da' \right) \mathcal{T}^a_{ss'}\, ds', \tag{1.2}$$

$$V^*(s) = \max_a \int_{s'} (r(s', a) + \gamma V^*(s'))\, \mathcal{T}^a_{ss'}\, ds', \tag{1.3}$$

$$Q^*(s, a) = \int_{s'} \left(r(s', a) + \gamma \max_{a'} Q^*(s', a') \right) \mathcal{T}^a_{ss'}\, ds'. \tag{1.4}$$

The two first equations (Equations 1.1 and 1.2) compute expected values and action-values over the distribution of actions generated by the policy π. In an RL problem, the objective is to find the policy corresponding to the *optimal value function* V^*, the one that yields maximum value:

$$V^*(s) = \max_\pi V_\pi(s),$$
$$Q^*(s, a) = \max_\pi Q_\pi(s, a).$$

If we were to know the optimal action-value function, then the optimal policy π^* is simply the one that always takes actions maximizing $Q^*(s, a)$ in each state s: $\pi^*(s) = \text{argmax}_a\, Q^*(s, a)$. When the dynamics $\mathcal{T}$ are known or learned from data, one can obtain Q^* with dynamic programming methods. When they are not, we can approximate it with RL algorithms.

This research builds on model-free, off-policy deep RL methods. Let us unpack these three concepts. *Model-free* algorithms—opposed to *model-based*—learn directly from experience without constructing any explicit dynamics model. *Off-policy* algorithms—opposed to *on-policy*—learn to approximate the action-value Q^* of a greedy optimal policy; not the one of the *behavioral policy* used for data collection. This is important as, in principle, it allows agents to leverage data collected by *any policy* (e.g. old policies, exploration policies, expert demonstrations, etc.) to improve the current behavioral policy. Finally, deep RL (DRL) refers to RL approaches that leverage deep neural networks as function approximators.[1] Now that we defined the characteristics of our algorithms, let us present the algorithm we use most often in this research and its origins in non-deep RL.

Model-free, off-policy algorithms are trained to approximate the optimal action-value function Q^* by minimizing the temporal difference error L_i:

$$L_i = \mathbb{E}_{(s, a) \sim \rho(\cdot)} \left[(y_i - Q(s, a))^2 \right],$$

where $y_i = \mathbb{E}_{s' \sim \text{env}}[r + \gamma \max_{a'} Q(s', a') \mid s, a]$ is the target at optimization iteration i bootstrapped from the current Q-network Q and $\rho(\cdot)$ is the distribution of states and

[1] These networks contain up to several millions of parameters when they handle visual inputs but are still relatively shallow compared to the architectures used in vision or language tasks (up to hundreds of billions of parameters).

actions in the dataset used to train the Q-network. The Q-learning algorithm is the original off-policy RL algorithm (Watkins, 1989). It stores Q-values in a table for each (s, a) pair and approximate expectations by a single sample from the current behavioral policy. Each step in the environment is followed by an update of the Q-table.

DRL extensions of model-free, off-policy learning represent value and action-value functions by deep neural networks $V^*(s) = V(s; \theta^V)$ and $Q^*(s, a) = Q(s, a; \theta^Q)$, where θ^V and θ^Q are sets of network parameters. The loss function can then be differentiated with respect to the weights θ_i while freezing the weights used to compute the target y_i. In the case of deterministic policies, the resulting gradient is:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{(s,\,a)\sim\rho(\cdot);\, s'\sim\mathcal{T}^a_{ss'}} \left[(y_i - Q(s,\, a;\, \theta_i))\, \nabla_{\theta_i} Q(s,\, a;\, \theta_i) \right].$$

Deep Q-Networks (DQN) is the first DRL extension of the original Q-learning algorithm. It integrates two additional ideas: experience replay and target networks. Experience replay stores past transitions into a replay buffer and samples transitions from it uniformly to compute updates (Lin, 1992). This allows to reuse experience (replay) and to minimize correlations between samples (shuffled replay). Targets y_i are computed by target networks. These are updated less frequently than the networks used to generate behavior, to stabilize their updates. These two mechanisms are simple but efficient techniques to stabilize the training of deep neural networks. They kick-started the field of DRL (Mnih et al., 2015).

In this research, we mainly use the *deep deterministic policy gradient* algorithm (DDPG), one of the first DRL algorithms to handle continuous action spaces (Lillicrap et al., 2016). In addition to the Q-network, DDPG trains a policy to generate actions maximizing the Q-network's output. It is done by following the gradient of the loss through the Q-network to the policy using the chain rule (Silver et al., 2014):

$$\begin{aligned} \nabla_{\theta^\pi} L &\approx \mathbb{E}_{s\sim\rho(\cdot)} \left[\nabla_{\theta^\pi} Q(s,\, a;\, \theta^Q) \mid_{s,\, a=\pi(s;\,\theta^\pi)} \right] \\ &= \mathbb{E}_{s\sim\rho(\cdot)} \left[\nabla_a \; Q(s,\, a;\, \theta^Q) \mid_{s,\, a=\pi(s;\,\theta^\pi)} \nabla_{\theta^\pi} \pi(s;\, \theta^\pi) \mid_s \right] \end{aligned}$$

DDPG transposed the techniques that made the success of DQN to the continuous case and achieved efficient non-linear continuous control for the first time (Lillicrap et al., 2016). The field of DRL encompasses many more algorithms: on-policy variants such as TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017), or more sophisticated off-policy algorithms such as TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018b)—see Figure 1.2. Note that our research is mostly orthogonal to the choice of underlying optimization algorithm.

Other Solutions to the RL Problem

Other sets of methods can be used to tackle RL problems. Evolutionary computing (EC) is a group of population-based approaches where populations of policies are trained to maximize episodic measures of rewards called *fitness* (see reviews in Stulp & Sigaud, 2013; Eiben & Smith, 2015; Qian & Yu, 2021). *Genetic algorithms* (GA)—a sub-family of EC—evolve a population of *individuals* (policies) characterized by their *genotype* (parameters) (Holland, 1992; Eiben & Smith, 2015). At each *generation* (optimization step), they mutate a selection of parent individuals to form offspring individuals. They

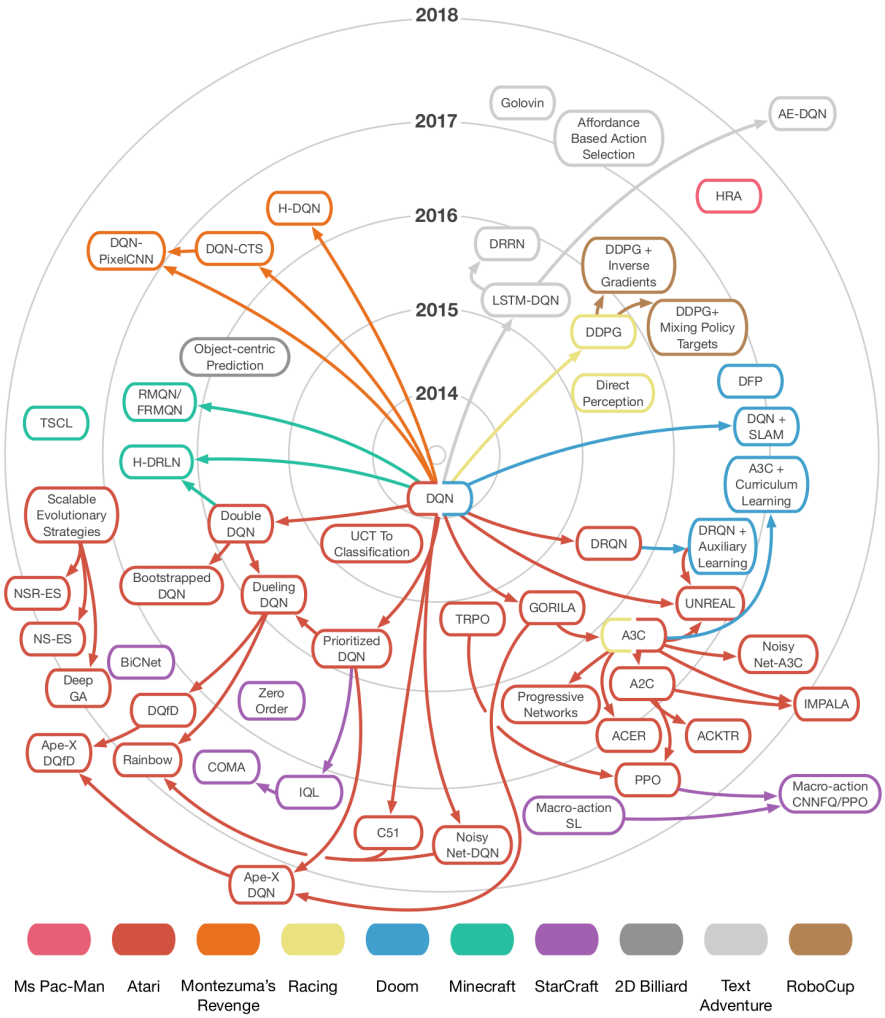


Figure 1.2: A recent history of algorithms tackling the RL problem with deep neural networks, from Justesen et al. (2020a).

evaluate the mutated controllers in the environment and rank them according to their *fitness*. Finally, the new set of parent individuals is selected among the offspring according to their rank and sometimes diversity metrics (e.g. proportional sampling, tournament selection). *Evolution strategies* (ES)—another sub-family of EC—also mutate and evaluate individuals, but do so from a reduced set of parent individuals. Mutated individuals and their fitness measures are used to update the parents (Rechenberg, 1973; Back et al., 1991; Wierstra et al., 2008; Salimans et al., 2017).

Closer to the field of RL, *imitation learning* (Bain & Sammut, 1995; Schaal & others, 1997; Ho & Ermon, 2016; Vecerík et al., 2017; Hester et al., 2018; Nair et al., 2018a; Torabi et al., 2018) and *inverse reinforcement learning* (Abbeel & Ng, 2004; Ziebart et al., 2008; Fu et al., 2018) techniques can be used to learn behaviors from demonstrations. Model-based RL algorithms learn a model of the transition function $\mathcal{T}$ and perform planning or standard RL within that model to improve sample efficiency (e.g. Chua et al., 2018; Schrittwieser et al., 2020; Charlesworth & Montana, 2020), see recent reviews in Moerland et al. (2020); Hamrick et al. (2020). Although this research builds on model-free

RL and EC algorithms, most of our proposed mechanisms can directly be transposed to imitation learning, inverse reinforcement learning or model-based approaches.

Summary

Reinforcement learning, inspired by operant theory, defines a mathematical and computational framework to train agents to solve a external task characterized by a hard-coded reward function. Beside model-free RL algorithms, others optimization algorithms can solve RL problems (evolutionary computations, imitation learning, inverse reinforcement learning and model-based RL). However, these approaches fall short of solving our problem because they only learn one skill at a time.

1.2 Learning Multiple Pre-Defined Skills with Goal-Conditioned Reinforcement Learning

Humans learn multiple skills because they can target multiple goals. Autotelic agents should demonstrate the same behavior. In this section, we turn to *goal-conditioned reinforcement learning* (GC-RL), the set of RL methods that train agents to learn multiple skills. Our general definitions of *goals* and *skills* allow us to organize the existing literature into a typology of goal representations.

Goal-Directed Behaviors in Humans

The *goal* construct is a central concept in most theories of motivation (Elliot & Fryer, 2008; Shah & Gardner, 2008). Most accounts even use the term explicitly, e.g. *"those psychological processes that cause the arousal, direction, and persistence of voluntary actions that are goal directed"* (Mitchell, 1982). Discussing the origin of the *goal* construct and its use in past psychological research, Elliot and Fryer propose a general definition:

> "*A goal is a cognitive representation of a future object that the organism is committed to approach or avoid.*" (Elliot & Fryer, 2008)

Because goals are *cognitive representations*, only animate organisms that represent goals qualify as goal-conditioned. Because this representation relates to a *future object*, goals are cognitive imaginations of future possibilities: goal-conditioned behavior is proactive, not reactive. Finally, organisms *commit* to their goal. Their behavior is thus directly influenced by this cognitive representation.

Goals structure human behaviors into chunks. This structure is deeply hierarchical. The most simple goals involve sensorimotor actions, something like *"move your hand a bit to the right,"* and benefit from direct proprioceptive feedback loops for us to assess progress. At a higher level, we can combine these short-term sensorimotor goals to achieve conscious sensorimotor goals like *"hand her the book."* This can go all the way up to very abstract *life-goals* such as *"be a good friend"* (Chulef et al., 2001). Goals can call upon

physical changes in the world or changes in our state of knowledge (Ram et al., 1995). They can take various forms of cognitive representations from unconscious sensorimotor goals to highly conceptual goals (e.g. *be a good friend*), through visual (e.g. *reproducing a painting*) or auditory goals (e.g. *playing an A chord*).

Skills can be defined as the association of a goal and the behavior to achieve it. Setting goals to oneself is thus a formidable way to practice skills or develop new ones. Setting known goals is a way to organize a consistent behavior—e.g. *brushing my teeth, going to work, taking care of grandma.* Setting unknown goals allows us to consistently explore the world and explore ways to achieve new objectives—e.g. *learning to work with wood, going on a trip to Japan.*

Goals and Skills for RL Agents

As argued above, goal-directed organisms need to condition their behavior on cognitive representations of their goals. Goal-directed behavior is active and aims at turning the present into the future encoded in the goal representation. Reinforcement learning satisfies part of these requirements: it trains learning agents (*organisms*) to maximize (*approach*) a cumulative reward (*future object*). Reward functions indeed characterize the agent's objective, the *future object* the agent tries to approach. To target goals, RL agents need to be explicitly conditioned (to commit) on cognitive goal representations.

Goals define learning problems with their objectives and constraints. When an agent pursues a goal expressed as a target image s_g, these constraints can be expressed by a goal-region S_g, such that the goal is reached when the current state s belongs to $S_g = \{s;\ \|s - s_g\| < \epsilon\}$ where ϵ is a tolerance margin (e.g. Nair et al., 2018b; Pong et al., 2020). Linguistic goals often express more abstract constraints. The goal-region of "*find a red object or a wooden one*," for instance, might contain several non-contiguous regions of the state space where the linguistic predicate is satisfied (e.g. Hermann et al., 2017; Chevalier-Boisvert et al., 2019). Goal-directed agents are conditioned on a compact cognitive representation of the goal (e.g. the image or the description) and update their behavior to make progress towards these goals (e.g. reaching the goal-region faster). This leads to the following formalization of RL goals:

> Each goal is a $g = (z_g, R_g)$ pair where z_g is a compact *goal parameterization* or *goal embedding* and R_g is a *goal-achievement function* measuring progress towards the goal.

In most cases, the goal-achievement function R_g is shared across goals, resulting in a *goal-parameterized* or *goal-conditioned reward function* $R_\mathcal{G}$ such that $R_g(\cdot) = R_\mathcal{G}(\cdot \mid z_g)$.

A *skill* υ_g is the association of a *goal* g and a *policy* π_g to reach that goal. A policy trained via standard RL characterizes a specific behavior oriented towards an implicit objective—the reward function it was trained to maximize. Only the combination of a goal and a goal-conditioned policy can form a skill: a specific behavior oriented towards an explicit objective that the agent can represent and select. Several of these make a repertoire of skills $\Upsilon_\mathcal{G}$; a collection of goal-oriented behaviors the agent can choose from to interact with its environment. Goal embedding, goal-achievement function and

goal-conditioned policy respectively encode a cognitive representation of the skill, how to recognize when it is performed and how to perform it. The semantics of the skill are grounded in the environment through these three representations: the policy uses the space of goal embeddings as a map, the goal embedding as target coordinates and the goal-achievement function as a compass.

Multi-Goal Reinforcement Learning Problems

A significant part of machine learning research is concerned with the *multi-task problem*—training a function to perform multiple tasks using shared representations to leverage cross-task learning (Caruana, 1997). Multi-task RL, in particular, trains RL agents to solve sets of MDPs. In this research, we focus on tasks that differ by their reward functions but keep their dynamics, action and state spaces constant. Tasks that differ by their reward functions are called *goals*, and the problem of training agents to solve multiple goals is known as the *multi-goal RL problem* (Schaul et al., 2015). The multi-goal RL problem can be formalized by extending the standard MDP formulation and replacing the unique reward function R with a space of reward functions $\mathcal{R}_\mathcal{G}$ and a space of goal embeddings $\mathcal{Z}_\mathcal{G}$: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0, \mathcal{R}_\mathcal{G}, \mathcal{Z}_\mathcal{G}\}$. In this problem, the environment presents a goal at each new episode and generates goal-specific rewards—see Figure 1.3.

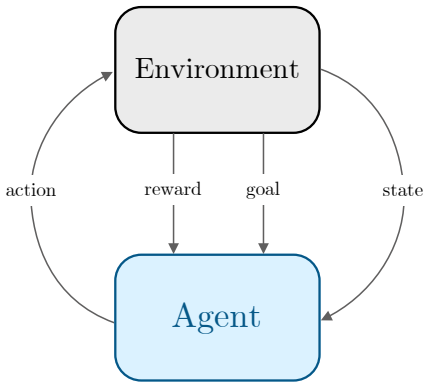


Figure 1.3: Agent-environment interactions in multi-goal RL. Note that the goal is generated once per episode.

Goal-Conditioned Reinforcement Learning Algorithms

Goal-conditioned agents see their behavior affected by the goal they pursue. This is formalized via goal-conditioned policies; policies that produce actions based on the environment state and the agent's current goal: $\Pi : \mathcal{S} \times \mathcal{Z}_\mathcal{G} \to \mathcal{A}$. Here, $\mathcal{Z}_\mathcal{G}$ denotes the space of goal embeddings corresponding to the goal space $\mathcal{G}$ (Schaul et al., 2015). Note that ensembles of policies can also be formalized this way as a high-level policy Π that retrieves the low-level policy from an index z_g (e.g. Kaelbling, 1993; Sutton et al., 2011). The idea of using a unique RL agent to target multiple goals dates back to Kaelbling (1993). In this approach and the following HORDE (Sutton et al., 2011), the experience collected while aiming at a particular goal is used to update knowledge

about all goals. Building on these results, *universal value function approximators* (UVFA) proposes to train a single goal-conditioned policy and value function to learn about all goals simultaneously (Schaul et al., 2015). As they show, uni-goal DRL algorithms can directly be used to train goal-conditioned policies and value functions. Using neural networks as function approximators, they show that UVFAs enable efficient transfer between goals and demonstrate strong generalization to new goals. This is made possible both by the exploitation of the structure encoded in the goal representations (similar goals are encoded similarly) and the transfer of representations across goals resulting from weight sharing within the unique policy and value function.

The idea of *hindsight learning* further improves knowledge transfer between goals (Kaelbling, 1993; Andrychowicz et al., 2017). Learning by hindsight, agents can reinterpret a past trajectory collected while pursuing a given goal in the light of a new goal. By asking themselves, *what are goals for which this trajectory is a good one?*, they can use the initially-failed trajectory as an informative trajectory to learn about another goal, thus making the most out of every trajectory (Eysenbach et al., 2020). This ability dramatically increases the sample efficiency of goal-conditioned algorithms and is arguably an important driver of the recent interest in goal-conditioned RL approaches (e.g. Mankowitz et al., 2018; Plappert et al., 2018a; Chan et al., 2019; Jiang et al., 2019; Lanier et al., 2019).

A Typology of Goal Representations

The generalized definition of goals presented above encompasses a wide diversity of goal representations. This section presents a typology of goal representations used in GC-RL and is a contribution of this work, see Figure 1.4. It is based on our recent review (Colas et al., 2020c) and includes works published in parallel or even after the studies presented in this manuscript. This is important to form a clear picture of this very recent field and understand the positioning of our studies.

Goals as choices between multiple objectives. Goals can be expressed as a list of different objectives the agent can choose from.

Goal embedding. In that case, goal embeddings z_g are one-hot encodings of the current objective being pursued among N available objectives. z_g^i is the i^{th} one-hot vector: $z_g^i = (\mathbb{1}_{j=i})_{j\in[\![1,\,N]\!]}$ (Oh et al., 2017; Mankowitz et al., 2018; Chan et al., 2019; Codevilla et al., 2018).

Reward function. The goal-conditioned reward function $R_\mathcal{G}$ is a collection of N distinct reward functions $R_\mathcal{G}(\cdot) = R_i(\cdot)$ if $z_g = z_g^i$, $i \in [\![1, N]\!]$. Each reward function can, for example, generate a positive reward whenever the agent reaches a specific object: e.g. reaching guitars and keys (Mankowitz et al., 2018) or monsters and torches (Chan et al., 2019).

Goals as target features of states. Goals can be expressed as target features of the state the agent desires to achieve.

Goal embedding. In this scenario, a state representation function φ maps the state space to an embedding space $\mathcal{Z} = \varphi(\mathcal{S})$. Goal embeddings z_g are target points in $\mathcal{Z}$ that the agent should reach. In the simple case, the representation function only selects a few

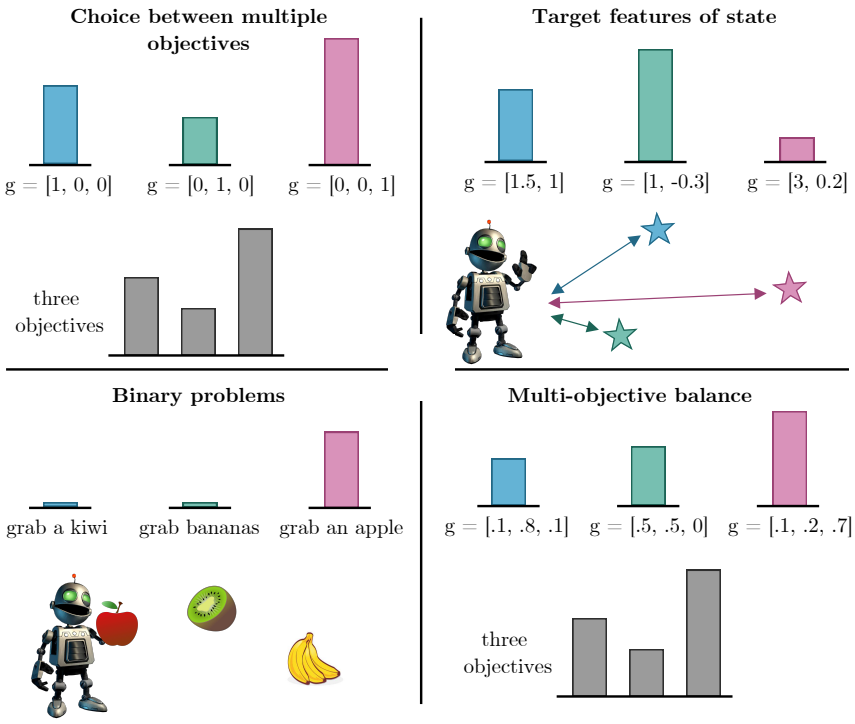


Figure 1.4: Representation of several goal types. The blue, green and pink bar plots depict goal-specific reward measures.

features of the original state. In manipulation tasks, for instance, z_g can be target block coordinates (Andrychowicz et al., 2017; Nair et al., 2018a; Plappert et al., 2018a; Fournier et al., 2019; Blaes et al., 2019; Li et al., 2020; Lanier et al., 2019; Ding et al., 2019). In navigation tasks, z_g can be target agent positions (e.g. in mazes Schaul et al., 2015; Florensa et al., 2018). Sometimes, the representation function needs to be learned. This is the case for visual goals, where the representation function φ is usually implemented by a generative model trained on visual states encountered by the agent during its lifetime. Goal embeddings can be sampled from the generative model or encoded from previously visited states (Zhu et al., 2017; Pong et al., 2020; Nair et al., 2018b; Warde-Farley et al., 2019; Codevilla et al., 2018; Florensa et al., 2019; Venkattaramanujam et al., 2019; Lynch et al., 2020; Lynch & Sermanet, 2020; Nair et al., 2020; Kovač et al., 2020).

Reward function. For this type of goals, the reward function $R_\mathcal{G}$ is based on a distance metric D. One can define a dense reward as inversely proportional to the distance between features of the current state and the target goal embedding: $R_g = R_\mathcal{G}(s \,|\, z_g) = -\alpha \times D(\varphi(s), z_g)$ (e.g. Nair et al., 2018b). The reward can also be sparse: positive whenever that distance falls below a pre-defined threshold: $R_\mathcal{G}(s \,|\, z_g) = 1$ if $D(\varphi(s), z_g) < \epsilon$, 0 otherwise.

Goals as abstract binary problems. Some goals cannot be expressed as target state features but can be represented as *binary problems*: each goal expresses a set of constraints on the state (or trajectory) such that these constraints are either verified or not (binary goal achievement).

Goal embeddings. In binary problems, goal embeddings can be any representation of the set of constraints that the state should respect. The Go-Explore approaches proposes a pre-defined discrete state representation by downsampling images (Ecoffet et al., 2021). These representations lie in a finite embedding space so that goal completion can be asserted when the current embedding $\varphi(s)$ matches the goal embedding z_g. Another way to express sets of constraints is via linguistic predicates. A sentence describes the constraints expressed by the goal and the state or trajectory either verifies them or does not (Hermann et al., 2017; Chan et al., 2019; Jiang et al., 2019; Bahdanau et al., 2019a; Chevalier-Boisvert et al., 2019; Hill et al., 2019; Cideron et al., 2020b; Lynch & Sermanet, 2020). When goals can be expressed by sentences, goal embeddings z_g are usually language embeddings learned jointly with either the policy (e.g. Hermann et al., 2017; Chevalier-Boisvert et al., 2019; Jiang et al., 2019) or the reward function (Fu et al., 2019; Bahdanau et al., 2019a).

Reward function. The reward function of a binary problem can be viewed as a binary classifier that evaluates whether state s (or trajectory τ) verifies the constraints expressed by the goal semantics (positive reward) or not (null reward). This binary classification setting has directly been implemented to learn language-based goal-conditioned reward functions $R_g(s \mid z_g)$ in Bahdanau et al. (2019a) and in our study in Chapter 5.

Goals as a multi-objective balance. Some goals can be expressed not as desired regions of the state or trajectory space but as more general objectives that the agent should maximize. In that case, goals can parameterize a particular mixture of multiple objectives that the agent should maximize.

Goal embeddings. Here, goal embeddings are simply sets of weights that balance the different objectives $z_g = (\beta_i)_{i \in [\![1, N]\!]}$ where β_i is the weight applied to objective i and N is the number of objectives. Note that, when $\beta_j = 1$ and $\beta_i = 0$, $\forall i \neq j$, the agent can decide to pursue any of the objective alone. For example, *Never Give Up* and *Agent57* agents are trained to maximize a mixture of extrinsic and intrinsic rewards (Badia et al., 2020b,a). Agent can select the mixing parameter β that can be viewed as a goal.

Reward function. When goals are represented as a balance between multiple objectives, the associated reward function can be represented neither as a distance metric nor as a binary classifier. Instead, the agent needs to maximize a convex combination of the objectives: $R_g(s) = \sum_{i=1}^{N} \beta_g^i \, R^i(s)$ where R^i is the i^{th} of N objectives and $z_g = \beta_i^g \mid_{i \in [\![1, N]\!]}$ is the set of weights.

Summary

Inspired by the goal-directed behavior of humans, this section defined the concepts of *goals* and *skills* in the context of RL. We presented the extension of the standard RL framework to the multi-goal setting and proposed a structured review of the different types of goals found in the literature. Interestingly, our goal formulation allowed us to integrate approaches that did not consider themselves as goal-conditioned (Badia et al., 2020b,a). Although they target multiple goals, GC-RL approaches are not autotelic. Indeed, they learn pre-defined skills characterized by rewards provided by the environment and are, thus, not autonomous.

1.3 Autonomous Learning with Intrinsic Motivations

Standard reinforcement learning, consistent with operant theory, represents agents as passive entities modeled by rewards and punishments. Although this can explain some of the behaviors of natural agents, it dramatically fails to characterize their *exploratory behaviors*; all the behaviors that are *not* caused by external reinforcement and punishments and do not satiate immediate needs (Berlyne, 1950; White, 1959). Indeed, most animals spend a significant portion of their time exploring their environment, interacting with objects or playing with others. Humans, especially, are curious, playful and self-motivated. They set and pursue their own goals, strive for learning and developing new skills (Ryan & Deci, 2000b; Csikszentmihalyi, 1997). To explain these behaviors, psychologists and cognitive scientists have posited and discovered the existence of *intrinsic motivations* (IM), a set of brain processes motivating animals to explore for the mere purpose of experiencing *interesting* situations (Berlyne, 1950; White, 1959; Berlyne, 1966; Loewenstein, 1994; Gopnik et al., 1999; Kidd & Hayden, 2015; Baldassarre, 2011; Baldassarre & Mirolli, 2013; Oudeyer & Smith, 2016; Gottlieb & Oudeyer, 2018). If intrinsic motivations can drive the autonomous behavior of humans, can they be used to power similar behaviors in artificial agents?

After taking inspiration from the definitions of intrinsic motivations developed in psychology and cognitive science, we review a typology of computational models of IMs and discuss their use to support autonomy in artificial agents. Finally, we take an evolutionary perspective and argue for the adaptiveness of intrinsically motivated exploratory behaviors in natural and artificial agents.

Intrinsic Motivations in Humans

Intrinsic motivations drive agents to experience *interesting* situations. But, how should we understand *"interesting"* here? Many theorists have attempted to answer this question. Although answers differ in their specifics, most mention the following principle: one way for situations to be interesting is to be *novel, but not too much*. Suggested answers include *optimal incongruity* (Hunt, 1965), *intermediate novelty* (Berlyne, 1950, 1966), *intermediate complexity* (Kidd et al., 2012) and *optimal challenge* (Czikszentmihalyi, 1990). In his theory of *flow*, the psychologist Csikszentmihalyi uses the term *autotelic* to describe intrinsically motivated agents or the activities they perform in that psychological state—see Figure 1.5 (Csikszentmihalyi, 1997). Some have even argued that intrinsically motivated experiences were strongly linked to positive attitudes towards failure, subjective well-being, improved performance at work and creativity (Csikszentmihalyi, 1997; Ryan & Deci, 2000b), or that they might be at the origins of some of the greatest humans' feats such as creativity in science, art, music and humor (Schmidhuber, 2010).

Computational Accounts of Intrinsic Motivations

These investigations led to a similar interest in the use of IMs to motivate artificial learning agents, both in developmental robotics (Oudeyer & Kaplan, 2009; Oudeyer et al., 2007; Gottlieb et al., 2013; Baldassarre & Mirolli, 2013; Baldassarre, 2014) and

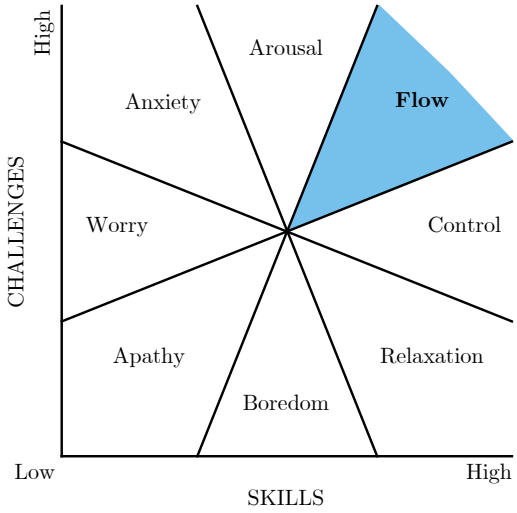


Figure 1.5: Quality of experience as a function of challenge, skills and their relation. The optimal experience, or *flow*, only occurs when challenges and skills are both high (optimal challenge). Adapted from Csikszentmihalyi (1997).

reinforcement learning (RL) (Schmidhuber, 1991a, 2010; Barto & Simsek, 2005; Singh et al., 2010; Barto, 2013). The following paragraphs sketch a high-level typology of intrinsic motivations based on Ryan & Deci (2000a) and Oudeyer & Kaplan (2009).

Intrinsic motivations are often defined in contrast to *extrinsic motivations*. Extrinsic motivations are the drives that push agents to reach separable outcomes—e.g. the child draws a tree for a class assignment (Ryan & Deci, 2000a; Oudeyer & Kaplan, 2009). On the other hand, *intrinsic motivations* drive agents to engage for the sake of experiencing the activity itself, with no notion of outcome—e.g. the child draws a tree because she likes drawing. Another way to put it is to define extrinsic motivations as *task-related* and intrinsic motivations as *task-independent*—enjoying drawing neither depends on what is being drawn nor on the recipient of the drawing (Barto & Simsek, 2005).

Extrinsic and *intrinsic* motivations are to be differentiated from *external* and *internal* motivations. External and internal simply refer to the source of the reward, whether it is generated within the agent (internal) or outside of it (external). All intrinsic rewards can be considered internal, but extrinsic rewards can either be internal—the child spontaneously draws a tree to please his dad—or external—the child draws a tree because his teacher asked him (Oudeyer & Kaplan, 2009). In fact, extrinsic motivations can be considered more or less internal depending on their degree of internalization by the agent (Ryan & Deci, 2000a). Although we could argue that all rewards are internal in definitive—because motivations are only influxes of neuro-transmitters—we take a computational point of view and understand *internal* and *external* as describing the location of the reward generator, either *in* or *out* of the learning agent. Figure 1.6 represents the internalization of the reward function in the design of intrinsically motivated reinforcement learning agents.

Intrinsic motivations are essential to the design of autonomous agents. As said before, *autonomy* is about setting our own rules, piloting our behaviors ourselves. Thanks to

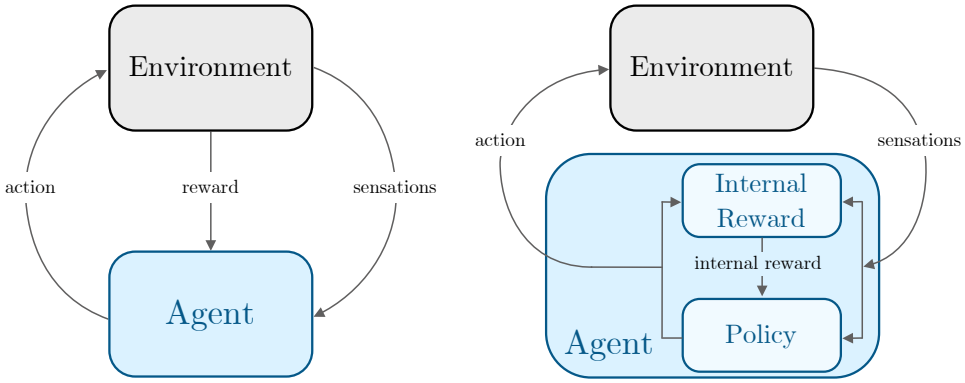


Figure 1.6: Agent-environment interaction in standard (left) vs intrinsically motivated reinforcement learning agents (right). Adapted from Barto et al. (2004) and Oudeyer & Kaplan (2009).

intrinsic motivations, artificial agents can be driven to spontaneously explore and learn about their environment, to discover new affordances and learn to master new skills. They can be asked to solve external tasks but do not require these task definitions to act in the world. In their typology, Oudeyer and Kaplan further distinguish two families of intrinsic motivations: *knowledge-based* and *competence-based* (Oudeyer & Kaplan, 2009).

Knowledge-based IMs (KB-IMs) compare the situations experienced by the agent to its current knowledge and expectations and reward it for experiencing either dissonance or resonance (Oudeyer & Kaplan, 2009). This family is then sub-divided into two categories: *information/distribution-based* IMs and *prediction-based* IMs (Oudeyer & Kaplan, 2009; Baldassarre & Mirolli, 2013; Barto et al., 2013).

Information/distribution-based approaches represent distributions of their past experiences such as the distribution of experienced states or experienced transitions. One can keep track of the distribution of experienced states via normalized counts of state visits (Thrun, 1992) or continuous approximations via density models (Bellemare et al., 2016; Tang et al., 2017). Intrinsic rewards can then be generated from these distributional models: when the agent experiences rare or novel states (Thrun, 1992; Bellemare et al., 2016), when the agent maximizes the entropy of visited states (Zhang et al., 2019), maximizes empowerment (Mohamed & Rezende, 2015; Zhang et al., 2019), distributional surprise—i.e. experiencing events that are improbable under the current distributional model—or distributional familiarity—experiencing familiar events.

Predictive-based approaches involve explicit predictions about the world (Barto et al., 2013). The agent trains predictive models from experience and can be rewarded when it experiences high-prediction errors (Pathak et al., 2017; Haber et al., 2018; Raileanu & Rocktäschel, 2020; Bougie & Ichise, 2021), learning progress in these predictions (Schmidhuber, 1991a, 2010; Lopes et al., 2012; Kim et al., 2020), information gains (Houthooft et al., 2016b), maximum surprise (Achiam & Sastry, 2017), minimum surprise (Berseth et al., 2019) or maximum uncertainty (Pathak et al., 2019; Sekar et al., 2020). A detailed review of knowledge-based IMs can be found in Linke et al. (2019).

Competence-based IMs (CB-IMs) motivate agents to generate and solve their own problems. In this category, agents need to represent, select and master self-generated goals. As a result, CB-IMs were often used to organize the acquisition of repertoires of

skills in task-agnostic environments (Baranes & Oudeyer, 2010; Rolf et al., 2010; Baranes & Oudeyer, 2013; Santucci et al., 2016, 2020; Forestier & Oudeyer, 2016b; Chitnis et al., 2020; Nair et al., 2018b; Warde-Farley et al., 2019; Colas et al., 2019a; Pong et al., 2020; Colas et al., 2020b). Goal selection can be uniform (Rolf et al., 2010; Baranes & Oudeyer, 2013; Forestier & Oudeyer, 2016b) or can be biased towards some intrinsic objective such as a drive for intermediate competence (Florensa et al., 2018; Racanière et al., 2020), diversity (Eysenbach et al., 2019; Pong et al., 2020) or high-learning progress (Baranes & Oudeyer, 2013; Forestier & Oudeyer, 2016b).

In developmental robotics, this approach enabled sample efficient learning of high-dimensional motor skills in complex robotic systems (Santucci et al., 2020), including locomotion (Baranes & Oudeyer, 2013; Martius et al., 2013), soft object manipulation (Rolf & Steil, 2013; Nguyen & Oudeyer, 2014), visual skills (Lonini et al., 2013) and nested tool use in real-world robots (Forestier & Oudeyer, 2016b). This set of methods was regrouped under the common framework of *intrinsically motivated goal exploration processes* (IMGEPs) that we describe in depth in the next section.

Exploratory Behaviors are Adaptive

Most exploratory behaviors, especially those witnessed in humans, do not benefit future task resolution, feeding, mating or survival. What would be my reward when I spend half an hour looking at the stars? What is the child's reward for digging a hole in the garden to hide a made-up treasure? How have intrinsic motivations evolved to drive such pure exploration?

In a computational study, Singh, Lewis and Barto take an evolutionary perspective on exploratory behaviors (Singh et al., 2009, 2010). Their system trains agents across two levels: the evolutionary process (phylogenetic) and the learning process (ontogenetic). The evolutionary process constructs the agent's reward function and optimizes *fitness*, a scalar measure that models the reproductive success of the agent over its lifetime. In the learning process, the agent is trained via RL to maximize the episodic returns computed from the evolved reward function. This simple mechanism allows for the emergence of reward functions reinforcing both exploitation (e.g. food consumption) and exploration (e.g. object manipulation) (Singh et al., 2009, 2010). A similar approach was recently developed to evolve curiosity-based RL algorithms (Alet et al., 2020). Across generations, the evolutionary algorithm evolves intrinsic reward functions from discrete building blocks and, surprisingly, discovers existing algorithms hand-defined by researchers (e.g. Pathak et al., 2017). These studies show that intrinsic motivations can be adaptive. Across generations, intrinsic reward functions driving agents to experience interesting situations can evolve because they confer an evolutionary advantage in the considered distribution of environments. As the real world is uncertain and changing, animals have evolved intrinsic motivations taking the form of an *epistemic hunger* (Berlyne, 1966), an adaptive tendency to collect and compress information.

Play is perhaps the most compelling example of a behavior that does not seem to serve any direct purpose. Could it be just a by-product of evolution, a form of pleasure towards the repetition of behaviors increasing reproductive success in other contexts? Scientists have proposed different evolutionary accounts of play: it might be a way to

signal the fitness of the player, to favor within-group social bonds, to practice valuable skills or to improve predictions and planning (see a review and discussion in Chu & Schulz, 2020b). These different accounts are not mutually exclusive and might all contribute to some aspects of play. Chu and Schulz, however, argue that they only partially explain the diversity and complexity of distinctively human play (Chu & Schulz, 2020b). In *pretend play*, human children invent made-up goals and generate plans to achieve them (Vygotsky, 1933; Singer & Singer, 2009). This type of play often occurs when the child is alone, can involve pointless goals (e.g. building a trap for a velociraptor Chu & Schulz, 2020b) and the use of known objects and mastered skills. Chu and Schulz thus propose an additional account of play:

> "*Distinctively human play involves manipulating our own utilities such that we invent problems for ourselves.*"
>
> "*We believe novel problems and goals may be critical to human cognition because problems constrain search, and narrowing the search space sufficiently to generate new hypotheses is arguably, far more than learning per se, the hard problem of cognition.*" (Chu & Schulz, 2020b)

Play might bring an evolutionary advantage because it drives humans to generate their own problems, hypotheses and plans. It drives them to learn how to solve problems in various contexts—i.e. to be adaptive. This hypothesis perfectly fits our autotelic framework: autotelic agents organize their self-directed exploration of the world by setting their own goals and practicing the associated skills.

Summary

In this section, we introduced the notion of *intrinsic motivations* (IM), discussed its origins in psychology and contrasted it with extrinsic motivations. Based on Oudeyer and Kaplan's work, we presented a typology of IMs into two main categories: *knowledge-based IMs* (KB-IMs) and *competence-based IMs* (CB-IMs) (Oudeyer & Kaplan, 2009).

Policies trained with KB-IMs are only trained to maximize a unique source of intrinsic reward. This objective changes as agents experience the world and learn, but agents have no control over it. For this reason, agents relying on KB-IMs are *not* goal-directed. These agents only learn one skill: optimizing the current objective or its mixture with an external objective. On the other hand, agents relying on CB-IMs are explicitly goal-conditioned: they learn to maximize several reward functions and control which reward function—i.e. which goal—they pursue. Only CB-IMs can be used to power the autonomous acquisition of repertoires of skills. This is why the remaining of this paper focuses on learning agents endowed with CB-IMs, *autotelic agents*.

1.4 Autotelic Learning with Intrinsically Motivated Goal Exploration Processes

The previous sections introduced reinforcement learning as a way to learn a single pre-defined behavior (Section 1.1), goal-conditioned RL to learn multiple pre-defined skills (Section 1.2) and intrinsic motivations to endow artificial agents with autonomy (Section 1.3). Competence-based IMs, in particular, drive agents to represent, generate, pursue and master their own goals and, thus, are essential in the design of autotelic agents.

Intrinsically motivated goal exploration processes (IMGEP) is a computational framework defining autotelic agents endowed with competence-based intrinsic motivations (Baranes & Oudeyer, 2013; Forestier et al., 2017). This section presents the core principles and general pseudo-code of IMGEP algorithms, reviews existing implementations, related approaches and discusses their limits. One contribution of this research is to overcome the limitations of existing IMGEP implementations by proposing new algorithmic architectures that leverage modern deep reinforcement learning methods.

Core Principles

IMGEPs are built on the following core principles (Forestier et al., 2017):

- *Agents generate their own goals.* In IMGEPs, goals are defined as fitness functions: functions that summarize the overall quality of a trajectory by a scalar. These are equivalent to the trajectory *return* in RL. For instance, the fitness function of the literal goal "reach coordinate x" might be the negative Euclidean distance between the gripper and x measured at the end of the trajectory.
- *Goal selection can be guided by intrinsic rewards.* CB-IMs underlie the motivation of IMGEP agents to select and pursue their own goals. Additional intrinsic motivations can drive the choice of goals to pursue—i.e. drives for competency or learning progress.
- *Two parallel processes: a goal-directed exploration and an offline data exploitation.* Agents pursuing their own goals conduct a goal-directed exploration of their environment. A batch learning algorithm uses the collected data to improve goal-reaching behaviors.
- *Cross-goal learning.* The information collected while aiming at a particular goal is systematically reused to learn about other goals. This cross-goal learning has recently been called *hindsight learning* (Andrychowicz et al., 2017).

Architecture 1 presents the general pseudo-code of the IMGEP learning architecture. The exploration and exploitation processes run in parallel. The exploration process samples goals from the goal space and collects experience by trying to reach them. The exploitation process uses exploration data to update a goal-reaching policy. This framework offers many implementation possibilities. This section surveys and discusses the limits of existing population-based implementations (POP-IMGEPs), implementations that define one policy per goal.

Architecture 1 Intrinsically Motivated Goal Exploration Process (IMGEP)

Require: Action space $\mathcal{A}$, State space $\mathcal{S}$, intrinsic reward function IR
Initialize memory empty $\mathcal{M}$
Initialize goal space $\mathcal{G}$ and goal generation policy Γ
Initialize exploitation Π and exploration Π^{ϵ} policies.
Initialize internal reward function R.
Launch asynchronously the two following loops:
loop ▷ Exploration loop
  Sample goal g from $\mathcal{G}$ with Γ
  Execute a roll-out with Π^{ϵ}_{g}, observe trajectory τ
  Compute the fitness $F = R_g(\tau)$ associated to goal g
  Compute intrinsic reward $r_i = IR(\mathcal{M}, g, F)$
  Update exploration policy Π^{ϵ} with $(\mathcal{M}, g, \tau, F)$
  Update goal generation policy Γ with $(\mathcal{M}, g, \tau, F, r_i)$
  Update memory $\mathcal{M}$ with (g, τ, F, r_i)
loop ▷ Exploitation loop
  Update internal reward function with $\mathcal{M}$
  Update exploitation Π and explorations Π^{ϵ} policies with $\mathcal{M}$
  Update goal space $\mathcal{G}$ with $\mathcal{M}$
return Π

Population-Based IMGEPs

The first IMGEPs were designed as variants of the IAC (Oudeyer et al., 2007) and R-IAC (Baranes & Oudeyer, 2009b) algorithms, two computational models of knowledge-based intrinsic motivations driving agents to explore areas of the sensorimotor space where their prediction capacities are expected to improve maximally (Baranes & Oudeyer, 2009a, 2010; Rolf et al., 2010; Baranes & Oudeyer, 2013). Instead of maximizing prediction progress, the first IMGEPs introduced a goal selection system maximizing the expected progress in *goal mastery* (Baranes & Oudeyer, 2009a).

IMGEP agents explore an *outcome space* or *behavioral space*, a space of abstract representations that compactly characterize their behaviors. Standard approaches include *time-specific* representations such as the final position of the agent in navigation tasks, the gripper in reaching tasks, or blocks in manipulation tasks. Outcomes can also be *time-extended* and represent the whole trajectory (e.g. a sequence of positions along the trajectory as in Forestier et al., 2017). Note that we can understand the fitness measure as a uni-dimensional outcome describing the *quality* of the trajectory, a point of view that will soon help us connect IMGEPs to a related approach called *quality-diversity* (Cully & Demiris, 2017).

Current IMGEP implementations rely on *population-based algorithms*, a subset of the evolutionary computation family that defines one policy per goal and learns from episodic (policy, outcome) pairs. Traditional evolutionary algorithms explore the space of controller parameters to maximize a fitness metric. They sample a set of parameters, run the controller in the environment and observe the resulting fitness. Based on the (parameter, fitness) pairs, they update the parameter sampling function to find higher-performing parameters. Instead of maximizing fitness, IMGEPs explore the *outcome space* by generating goals within it and trying to reach them (Forestier et al., 2017). In complicated tasks, most sets of parameters result in uninteresting trajectories characterized by very similar outcomes. As a result, a uniform exploration of the parameter space does

not result in a uniform exploration of the outcome space. By exploring the outcome space directly, IMGEPs bias the exploration of the parameter space in areas that lead to diverse outcomes. This was shown to be a very efficient strategy in various applications (e.g. Lehman & Stanley, 2011a; Baranes & Oudeyer, 2010; Rolf et al., 2010; Baranes & Oudeyer, 2013).

IMGEP algorithms include computational models of competence-based IMs (CB-IMs) to guide goal selection. This often takes the form of a drive towards *absolute competence progress*, often called *absolute learning progress* (LP) (Oudeyer et al., 2007; Baranes & Oudeyer, 2010, 2013). *Future* LP—what we want to optimize for—is crudely estimated by *recent* LP; see Figure 1.7. However, even recent LP is hard to estimate for a particular goal, as it requires several attempts towards that goal at different moments in time.

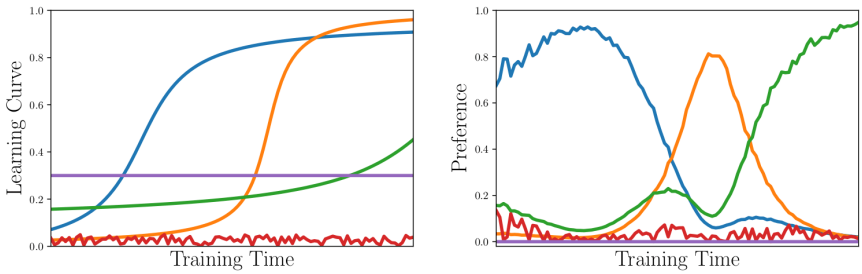


Figure 1.7: Competence-based intrinsic motivation towards learning progress. For each of the five goals, the left graph represents the average competence and the right graph represents the estimated derivative (competence progress, or learning progress). Agents favor goals where they progress the most, thus focus on the blue, orange and green goals over time. Adapted from Forestier (2019).

Robust LP measures are often obtained by partitioning the outcome space into *regions* and averaging progress measures for goals of the same region. This region-based estimation acts as a regularization of the LP model and favors robust predictions of expected LP for new sampled goals. Within each region, one can compute LP as the difference in competence measures averaged over the recent and distant pasts (Baranes & Oudeyer, 2013; Forestier & Oudeyer, 2016b; Forestier et al., 2017). The absolute value of LP is often used as an intrinsic motivation so that negative learning progress—i.e. forgetting—drives the agent to refocus on the corresponding regions. Inspired by R-IAC (Baranes & Oudeyer, 2009b), SAGG-RIAC automatically and recursively partitions the outcome space in regions so that the splits maximize the contrast between LP measures in the two sub-regions (Baranes & Oudeyer, 2013). Region selection can be formalized as a multi-arm bandit problem where arms are regions and values are LP measures. One can sample a goal by 1) sampling a region with any bandit algorithm and 2) sampling a goal from the region (e.g. uniformly, or with minimum competence).

Many algorithms were built on the efficient SAGG-RIAC. SGIM integrates human demonstrations and trains a robot to control a flexible fishing rod (Nguyen et al., 2011). Its extension SGIM-ACTS enables agents to actively choose what and how to learn by selecting: 1) which goal to target (what to learn); 2) which data collection strategy to use between autonomous exploration, mimicry and emulation (how to learn) and 3) in the mimicry and emulation strategies, which teacher to learn from (from whom to learn)

(Nguyen & Oudeyer, 2012). Another study extended SAGG-RIAC to train a simulated vocal tract to discover vocalizations autonomously (Moulin-Frier et al., 2014).

Besides their empirical achievements, IMGEP algorithms have been proposed as computational models of child development. Piaget first described child development as a succession of developmental stages (Piaget & Cook, 1952). Whereas Piaget thought that children in a specific stage had only access to a specific method for each problem, subsequent theories argued that children maintain a set of methods, including sub-optimal ones and alternate between them. In the *overlapping waves theory*, for example, Siegler argues for the adaptiveness of maintaining a diversity of solutions (Siegler, 1998). Interestingly, LP-based IMGEP implementations seem to demonstrate similar behaviors: agents mainly focus on high-LP tasks but occasionally sample yet-impossible or already-mastered goals to maintain accurate estimations of learning progress or robustify acquired skills (Oudeyer et al., 2007; Baranes & Oudeyer, 2013; Forestier & Oudeyer, 2016b,c). Previous works analyzed parallels between artificial and natural development trajectories in the context of tool-use (Forestier & Oudeyer, 2016c,b; Forestier et al., 2017) and early vocal development (Moulin-Frier et al., 2014)—see Figure 1.8. This led to the formulation of new hypotheses to explain the origins of both the diversity and regularities observed in the vocal developmental trajectories of children (Moulin-Frier et al., 2014; Oudeyer & Smith, 2016).

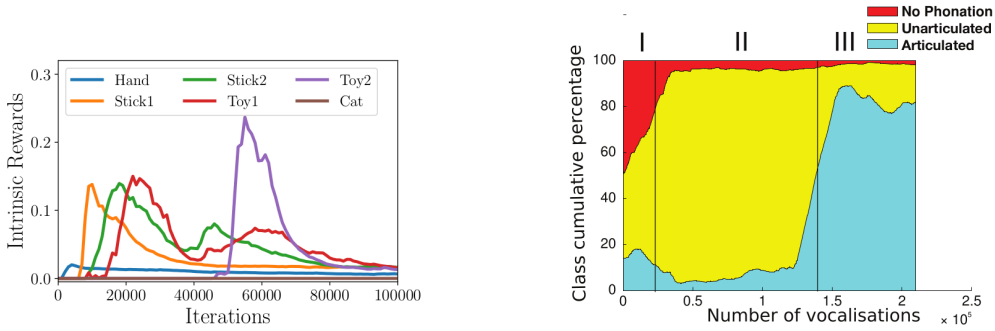


Figure 1.8: Overlapping waves in population-based IMGEPs. Left: LP-based intrinsic motivations towards different objects in a tool-use task, from Forestier (2019). Right: proportion of vocalizations without phonation (red), unarticulated (yellow) and articulated (blue) in a simulated vocal tract experiment guided by LP-based intrinsic motivations, from Moulin-Frier et al. (2014).

IMGEPs can be used as general exploration processes in other contexts. Exciting applications include the use of IMGEPs for the automatic discovery of artificial life specimens in a continuous version of the famous *Game of Life* called *Lenia* (Chan, 2019; Reinke et al., 2019), to allow human-assisted discovery in the same environment (Etcheverry et al., 2020) or to discover new states in a complex chemical system (Grizou et al., 2019).

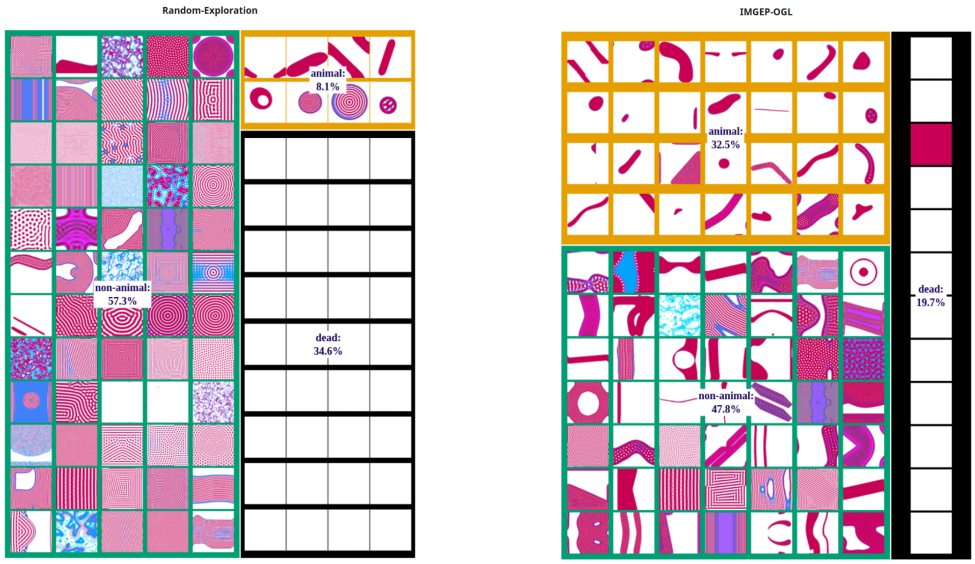


Figure 1.9: Lenia patterns discovered by random exploration (left) and IMGEP-based exploration (right), adapted from (Reinke et al., 2019).

Novelty Search and Quality-Diversity

Research in *artificial life* and *evolutionary robotics* led to two families of population-based algorithms sharing strong similarities with IMGEPs: *novelty search* (NS) and *quality-diversity* (QD). Just like IMGEPs, NS and QD algorithms both train populations of policies to explore a behavioral space using episodic (policy, outcome) pairs (NS) or (policy, outcome, fitness) triplets (QD).

Novelty Search. Novelty search is designed to generate diversity in a behavioral space (Lehman & Stanley, 2008, 2011a). Standard machine learning algorithms rely on small improvements to build up and eventually solve the task—i.e. they assume stepping stones to lie along the gradient of performance. In NS, the assumption is that solutions that are the *most novel now* will be good stepping stones to achieve *further novelty later* (Lehman & Stanley, 2011a). Here, the novelty of a controller is defined as the average behavioral distance to its k nearest neighbors in the set of previously evaluated controllers. At each step, NS selects the most novel solutions, mutates them and evaluates the outcomes of these new—hopefully behaviorally novel—solutions. As NS discovers new behaviors, the meaning of *novel* is redefined. New individuals emerge and previously novel individuals tend to become less novel, which results in the progressive exploration of the behavioral space. In some tasks where local performance improvements do not lead to global performance improvements—breaking the implicit assumption behind fitness-oriented machine learning approaches—NS can beat fitness-oriented methods *without ever knowing about the fitness function* (Lehman & Stanley, 2011a).

Quality–Diversity. Despite the successes of NS, objectives may still convey useful information to solve tasks. As the space of possible behaviors increases in size, NS can endlessly generate novel outcomes, few of which may be relevant to the task at hand.

This problem and the objective of building repertoires of high-performing behaviors led to the *quality-diversity* family (QD). QD algorithms address this issue by searching for a collection of solutions that is both diverse and locally high-performing in the external task (Lehman & Stanley, 2011b; Mouret & Clune, 2015; Cully & Mouret, 2016; Cully et al., 2015; Pugh et al., 2015; Cully & Demiris, 2017).

In contrast to multi-objective approaches that trade-off novelty for quality (read performance) or attempt to optimize a Pareto front of solutions, QD algorithms optimize for novelty and *local quality*. This is inspired by natural evolution—if two fish species might compete for resources, they will never compete with giraffes. This *local competition* is also implemented in QD algorithms—only solutions that behave similarly compete in terms of performance. Although algorithms differ in their implementations, all variants grow an archive by adding new solutions that are either novel or higher-performing than known behaviorally similar solutions; see Cully & Demiris (2017) for a general framework. The curated archive—also called *behavioral repertoire* or *behavioral map*—is a collection of high-performing and diverse controllers encountered during the learning process.

NS, QD and IMGEPs. Several parallels can be drawn between NS, QD and IMGEP algorithms. First, they all aim at generating a population of solutions leading to diverse outcomes. Second, current implementations all rely on population-based methods learning from episodic (policy, outcome) pairs. Basic implementations of IMGEP can behave similarly to NS. In such implementations, IMGEP uniformly samples a goal and obtains the corresponding policy by mutating the policy corresponding to the nearest outcome in memory (e.g. see Forestier & Oudeyer, 2016b). This tends to favor policies leading to *novel outcomes* at the border of dense behavioral clusters just like the novelty score of NS—see Figure 1.10. When IMGEPs target high-LP regions, they optimize both diversity (uniform sampling within regions) and an intrinsic, local measure of quality (making progress in that region), in a similar way to QD algorithms.

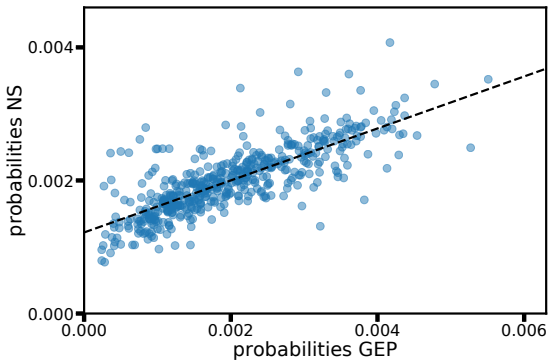


Figure 1.10: Correlation between outcomes selection probabilities in NS and a simple IMGEP implementation using the *nearest-neighbor-of-uniform-goal* strategy to select the next policy to evaluate.

The other way around, QD algorithms can also be interpreted as IMGEPs. When sampling the next policy to mutate, QD algorithms sample a behavioral cell and retrieve its associated policy. The choice of cell can be interpreted as the choice of a learning problem: the agent tries to improve (local quality) or to explore (diversity) *from that*

(outcome, policy) pair. This can be loosely seen as a goal where the behavioral description is the goal encoding and the reward function is shared across goals (positive reward is successfully added to the archive). The evolved archives in IMGEPs and QD serve the same purposes. One can define a meta-policy to reach specific behavioral goals by retrieving the corresponding policies. A recent work studies in depth the relations and differences between these methods (Chenu et al., 2021).

Just like IMGEPs, QD algorithms train agents to discover new skills—here called behaviors—but do so by considering a quality measure that integrates an external task into the learning process. The collection of behaviors needs to be diverse *and* perform well on the task—at least locally. On the other hand, IMGEPs are centered around the notion of *goals* and do not require external tasks.

Limits

Most of the IMGEP implementations presented above rely on open-loop controllers that are not reactive to the environment and thus might have trouble generalizing to novel situations: stochasticity, diverse initial conditions or procedurally-generated scenes (Forestier & Oudeyer, 2016a,b; Forestier et al., 2017). Thus far, the most successful demonstrations of NS, QD and IMGEPs algorithms have been in robotics problems with relatively ad-hoc, low-dimensional controllers (Lehman & Stanley, 2011a,b; Baranes & Oudeyer, 2013; Mouret & Clune, 2015; Cully et al., 2015; Forestier & Oudeyer, 2016a; Cully & Demiris, 2017). More expressive controllers such as the ones based on neural networks can have millions of parameters, making them difficult—though not impossible (Such et al., 2017)—to optimize with population-based methods such as genetic algorithms.

In the current implementations of IMGEPs, engineers must define goal spaces and associated reward functions *a priori*. Goal spaces are generally tailored to the learning tasks so that each outcome corresponds to a plausible physical situation (e.g. 2D final coordinates of the gripper). Reward functions are often defined via custom distance metrics between the experienced trajectory and the goal. Prior definitions of goal spaces and reward functions considerably restrict the autonomy and open-endedness of the learning agents; the agent can only learn a pre-defined and bounded set of skills. Recent approaches learns this representation space automatically by training generative models of image-based states (Péré et al., 2018; Laversanne-Finot et al., 2018; Cully, 2019). Training *variational auto-encoders* (VAE) (Kingma & Welling, 2014) to model the distribution of images can lead to low-dimensional representations for the control of a single object (Péré et al., 2018) and can be extended to learn disentangled multi-object representations (Laversanne-Finot et al., 2018). These first works implemented representation learning and reinforcement learning in sequence, but both can be performed in parallel (Cully, 2019). These studies, however, are limited to relatively simple setups with one or two objects.

Existing goal selection mechanisms based on LP have been restricted to low-dimensional outcome spaces, as higher-dimensional representations lead to a combinatorial explosion of the number of regions. Two approaches might help circumvent this problem. The outcome space can be split into a fixed number of cells via Voronoi tesselations (Vassiliades et al., 2017). This helps to scale the outcome space to thousands of dimensions with no drop in

performance. Another approach called GRIMGEP proposes to embed high-dimensional images in low-dimensional representations with a VAE, then to cluster the latent space and track LP in each of them. This approach is then used to detect and filter controllable areas of the goal space—i.e. high-LP clusters—before conducting any further goal selection (Kovač et al., 2020).

Summary

This section introduced the computational framework of *intrinsically motivated goal exploration processes* (IMGEP) and surveyed existing population-based IMGEP methods and related approaches. The IMGEP family trains learning agents to select and reach their own goals, thus offers an interesting framework towards our main objective: the autonomous acquisition of skill repertoires. Current POP-IMGEP implementations have achieved great successes in a wide range of control problems (tool use, vocalization, unknown space exploration, etc.) and were used to model aspects of children development (overlapping waves, early vocalization). However, current approaches suffer from several of limitations: they are not robust to changing environmental conditions (stochasticity, diverse initial conditions, procedural generation), they cannot train high-dimensional policies and require pre-defined goal spaces and reward functions. Although some methods have emerged to learn goal spaces, they are limited to simple visual environments with few objects and only represent concrete goals: time-specific goals as targets in the state space.

Recent advances in *deep reinforcement learning* (DRL) seem to offer solutions to some of these problems. DRL approaches train deep neural controllers that are robust to diverse environmental conditions and can handle high-dimensional state spaces. These methods can be coupled with other deep learning techniques to augment the interaction capabilities of agents: e.g. sequence encoders to receive linguistic inputs, deep generative models to encode visual inputs. The recent successes in language-conditioned RL also offer the possibility to represent abstract goals via linguistic descriptions. The next section builds on the frameworks of IMGEP and DRL to propose novel architectures at their intersection.

1.5 Autotelic Learning with RL-Based IMGEPs

The previous sections introduced RL and goal-conditioned RL as a mathematical and computational framework to train agents to learn multiple pre-defined skills. We then introduced the concept of intrinsic motivations and their various computational implementations to endow artificial agents with autonomy. This led us to introduce the IMGEP framework, the family of algorithms that leverage competence-based intrinsic motivations to implement agents that represent, generate, pursue and master their own goals. However, as we saw in the last section, current implementations show various limitations. We think some of them can be overcome by leveraging modern optimization methods from *deep reinforcement learning* (DRL). This section integrates the concepts and framework presented in the previous ones into a new family of algorithms: *rl-based IMGEPs* (RL-IMGEP). RL-IMGEPs are autotelic RL algorithms. They use DRL methods to train goal-conditioned policies (like GC-RL) endowed with competence-based intrinsic

motivations to target the autonomous acquisition of open-ended repertoires of skills (like IMGEPs).

We first present existing intrinsically motivated reinforcement learning approaches and discuss how they fail to solve our problem. We then cast the problem of the autonomous acquisition of open-ended repertoires of skills into the RL framework as the *autotelic RL problem*. Finally, we introduce a new family of algorithms called *rl*-based IMGEP (RL-IMGEP) to target the autotelic RL problem with a combination of ideas from the IMGEP framework and state-of-the-art DRL approaches. As we introduce this family, we review recent works in this new light.

Intrinsically Motivated Reinforcement Learning

Research in RL has been interested in intrinsic motivation from the start (Schmidhuber, 1990, 1991a; Kaplan & Oudeyer, 2004), but primarily focused on knowledge-based intrinsic motivations. The general idea is to replace the external reward function with an intrinsic one. Intrinsically motivated RL agents either target their intrinsic reward alone (e.g. Pathak et al., 2017; Burda et al., 2019) or use it as an exploration bonus in addition to the external rewards (e.g. Bellemare et al., 2016; Tang et al., 2017; Pathak et al., 2017; Burda et al., 2019; Raileanu & Rocktäschel, 2020; Badia et al., 2020b,a). This approach is useful to generate exploratory behaviors in environments characterized by sparse or deceptive rewards but only lead to the emergence of a single behavior—the one maximizing the current intrinsic reward (when used alone) or the external reward (when annihilated to the benefit of the external reward). As argued before, this does not allow the emergence of repertoires of skills. Only competence-based intrinsic motivations can organize such autotelic exploration.

In parallel with developmental robotics, RL researchers have also explored forms of competence-based IMs. An interesting approach proposes to train a set of policies to reach goals expressed as target states in a discrete grid world, one exploration and one exploitation policy per goal (Stout & Barto, 2010). The exploration policy is trained to maximize improvements in the expected return of the corresponding exploitation policy (competence maximization). Goal selection is implemented by a greedy policy selecting the goal associated with the maximum expected return improvement averaged over the state space. In parallel, exploitation policies are trained to reach their goal via standard RL, reusing data collected by all exploration policies (cross-goal learning). In the sub-field of hierarchical reinforcement learning, early works focused on the intrinsically motivated discovery of sub-goals, although the main high-level task remained externally defined (McGovern & Barto, 2001; Simsek & Barto, 2004, 2008). These early approaches targeted external tasks at the high level and considered pre-defined goal spaces and reward functions at the low level. Instead, we propose to leverage the more efficient cross-goal learning abilities of goal-conditioned policies (UVFA Schaul et al., 2015) to target self-generated goals and learn both goal representations and reward functions.

The Autotelic Reinforcement Learning Problem

It is now time to formally define our main objective: the autonomous acquisition of repertoires of skills. In this problem, the agent is set in an open environment without pre-defined goals or external rewards and needs to acquire a repertoire of skills. We can formalize this within the RL framework by removing the space of reward functions from the MDP of the multi-goal RL problem: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0\}$. Instead of the traditional policy π, the agent needs to learn the goal space $\mathcal{G}$, goal representations $\mathcal{Z}_g$, a goal-conditioned reward function R_g and a goal-conditioned policy π_g. Agents draw goals from $\mathcal{G}$, represent them by z_g, reach them with π_g and assess their completion with R_g (see Figure 1.11). We call these *autotelic reinforcement learning problems*.

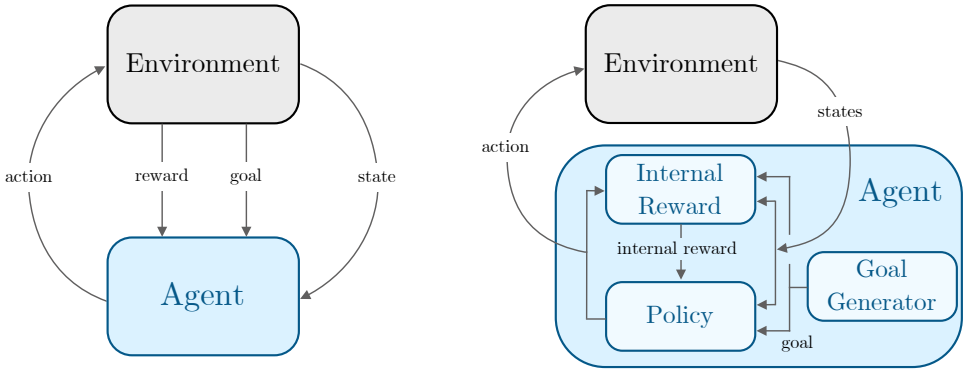


Figure 1.11: Agent-environment interaction in goal-conditioned RL (left) and autotelic RL (right). Adapted from Barto et al. (2004) and Oudeyer & Kaplan (2009).

It is often straightforward to evaluate standard goal-conditioned RL agents because we know the set of possible interactions in advance. We can compute their average competence over a set of goals drawn from the training distribution and their generalization performance over goals from test distributions. In contrast, our autotelic agents evolve in reward-free environments and learn to represent and form their own set of skills. In this context, the space of possible behaviors might quickly become intractable for the experimenter, and their evaluation becomes subjective.

Interestingly, the evaluation of autotelic agents presents similar challenges to the evaluation of self-supervised learning systems such as generative adversarial networks (GAN) (Goodfellow et al., 2014) or self-supervised language models (Devlin et al., 2019; Brown et al., 2020). In both cases, learning is *task-agnostic* and outputs comparisons are subjective: comparing acquired repertoires of skills might be just as complicated as comparing images generated by a GAN. One approach might be to think about the properties we expect from our autotelic agents. Here is a list of such properties: the discovered sets of skills should 1) be diverse, 2) be human-interpretable, 3) generalize to additional similar skills, 4) help agents to explore their environment, 5) be robust to non-stationarities in the environment, 6) be transferable to downstream tasks specified by humans, 7) be composable to form more complex skills in an open-ended way. Let us list some approaches to evaluate autotelic agents:

- **Measuring exploration:** one can compute task-agnostic exploration proxies such as the entropy of the distribution of visited states, state or outcome space coverage (e.g. coverage of the high-level x-y state space in mazes Benureau & Oudeyer, 2016; Florensa et al., 2018). One can also count interactions from a set of *interesting* interactions defined subjectively by the experimenter; e.g. interactions with objects (see Chapter 5).

- **Measuring generalization:** The experimenter can subjectively define a set of relevant target goals and prevent the agent from training on them (e.g. with rejection sampling). Evaluating agents on this held-out set at test time provides a measure of generalization (Hill et al., 2019; Ruis et al., 2020), although it is biased towards what the experimenter judges to be relevant. Note that this requires the possibility for the experimenter to express goals the agent can represent (e.g. linguistic goals or target images). This can be challenging is learned goal spaces (e.g. Eysenbach et al., 2019).

- **Measuring transfer learning:** Here, we view the autotelic exploration of the environment as a pre-training phase to bootstrap learning in a subsequent downstream task. In the downstream task, the agent is trained to achieve external goals. We can report its performance and learning speed on these goals. This is akin to the evaluation of self-supervised language models, where the reported metrics evaluate performance in various downstream tasks (e.g. Brown et al., 2020).

- **Opening the black box:** Investigating internal representations learned during autotelic exploration is often informative. One can investigate properties of the goal generation system (e.g. does it generate out-of-distribution goals?), investigate properties of the goal embeddings (e.g. are they disentangled?) (e.g. Florensa et al., 2018; Laversanne-Finot et al., 2018; Pong et al., 2020). One can also look at the agents' learning trajectories, especially when they implement their own learning curriculum (e.g. Moulin-Frier et al., 2014; Forestier & Oudeyer, 2016c).

- **Measuring robustness:** Autonomous learning agents evolving in realistic environments should be robust to the properties of natural environments. This includes large environments where possible interactions might vary in terms of difficulty (trivial interactions, impossible interactions, interactions whose result is purely stochastic, thus prevents any learning progress). Environments can also include distractors (e.g. non-controllable objects) and various forms of non-stationarities (e.g. sensory perturbations, evolving dynamics).

Goal-Conditioned IMGEPs

Intrinsically motivated goal exploration processes (IMGEP) is the family of algorithms that tackle the autonomous acquisition of open-ended skill repertoires. Whereas previous approaches relied on population-based optimization mechanisms (see review and discussion of POP-IMGEPs in Section 1.4), this research introduces IMGEP agents leveraging state-of-the-art DRL methods. We call this family of autotelic RL agents *rl-based intrinsically motivated goal exploration processes* or RL-IMGEP. One can see this family as the convergence of IM-based developmental robotics and deep reinforcement learning. RL-IMGEPs

must be distinguished from standard goal-conditioned RL agents targeting pre-defined goals—we call this family *goal-conditioned externally motivated goal exploration processes* or RL-EMGEP. Figure 1.12 represents these different families, how they relate to each other and should help the reader make sense of the terminology.

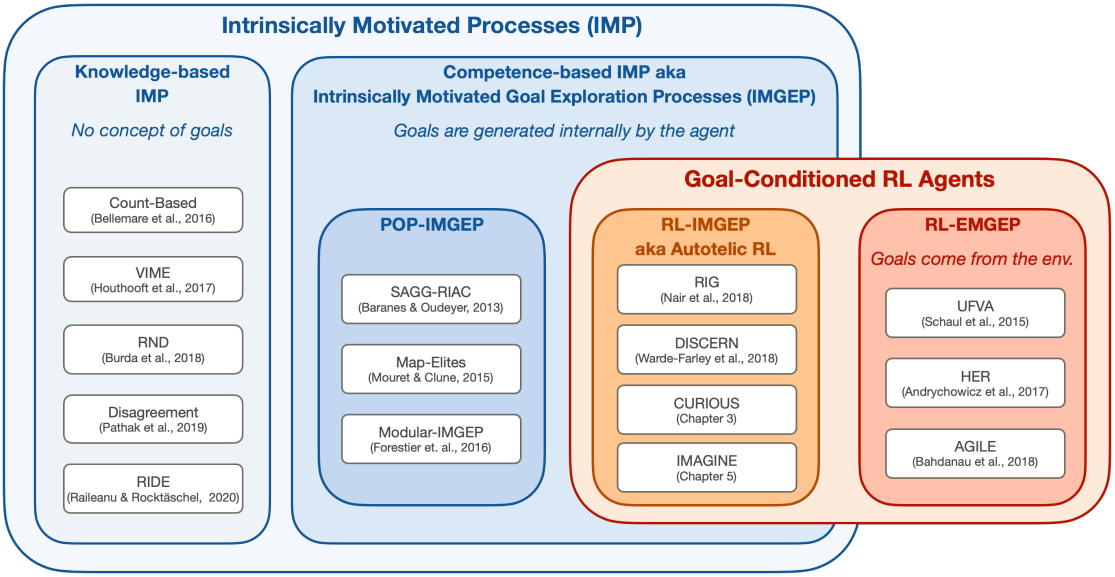


Figure 1.12: A typology of intrinsically-motivated and/or goal-conditioned RL approaches. POP-IMGEP, RL-IMGEP and RL-EMGEP refer to *population-based intrinsically motivated goal exploration processes*, *rl-based* IMGEP and *rl-based externally motivated goal exploration processes*, respectively. POP-IMGEP, RL-IMGEP and RL-IMGEP all represent goals, but knowledge-based IMs do not. While IMGEPs (POP-IMGEP and RL-IMGEP) generate their own goals, RL-EMGEPs require externally-defined goals. Our research is interested in RL-IMGEPs, the intersection of *goal-conditioned RL agents* and *intrinsically motivated processes*, that is, the set of methods training autotelic learning agents to generate and pursue their own goals with goal-conditioned RL algorithms.

RL-IMGEPs build on standard goal-conditioned RL methods and incorporate mechanisms to represent, generate and select their own goals—see Figure 1.13. This requires the agent to learn several additional modules compared to standard externally-motivated RL algorithms. Agents need to learn a goal representation function that generates goal embeddings—i.e. compact goal representations. They also need to represent the space of possible goals and a sampling mechanism selecting relevant goals to target and learn about. Finally, they learn a goal-conditioned reward function to measure progress towards goals and generate learning signals to train the goal-conditioned policy. Only RL-IMGEPs agents implementing all these modules can be considered truly autotelic.

The next sections delve into existing methods to learn each of these modules. The first section surveys mechanisms used to learn goal representations (goal embeddings, goal spaces and goal-conditioned reward function), whereas the second section focuses on goal selection mechanisms.

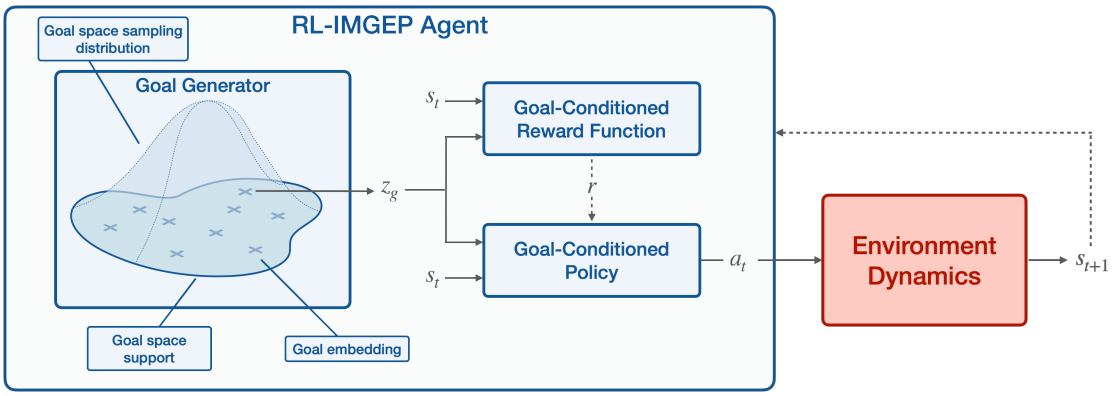


Figure 1.13: Representation of the different learning modules in a RL-IMGEP algorithm. In contrast, externally motivated goal exploration processes (RL-EMGEPs) only train the goal-conditioned policy and assume *external* goal generator and goal-conditioned reward function.

How to Learn Goal Representations?

Most existing approaches tackle the multi-goal RL problem (Section 1.2), where goal spaces and associated rewards are pre-defined by the engineer and are part of the problem definition. For example, navigation and manipulation tasks pre-define goal spaces (e.g. target agent position and target block positions respectively) and use the Euclidean distance to compute rewards (Andrychowicz et al., 2017; Nair et al., 2018a; Plappert et al., 2018a; Li et al., 2020; Lanier et al., 2019; Ding et al., 2019; Schaul et al., 2016; Florensa et al., 2018). Autotelic agents, however, need to learn goal representations. While individual goals are represented by their embeddings and associated reward functions, representing multiple goals also requires representing of the *support* of the goal space, i.e. the space of *valid goals* the agent can sample from, see Figure 1.13.

Learning goal embeddings. Some approaches assume the pre-existence of a goal-conditioned reward function but learn to represent goals by learning goal embeddings. This is the case of instruction-following approaches, which receive linguistic instructions and rewards from the environment (thus are RL-EMGEP), but learn goal embeddings jointly with the policy during policy learning (Hermann et al., 2017; Chan et al., 2019; Jiang et al., 2019; Bahdanau et al., 2019b; Hill et al., 2019; Cideron et al., 2020b; Lynch & Sermanet, 2020). When goals are images, goal embeddings can be learned via generative models of states and the reward function is generally assumed to be a fixed distance metric computed in the embedding space (Nair et al., 2018b; Florensa et al., 2019; Nair et al., 2020).

Learning reward functions. A few approaches go even further and learn their own goal-conditioned reward function. In instruction-following setups, the reward function can be learned from expert demonstrations via inverse reinforcement learning methods (Fu et al., 2019; Bahdanau et al., 2019a). In visual goals settings, two studies proposed to learn a distance metric estimating the square root of the number of steps required to move from any state s_1 to any s_2 (Venkattaramanujam et al., 2019; Hartikainen et al., 2020). One can also use the mutual information between states encoded in latent space

focused on controllable aspects of the environment as distance metric (Warde-Farley et al., 2019). In both cases, agents receive rewards for getting closer to their goal. Another approach is to train agents to develop a set of skills leading to maximally discriminable distributions of states (Gregor et al., 2017; Eysenbach et al., 2019; Sharma et al., 2020). Here, agents are rewarded for experiencing states that are easy to discriminate across skills, while a discriminator is trained to infer the skill z_g from the visited states. Except from the language-based approaches (Fu et al., 2019; Bahdanau et al., 2019a), all these methods set their own goals and learn their own goal-conditioned reward function, thus are RL-IMGEPs.

Learning the supports of goal distributions. To represent collections of goals, one also needs to represent the support of the goal distribution, which embeddings correspond to valid goals and which do not. Most approaches consider a pre-defined, bounded goal space in which any point is a valid goal (e.g. target positions within the boundaries of a maze, target block positions within the gripper's reach, Schaul et al., 2015; Andrychowicz et al., 2017; Nair et al., 2018a; Plappert et al., 2018a; Li et al., 2020; Lanier et al., 2019; Ding et al., 2019). However, not all approaches assume pre-defined goal spaces. Some approaches use the set of previously experienced states or representations to form the support of the goal distribution (Veeriah et al., 2018; Ecoffet et al., 2021). GOAL-GAN trains a GAN to model the distribution of visited outcomes. Taken as a goal generator, it models the distribution of goals and thus its support (Florensa et al., 2018). In the same vein, approaches that handle image-based goals usually train a VAE to model the distribution of visited visual states and use it as a goal distribution (Nair et al., 2018b; Pong et al., 2020; Nair et al., 2020). In both cases, goals generated by the generative model are considered valid.

How to Select Goals?

Once autotelic agents have represented their goals and goal spaces, they need to specify a goal selection policy. Although agents can sample their goal space uniformly, informed goal selection can be a way for agents to organize their learning curriculum automatically.

Automatic curriculum learning for goal selection. *Automatic curriculum learning (ACL)* is defined as *"the family of mechanisms that automatically adapt the distribution of training data by adjusting the selection of learning situations to the capabilities of learning agents"*—see our review in Portelas et al. (2020b). Although ACL can be applied to the control of any element of the task MDPs, we here focus on the automatic selection of goal—i.e. of reward functions—to optimize long-term goal mastery.

One may wonder why such mechanisms would be useful. In real-world scenarios, goal spaces can be too large for the agent to master all goals in its lifetime. Some goals might be trivial, others impossible. Sometimes, goals might be reached by chance, although the agent cannot make any progress on them. Some goals might be reachable only after the agent mastered more basic skills. For all these reasons, it is essential to let intrinsically motivated agents optimize their goal selection mechanism. Like any ACL mechanism, automatic goal selection aims to maximize the long-term performance improvement (distal

objective). As this objective is usually not directly differentiable, automatic goal selection techniques rely on a proximal objective usually formalized as a competence-based intrinsic motivation. In this section, we look at various proximal objectives used in ACL strategies to organize goal selection. Interested readers can refer to Portelas et al. (2020b) and Romac et al. (2021) for a review and a benchmark of ACL methods for DRL.

Intermediate difficulty. Intermediate difficulty can be used as a proxy for long-term performance improvement. This follows the intuition that focusing on goals of intermediate difficulty results in short-term learning progress that, eventually, turns into a long-term performance increase. GOAL-GAN assigns feasibility scores to goals as the proportion of time the agent successfully reaches it (Florensa et al., 2018). Based on this data, a GAN is trained to generate goals of intermediate difficulty, whose feasibility scores are contained within an intermediate range. Other methods train an RL policy to propose challenging goals to the main learning policy (Sukhbaatar et al., 2018; Campero et al., 2021). The goal selection policy is rewarded for setting goals that are neither too easy nor impossible. Recently, Zhang and colleagues proposed to train an ensemble of value functions and to select goals that maximize the disagreement between value functions from the ensemble (Zhang et al., 2020). Indeed, value functions agree when the goals are too easy (the agent is always successful) or too hard (the agent always fails) but disagree for goals of intermediate difficulty.

Uniform feasibility. Uniform feasibility is a variant of the intrinsic motivation towards intermediate feasibility (Racanière et al., 2020). Training a goal generator to sample goals of uniform feasibility seems to improve stability and performance on complex visual tasks compared to GOAL-GAN (Florensa et al., 2018; Racanière et al., 2020).

Novelty–diversity. Intrinsic motivations towards novelty propose to select goals in sparse areas of the goal space (Pong et al., 2020; Warde-Farley et al., 2019; Pitis et al., 2020). After training a density model on the history of experienced outcomes, one can sample goals to maximize the diversity of outcomes (Warde-Farley et al., 2019; Pong et al., 2020) or skew sampling even more towards novel outcomes (Pitis et al., 2020). GRIMGEP is a complementary approach from the IMGEP framework that proposes to filter controllable areas of the goal space before undertaking a novelty-based goal selection (Kovač et al., 2020).

Short-term learning progress. As discussed in the review of IMGEP approaches, a common way to sample goals is to estimate the learning progress of the agent in different regions of the goal space and to bias sampling towards areas of high absolute learning progress using bandit algorithms (Baranes & Oudeyer, 2013; Forestier & Oudeyer, 2016b; Blaes et al., 2019; Fournier et al., 2018, 2019). In a multi-task setting, ALP-GMM models LP in a continuous task representation space with mixtures of Gaussians. The sampling of task parameters is then two-fold: first, sample a Gaussian with LP, then sample from that Gaussian (Portelas et al., 2020a).

Hierarchical reinforcement learning for goal sequencing. Hierarchical reinforcement learning (HRL) can organize goal sequencing (Dayan & Hinton, 1993; Sutton et al., 1998, 1999; Precup, 2000). In HRL, a high-level policy is trained via RL or planning to generate sequences of goals for a lower-level policy to maximize a higher-level reward. This allows the decomposition tasks with long-term dependencies into simpler goal-oriented

sub-tasks. Low-level policies can be implemented by traditional goal-conditioned RL algorithms (Levy et al., 2019; Röder et al., 2020) and are either trained independently from the high-level policy (Kulkarni et al., 2016; Frans et al., 2018) or jointly (Levy et al., 2019; Nachum et al., 2018; Röder et al., 2020). Most approaches consider hand-defined spaces for the sub-goals (e.g. positions in a maze), but recent approaches propose to use the state space directly (Nachum et al., 2018) or to learn the sub-goal space (e.g. Vezhnevets et al., 2017; Nasiriany et al., 2019).

Figure 1.14 presents a history of QD, POP-IMGEP, intrinsically motivated DRL and goal-conditioned DRL. It helps to see the historical progressions of the algorithms we reviewed in this chapter. Table 1.1 summarizes the surveyed approaches organized along four dimensions: 1) the goal type; 2) whether goal embeddings are learned; 3) whether reward functions are learned and 4) the goal sampling method. We give the label RL-IMGEP to all algorithms that can be seen as sampling their own goals. Some of these algorithms might be incomplete RL-IMGEPs as they do not learn all of the modules represented in Figure 1.13. It is important to note that very few of these approaches recognize themselves as *intrinsically motivated* or even consider the autonomous acquisition of skill repertoires as their target. A contribution of this section is to offer a novel viewpoint, from which all these mechanisms can be seen as essential components of autotelic agents targeting the autonomous acquisition of skill repertoires.

Summary

This section framed the autonomous acquisition of skill repertoires as the autotelic RL problem and bridged the gap between population-based IMGEP formulations and reinforcement learning methods. RL-IMGEPs is the family of autotelic RL agents at their intersection. It trains goal-conditioned policies to represent, select and pursue their own goals autonomously with DRL.

As we will see in our experiments (Chapters 2, 3, 5) and 6, relying on DRL optimization methods allows RL-IMGEPs to overcome the limitations of POP-IMGEP. RL-IMGEPs learn goal representations, handle high-dimensional inputs and demonstrate better robustness. On the other hand, the IMGEP approach endow RL agents with the ability to represent and select their own goals, to become truly autonomous in their acquisition of skill repertoires.

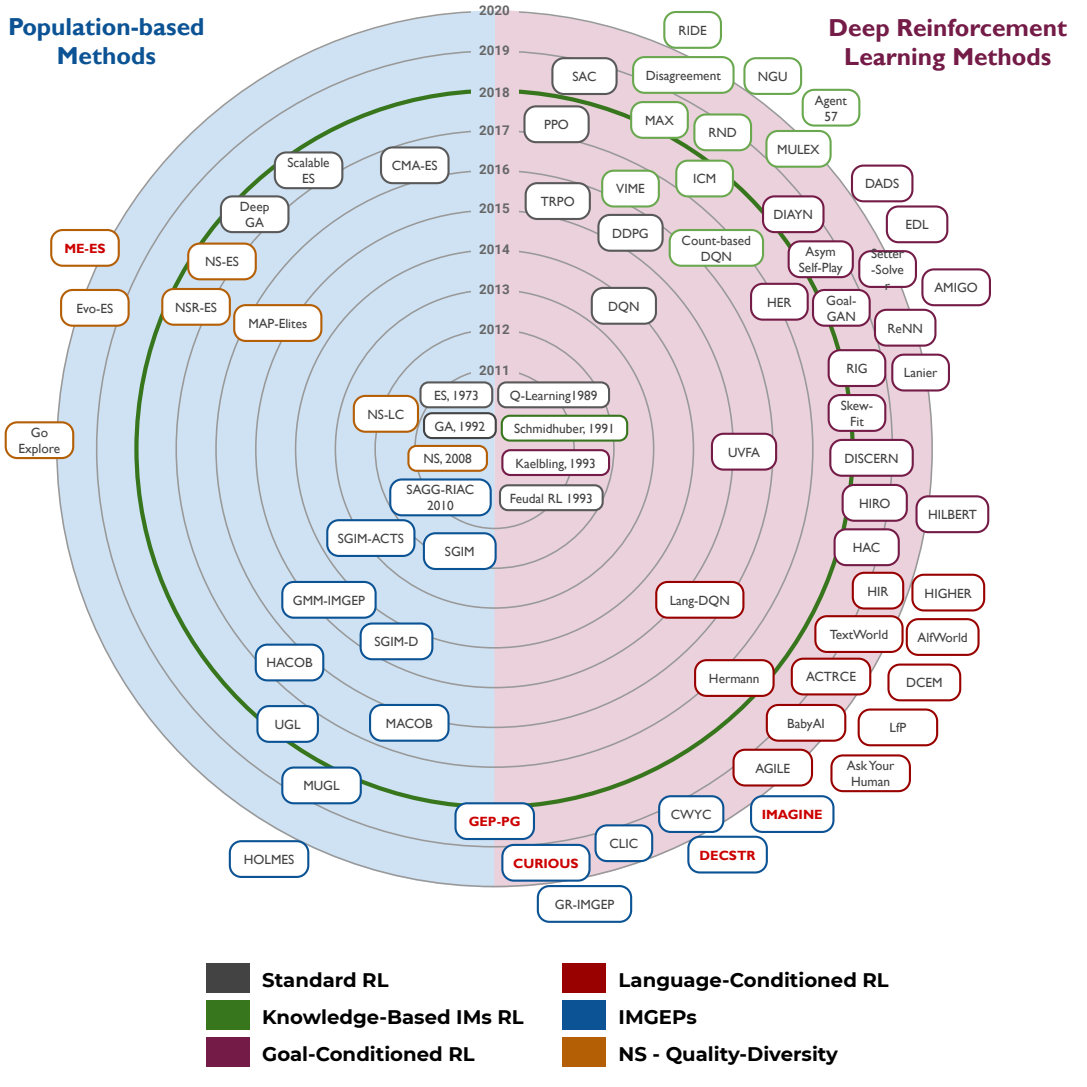


Figure 1.14: An history of deep reinforcement learning and population-based methods. The research presented in this manuscript started in November 2017 (green ring). Our contributions are indicated in red. Colors indicate the most relevant categories, although some approaches can belong to several (e.g. all language-conditioned RL approaches are also goal-conditioned RL). We list a few seminal works pre-2011. This representation is, of course, not exhaustive. The list of references can be found in Appendix D. Inspired from the history in Justesen et al. (2020a).

Approach	Goal Type	Goal Embeddings	Reward Functions	Goal Sampling
RL-IMGEPs that assume goal embeddings and reward functions				
Fournier et al. (2018)	Target features (+tolerance)	Pre-def	Pre-def	LP-Based
HAC, Levy et al. (2019)	Target features	Pre-def	Pre-def	HRL
HIRO, Nachum et al. (2018)	Target features	Pre-def	Pre-def	HRL
CURIOUS, Chapter 3	Target features	Pre-def	Pre-def	LP-based
CLIC, Fournier et al. (2019)	Target features	Pre-def	Pre-def	LP-based
CWYC, Blaes et al. (2019)	Target features	Pre-def	Pre-def	LP-based
GO-EXPLORE, Ecoffet et al. (2021)	Target features	Pre-def	Pre-def	Novelty
NGU, Badia et al. (2020b)	Objective Balance	Pre-def	Pre-def	Uniform
AGENT57, Badia et al. (2020a)	Objective Balance	Pre-def	Pre-def	Meta-learned
DECSTR, Chapter 6	Binary problem	Pre-def	Pre-def	LP-based
RL-IMGEPs that learn their goal embedding and assume reward functions				
RIG, Nair et al. (2018b)	Target features (images)	Learned (VAE)	Pre-def	From VAE prior
GOALGAN, Florensa et al. (2018)	Target features	Pre-def + GAN	Pre-def	Intermediate difficulty
Florensa et al. (2019)	Target features (images)	Learned (VAE)	Pre-def	From VAE prior
SKEW-FIT, Pong et al. (2020)	Target features (images)	Learned (VAE)	Pre-def	Diversity
SETTER-SOLVER, Racanière et al. (2020)	Target features (images)	Learned (Gen. model)	Pre-def	Uniform difficulty
MEGA, Pitis et al. (2020)	Target features (images)	Learned (VAE)	Pre-def	Novelty
CC-RIG, Nair et al. (2020)	Target features (images)	Learned (VAE)	Pre-def	From VAE prior
AMIGO, Campero et al. (2021)	Target features (images)	Learned (with policy)	Pre-def	Adversarial
GRIMGEP, Kovač et al. (2020)	Target features (images)	Learned (with policy)	Pre-def	Diversity and ALP
Full RL-IMGEPs				
DISCERN, Warde-Farley et al. (2019)	Target features (images)	Learned (with policy)	Learned (similarity)	Diversity
DIAYN, Eysenbach et al. (2019)	Discrete skills	Learned (with policy)	Learned (discriminability)	Uniform
Hartikainen et al. (2020)	Target features (images)	Learned (with policy)	Learned (distance)	Intermediate difficulty
Venkattaramanujam et al. (2019)	Target features (images)	Learned (with policy)	Learned (distance)	Intermediate difficulty
IMAGINE, Chapter 5	Binary problem (language)	Learned (with reward)	Learned	Uniform + Diversity

Table 1.1: **A classification of RL-IMGEP approaches.** The classification groups algorithms depending on their degree of autonomy: 1) RL-IMGEPs that rely on pre-defined goal representations (embeddings and reward functions); 2) RL-IMGEPs that rely on pre-defined reward functions but learn goal embeddings and 3) RL-IMGEPs that learn complete goal representations (embeddings and reward functions). For each algorithm, we report the type of goals agents pursue, whether they learn goal embeddings, reward functions and how they sample goals. We mark in bold algorithms taking a developmental approach and explicitly tackling the problem of the autonomous acquisition of repertoires of skills.

1.6 Exploration Methods for Reinforcement Learning

Exploratory behaviors are not shaped by external reinforcements and punishments, i.e. not directed towards external tasks. So far, we have discussed how autotelic agents could organize the acquisition of repertoires of skills on their own, without external tasks. However, RL researchers are often more interested in the organization of exploratory behaviors *to help the resolution of external tasks*, i.e. to help *exploitation*. This section describes exploration problems faced by reinforcement learning algorithms and introduces a typology of exploration mechanisms to overcome them. We argue that autotelic exploration, i.e. the pursuit of self-generated goals, is an efficient way to explore.

On the Exploration Problem in RL

RL methods are active learning algorithms (Thrun, 1995; Cohn et al., 1996); they actively sample the data they train on. In that context, one can understand *exploration* as the active sampling of relevant experiences to favor further task-oriented learning. Some tasks require less exploration than others. An agent that must reach the end of a corridor and is rewarded by the negative distance to this point barely needs any exploration. All it needs to do is exploit the gradient of performance, because getting closer and closer to its goal will quickly yield the expected result. Now picture the same agent in a maze. Following the gradient of performance will surely lead the agent to get stuck in a nearby corner—see Figure 1.15. Here, the agent should explore, learn to navigate the maze and find new rooms if we ever hope for it to find the exit. We can mention two types of hard exploration problems: deceptive and sparse. In deceptive exploration problems, the reward function provides a dense signal but leads to local optima—the agent gets stuck in corners. In sparse exploration problems, rewards are only delivered when the task is completed—e.g. when the agent finds the exit. Because it never experiences any positive reinforcement, our agent wanders without objective and never finds the exit.

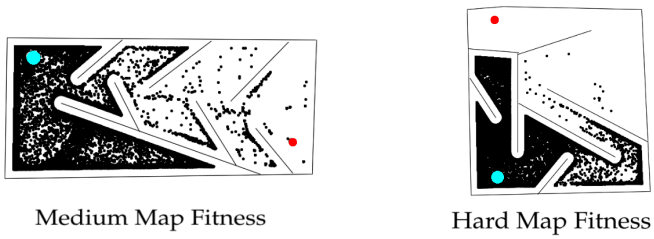


Figure 1.15: Medium and hard deceptive mazes from Lehman & Stanley (2011a). Cyan and red dots respectively indicate the initial position and goal. Black dots show the final positions of agents attempting to maximize performance, i.e. minimize their distance to the goal. The hard maze is strongly deceptive: the agent needs to get further before it can get closer.

How should we explore? The intra-life optimization of experience collection to maximize performance on future and yet-unknown tasks is intractable. Instead, evolution must have favored the development of exploratory behaviors and heuristics to favor task-oriented learning in the distribution of environments experienced by humans (see

discussion of the evolutionary origins of IMs in Section 1.3 and Singh et al., 2010). To organize exploration, most RL researchers tend to hard-code exploration heuristics or to motivate them with intrinsic motivations. One can classify exploration mechanisms depending on three main design choices:

1. **What to explore?** Exploration mechanisms target a space to explore (e.g. state space, outcome space, trajectory space).
2. **How to explore, what to control?** Exploration mechanisms control some aspects of behaviors, the *choice space* (actions, parameters, goals).
3. **How to explore, how to sample controls?** Exploration mechanisms organize exploratory behaviors by sampling controls from the choice space. The sampling strategy can vary (undirected vs directed, temporally structured vs temporally unstructured).

Let us get back to our example. Here, the robot explores a maze represented as a 2D outcome space that abstracts information about the robot's state and focuses on its 2D spatial position. An exploration mechanism might generate random goals within that outcome space and orient the robot's behavior towards them. The goal selection mechanism might itself focus on novel goals; goals in parts of the maze that the agent does not reach often. This exploration strategy explores a 2D outcome space (1) by sampling a goal space (2) with a novelty bias (3).

What to Explore?

First, we need to talk about *what to explore*. In RL, one can explore the state space, outcome spaces or the space of state or outcome trajectories. Let us get back to our robot in its maze. In state-space exploration, the agent tries to discover new configurations of its perceptual inputs s. This can include new configurations of its joint positions or new visual patterns on the maze walls for example. As said above, agents could compactly represent their state by their current (x, y) position in the 2D space and explore this resulting outcome space. Other exploration mechanisms might want to explore the space of state or outcome trajectories. Exploring in outcome space might be more efficient because it abstracts away features of the state space that might be irrelevant for exploration (e.g. the color of the walls) and thus restricts the space to explore. However, it requires defining or learning outcome representations. Now that we discussed *what* we can explore, let us discuss *how* to explore.

How to Explore? What to Control?

The agent exerts control on its environment and thus on the collected experience via its actions. Exploration mechanisms can control these actions at various levels: controlling actions directly, controller parameters or goals.

One set of exploration methods define agents that observe the current state and sample *actions* thought to generate informative data for further learning. The crudest exploration

scheme is *epsilon-greedy* exploration: sampling random actions (in the discrete action case, Mnih et al., 2015) or random perturbations on actions (in the continuous action case, Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018b) to diversify behavior and experience. Exploratory actions can also be generated by *exploration policies*. For example, *deep exploration via bootstrapped Q-networks* trains an ensemble of Q-networks and, at each episode, samples actions from one of them selected at random as a way to explore. It relies on the sub-optimality of each Q-network to drive a consistent exploration (Osband et al., 2016). Exploration policies can also be trained with knowledge-based IMs to generate exploratory actions (e.g. Bellemare et al., 2016; Pathak et al., 2017; Burda et al., 2019; Achiam & Sastry, 2017; Haber et al., 2018; Bougie & Ichise, 2021).

Controlling *policy parameters* to explore is the strategy used in all algorithms from the field of evolutionary computation (Stulp & Sigaud, 2013; Eiben & Smith, 2015; Qian & Yu, 2021). A simple—but not very effective—method is to sample random parameters from some parameter space. More standard methods involve the mutation of good controllers via parameter noise or cross-overs (Rechenberg, 1973; Back et al., 1991; Holland, 1992; Eiben & Smith, 2015; Wierstra et al., 2008; Salimans et al., 2017). RL methods can also use *parameter-perturbations* to improve on the traditional action-perturbations of epsilon-greedy (Plappert et al., 2018b; Fortunato et al., 2018).

Exploration mechanisms can control the selection of *goals* to explore. IMGEP methods, for example, rely on competence-based IMs to define agents that sample their own goals as a way to explore their environment—see a review in Section 1.4. The RL-based implementations of IMGEPs we propose in the present research (RL-IMGEPs) also conduct an *autotelic exploration*.

Although less frequent, exploration mechanisms can also exert other types of control on their experience by selecting social partners to interact with (Nguyen & Oudeyer, 2012) or learning environments (Risi & Togelius, 2019). More generally, automatic curriculum learning techniques—the set of methods that control the presentation of learning experiences as a function of the learner's abilities—can also be understood as exploration methods (Portelas et al., 2020b).

How to Explore? How to Sample Behaviors?

Now that we discussed the type of control exerted by exploration methods (actions, parameters, goals), we need to discuss how exploration methods sample these choice spaces. Sampling methods can be either directed or undirected and either temporally structured or temporally unstructured.

Undirected exploration methods are not updated by the learning experience of the agent but use a fixed distribution to sample from the choice space. Epsilon-greedy methods, for example, can sample uniformly from the action space (e.g. Mnih et al., 2015) or sample action perturbations from a fixed distribution (e.g. Lillicrap et al., 2016). Mutations in genetic algorithms (GA) are often undirected and apply a fixed noise to policy parameters (Holland, 1992; Eiben & Smith, 2015). Undirected selection of actions or parameters often falls short in hard exploration tasks that require a consistent time-extended exploration (Mnih et al., 2015; Bellemare et al., 2016). On the other hand, random goal selection is

often a hard baseline to beat (see examples in Chapter 3, Benureau & Oudeyer, 2016; Forestier & Oudeyer, 2016b).

The exploration problem is ill-posed, as one cannot know now which decision will lead to the collection of the most valuable data for later yet-unknown tasks. The diversity of exploration mechanisms thus result from the definition of various proximal objectives, usually involving forms of knowledge-based or competence-based IMs: high learning progress, surprise, uncertainty, novelty, etc.—see Section 1.3 for examples and references or Linke et al. (2019); Aubret et al. (2019) for detailed reviews.

Exploration methods can lead to temporally structured or temporally unstructured explorations. *Temporally-unstructured* describes a myopic exploration where each exploratory action is taken independently from others (e.g. epsilon-greedy). In contrast, temporally-structured exploration organizes coherent sequences of exploratory actions. Parameter-based and goal-directed explorations are both temporally structured by design because they sample the parameter or goal that conditions the structured generation of actions for the full episode. Exploratory actions undertaken by exploration policies are also temporally structured (e.g. exploration policies trained with KB-IMs). Although epsilon-greedy is unstructured, a quick and efficient fix is to keep the sampled action perturbation fixed for several consecutive timesteps (Raffin & Stulp, 2020).

Curiosity-Based Exploration for Autonomous Skill Acquisition

The knowledge-based and competence-based intrinsic motivations defined in Section 1.3 can either lead to safe behaviors (e.g. drives towards low surprise, high familiarity, high competence) or exploratory behaviors (e.g. high surprise, high novelty, high learning progress). This last category of motivations powers *curiosity-based exploration*: a temporally structured, directed exploration that pushes agents to experience *interesting* situations thought to benefit further learning. Temporally structured, directed exploration can actively organize the time-extended, consistent exploration necessary to solve hard exploration problems characterized by sparse or deceptive reward functions.

Let us now take a step back and consider our primary objective—the autonomous acquisition of skill repertoires. Skills are defined as the association between a goal representation and a policy to reach that goal. In autotelic exploration, agents represent goals and learn associated skills as they explore. On the other hand, action-based and parameter-based exploration methods do not need goal representations and thus do not learn skills. Among autotelic exploration methods, we will focus on those that explore outcome spaces. Although state and outcome space explorations can both be used to build repertoires of skills (state-based skills or outcome-based skills), skills targeting outcomes might offer a more abstract set of skills. We might not care about an agent able to reach any specific joint and visual input configurations (state-skills) but be more interested in agents able to reach any location in a maze (outcome-skills).

Exploration–Exploitation: Coupled vs Decoupled

In the previous sections, we looked at a typology of exploration mechanisms: what to explore and how to explore. Although learning agents can explore freely, they often need to solve extrinsic tasks sometimes—i.e. they need to *exploit*. This raises a question: how should we trade-off exploration and exploitation? There are two main approaches, *coupling* or *decoupling*.

In *coupled* strategies, the agent tries to explore *and* exploit at the same time. Adding noise (exploration) on top of continuous actions selected by a task-oriented policy (exploitation) does just that. Parameter-perturbation methods also explore (adding noise on policy weights) *and* exploit (the original policy is task-oriented). The *bootstrapped* DQN method based on an ensemble of Q-networks also explores (follows a random Q-network) *and* exploits (all Q-networks are trained to approximate the expected value, Osband et al., 2016). Another way to couple exploration and exploitation is to combine the external task reward (r_e) and the exploratory reward (r_i)—also called *exploration bonus*—into an aggregated reward: $r = \beta\, r_e + (1 - \beta)\, r_i$ (e.g. Pathak et al., 2017; Burda et al., 2019; Bellemare et al., 2016; Badia et al., 2020b).

In *decoupled* strategies, the agent either explores or exploits in a given learning episode. When it exploits, its behavior is entirely oriented towards the external reward ($r = r_e$). When it explores, its behavior is entirely oriented towards the intrinsic exploratory reward ($r = r_i$) when there is one or can be unguided (e.g. random controller parameters, random actions). Although most RL methods couple exploration and exploitation, there are a few exceptions: some use two distinct policies and follow either one or the other while training both policies on a shared replay buffer (Zhang et al., 2019; Beyer et al., 2019). Another way to decouple exploration and exploitation is to train a single policy to maximize a parameterized mixture of extrinsic and intrinsic rewards (Badia et al., 2020a,b). Extreme values of the mixing weight perfectly decouple exploration from exploitation.

In the coupled scenario, exploration is constrained around the current optimal policy. Because this limits the scope of the exploration, it can be a very efficient bias when the task requires little exploration. However, this might be a problem in strongly *deceptive tasks* when the exploratory gradient opposes the performance gradient. Decoupling, on the other hand, offers a freer exploration unconstrained by task specifications. This can be an advantage in strongly deceptive environments or a burden in simple ones.

Summary

This section presented a typology of exploration mechanisms and looked at how to combine exploration and exploitation in practice. These mechanisms help overcome the lack of exploratory abilities in standard RL approaches. After reading these lines, you might be convinced that efficient exploration mechanisms need to be *temporally structured*, *directed* and, in environments that require extensive exploration, *decoupled*. IMGEPs, and RL-IMGEPs in particular, organize such exploration by targeting self-generated goals.

1.7 Chapter Summary

This chapter covered the fundamental concepts and literature underlying the design of autotelic agents facing the problem of the autonomous acquisition of open-ended repertoires of skills. We first introduced RL and goal-conditioned RL, two mathematical and algorithmic frameworks to train agents to learn either one or several pre-defined skills, respectively (Sections 1.1 and 1.2). Because these agents lacked autonomy, we introduced the concept of intrinsic motivations developed in psychology, cognitive science and developmental robotics (Section 1.3). Intrinsically motivated agents are autonomous, but only agents endowed with *competence-based* intrinsic motivations can target multiple self-generated goals and thus develop repertoires of skills. We called such agents *autotelic* and introduced a computational framework from developmental robotics called *intrinsically motivated goal exploration process* (IMGEP) to train them (Section 1.4). After pointing the limitations of current IMGEP implementations, we turned back to RL. With our new computational framework, RL-based IMGEPs (RL-IMGEPs), we propose to overcome some of the limitations of existing POP-IMGEP with modern DRL methods (Section 1.5). RL-IMGEPs define autotelic RL agents that learn to represent, generate, pursue and master their own goals and, doing so, build repertoires of skills. Finally, we discussed how the methods presented in the previous sections tackle the exploration problem (Sections 1.6).

The next two chapters present empirical contributions. In Chapter 2, we leverage population-based autotelic methods to conduct exploration with the objective of facilitating the resolution of external pre-defined tasks. We show that the directed, temporally structured and decoupled exploration conducted by our autotelic algorithms facilitates the resolution of hard exploration problems. In Chapter 3, we move away from external tasks and define a purely autotelic RL agent called CURIOUS. Thanks to a new modular architecture and an intrinsic motivation towards learning progress, CURIOUS can efficiently organize its learning trajectory, develop multiple skills and efficiently transfer knowledge between them.

Chapter 2

Autotelic Exploration for External Tasks

This first experimental chapter focuses on the standard RL problem: a learning agent needs to learn how to solve a pre-defined task using learning signals generated by an external, hand-defined reward function. As discussed in our section on exploration (Section 1.6), hand-defining reward functions sometimes—maybe often—leads to the definition of tasks characterized by either deceptive or sparse rewards, sometimes both. To solve them, agents need to explore. In our typology of exploration mechanisms, we concluded that efficient exploration methods need to be temporally structured and directed. In problems that require extensive exploration, decoupling it from exploitation might be necessary.

One way to organize such exploration is to leverage *curiosity-based intrinsic motivations*—KB-IMs or CB-IMs driving agents to explore. In curiosity-based KB-IMs, agents build distributive or predictive models of the world and reward the agent for increasing the entropy of the distribution (Bellemare et al., 2016; Tang et al., 2017) or improving their predictive capacities (Schmidhuber, 1991a; Pathak et al., 2017; Burda et al., 2019). In both cases, KB-IMs drive agents to explore the state space: *to predict the world better, I need to experience every part of it.*

KB-IMs agents pursue a single, slowly-evolving reward function—e.g. the state novelty—but do not represent goals, thus do not learn controllable skills. Curiosity-based CB-IMs, on the other hand, explicitly motivate agents to generate and pursue their own goals, i.e. to practice their own skills. CB-IMs methods usually require the definition of a *behavioral space* or *outcome space* in which compact encodings can characterize relevant aspects of states or trajectories. As they target goals, CB-IMS agents conduct a directed and temporally-structured exploration of the *outcome space* but also grow a repertoire of skills autonomously as a by-product.

This chapter demonstrates that this directed and temporally-structured autotelic exploration helps the resolution of external tasks. In the first section, we conduct autotelic exploration with a basic implementation of a population-based IMGEP (POP-IMGEP), then use the collected experience to bootstrap a standard task-oriented DRL algorithm (Section 2.1). In the second section, we decouple exploration and exploitation in a population of policies trained by evolution strategies targeting either diversity (exploration) or quality (exploitation, Section 2.2).

2.1 Bootstrapping Deep RL with Autotelic Exploration

This section asks whether the acquisition of skill repertoires can be used as an auxiliary task to organize exploration in the context of an external pre-defined task. To answer this question, we introduce a simple algorithm that sequentially runs an autotelic exploration method from the POP-IMGEP family and a standard DRL method trained to solve the external task. The DRL algorithm benefits from the exploration phase via a simple mechanism—we pre-fill its replay buffer with transitions collected during the exploratory phase. The set of skills acquired during the exploration phase is a positive by-product. We call the resulting algorithm GEP-PG for *Goal Exploration Process–Policy Gradient*. We evaluate it on two standard benchmarks *Continuous Mountain Car* and *Half-Cheetah* (Section 2.1.1). We first look at the impact of various undirected exploration methods on performance, then compare these to our autotelic exploration.

2.1.1 External Tasks

We use two standard continuous control benchmarks as external tasks: *Continuous Mountain Car* (CMC) and *Half-Cheetah* (HC).

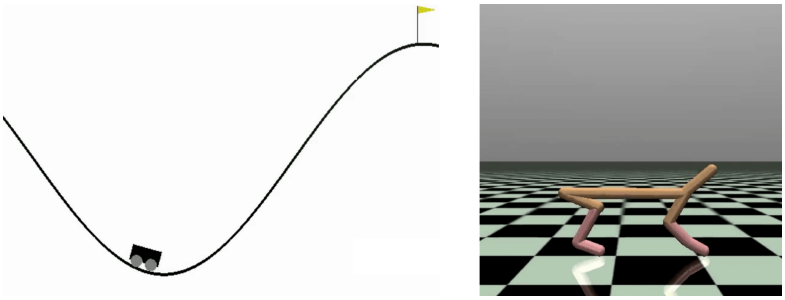


Figure 2.1: Left: Continuous Mountain Car (CMC). The agent learns to reach the flag at the top of the hill. Right: Half-Cheetah (HC). The agent learns to run forward (to the right) as fast as possible.

Continuous Mountain Car

Mountain Car is a standard benchmark for the continuous state, discrete action RL setting. An underpowered car whose actions are in $\{-1, 0, 1\}$ must reach a flag at the top of a hill by gaining momentum from another hill. The state space contains the horizontal position and velocity, and the action is a scalar acceleration. In the standard setting, the objective is to reach the flag as fast as possible. The so-called *bang-bang* policy—using only $\{-1, 1\}$ as actions—is the optimal solution. The environment is reset after 1000 time steps or when the agent reaches the flag.

In the Continuous Mountain Car (CMC) environment, the continuous action is defined in $[-1, 1]$, and the car should reach the flag while spending as little energy as possible (see Figure 2.1). To ensure this, the reward function generates a reward of 100 for reaching the flag and, at each step, an energy penalty for using acceleration a: penalty $= -0.1 \times a^2$.

This reward function raises two exploration issues at once—it is both *sparse* and *deceptive*. The reward function is *sparse* because the large reward (+100) is only received when the flag is reached, which might be difficult to achieve with an unstructured exploration method such as random actions. The reward function is also *deceptive*: before the agent reaches the flag, the gradient of performance pushes the agent to minimize the energy cost—i.e. not to perform any acceleration. In contrast, receiving the large reward requires gaining momentum against the left hill, thus requires acceleration. This problem appears very simple but embodies two exploration issues at once.

Half-Cheetah

Our second benchmark is MUJOCO's Half-Cheetah (HC), where the agent is a 2D bipedal robot made of 8 rigid, articulated links—see Figure 2.1 (Todorov et al., 2012). The agent observes the angular positions and velocities of its joints, its Euclidean position and velocity (17D) and can act on the torques of its legs joints (6D). The reward r sums the instantaneous velocity v_x on the x axis and an energy cost on a: $r = v_x - 0.1 \times |a|^2$. The environment is reset after 1000 steps. HC is not known as a hard exploration problem and DRL algorithms equipped with crude exploration mechanisms achieve great results (e.g. Lillicrap et al., 2016).

2.1.2 Goal Exploration Process – Policy Gradient (GEP-PG)

In this section, we introduce the *Goal Exploration Process – Policy Gradient* algorithm (GEP-PG). GEP-PG combines a temporally structured autotelic exploration phase with a more traditional exploitation phase. Let us look at these independent phases first, then how they are combined.

Autotelic Exploration Phase

The first phase is an autotelic exploration phase implemented by a population-based implementation of IMGEP (POP-IMGEP). By GEP, we refer to a specific implementation of POP-IMGEPs 1) that relies on a genetic algorithm for optimization (GA) and 2) that samples goals uniformly from the outcome space (IMGEPs, see Section 1.4).

In GEP, we consider two spaces: the policy parameter space Θ and the outcome space $\mathcal{O}$. The experimenter defines the outcome representation function that maps trajectories to low-dimensional embeddings characterizing them. For instance, the final x-y position of an agent in a maze can be the outcome that characterizes its whole trajectory. The outcome space is defined as the image space of the trajectory space via the outcome representation function. Given a set of parameters θ, one can evaluate the corresponding policy in the environment over a roll-out and compute the outcome o from the trajectory to obtain a $\langle \theta, o \rangle$ pair, see Figure 2.2.

GEP learns from pairs of policies and corresponding outcomes $\langle \theta, o \rangle$. The general process can be described as follows. In a first *bootstrap* stage, GEP draws N policies θ from Θ and observes, for each policy, the resulting outcome o in $\mathcal{O}$. This stage implements a

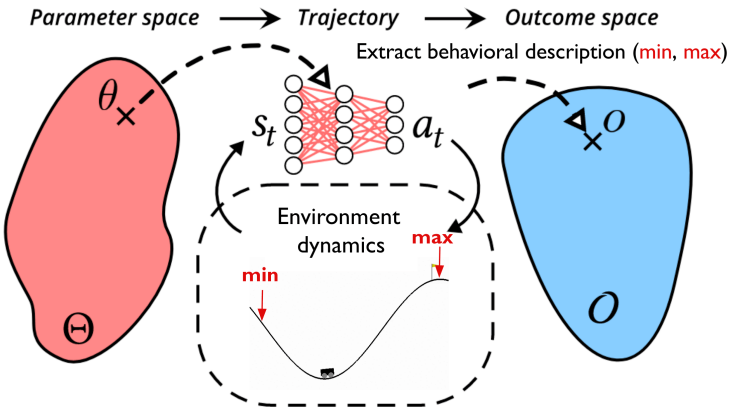


Figure 2.2: Each policy θ can be evaluated in the environment (here CMC). To each policy θ corresponds an outcome o in the outcome space $\mathcal{O}$. Here, o is defined as the minimum and maximal values on the x-axis over the trajectory.

pure random policy search. In the second stage, GEP starts exploring the outcome space. It uniformly draws outcome vectors o from $\mathcal{O}$ and uses them as *goals*. To reach these goals, GEP must find suitable candidate policies. We use a simple strategy here. For each goal, GEP finds the $\langle \theta, o \rangle$ pair in memory for which the outcome is the nearest neighbor to the goal. In an attempt to get closer to the goal, GEP applies a random perturbation to the policy parameters, a simple Gaussian noise $\mathcal{N}(0, \sigma^2 I)$. This strategy corresponds to a simple implementation of a genetic algorithm (GA). More advanced interpolation-based techniques could be used (e.g. Baranes & Oudeyer, 2013).

One essential aspect of this algorithm is that whether the random perturbation actually helps the policy get closer to the goal does not matter; the algorithm learns something anyway. Indeed, it just found the perfect policy to reach one specific goal: the achieved outcome. This is known as *cross-goal learning* or *hindsight learning*—the policy–outcome pair $\langle \theta, o \rangle$ is always a good source of information for some goals, e.g. $g = o$.

By storing more and more $\langle \theta, o \rangle$ pairs, GEP gets better and better at reaching various goals. Note that this is different from *exhaustive search* or *random policy search*; i.e. randomly sampling the parameter space to accumulate knowledge. GEP tries to cover the outcome space uniformly. This leads to an exploration of the parameter space that is *not* uniform. Indeed, we expect that most of the parameter space collapses into small regions of the outcome space corresponding to uninteresting behaviors.

To summarize, GEP organizes exploration via the acquisition of a repertoire of skills, where each skill corresponds to an outcome. This exploration is temporally structured because each trajectory is governed by a unique policy designed to solve a goal. Exploration policies are designed to optimize goal-reaching and are thus *directed*, even though they rely on a simple mutation-based optimization. Finally, exploration is *decoupled* from exploitation, as GEP does not use external reward signals.

Exploitation Phase

We use *deep deterministic policy gradient* (DDPG) to conduct the exploitation phase (Lillicrap et al., 2016; Henderson et al., 2018). DDPG is a DRL algorithm that was—at the time of this study—a state-of-the-art RL algorithm for continuous control (see description in Section 1.1). Here, we consider a standard implementation and a variant. In the standard implementation, DDPG explores by adding an Ornstein-Uhlenbeck correlated noise to the actions (OU); a crude action-perturbation exploration mechanism (Lillicrap et al., 2016). In the variant, DDPG relies on *parameter-perturbation*: it applies noise to policy parameters (Plappert et al., 2018b). Just like in the original paper, a simple mechanism helps to tune the standard deviation of the Gaussian noise applied on policy parameters to keep the standard deviation of the distribution of actions close to a pre-defined level (here $\sigma = 0.2$).

Exploration–Exploitation Trade-Off

The GEP-PG methodology takes a decoupled approach and comes in two steps. In the first autotelic exploration stage, we train GEP to collect a diversity of samples using either a linear policy or the same policy as DDPG. In a second *exploitation* stage, we pre-load DDPG's replay buffer with GEP-generated samples and start DDPG on the external task. Note that the transfer of transitions is the only connection between both phases. In particular, DDPG starts learning from randomly-initialized actor and critic networks. We expect the diverse data collected by GEP to help DDPG overcome exploration issues, learn faster and achieve higher performance.

Experimental parameters

For DDPG, we use the default set of parameters provided in Henderson's version (Henderson et al., 2018): a 2M-transition replay buffer, batch size of 64, discount factor of 0.99. The actor and critic networks use Xavier's initialization (He et al., 2015), two hidden layers of size (64, 64) and ReLU activation functions. The output layer activation functions are *tanh* and linear, respectively. We use an ADAM optimizer with learning rates of 10^{-4} and 10^{-3} (Kingma & Ba, 2015).

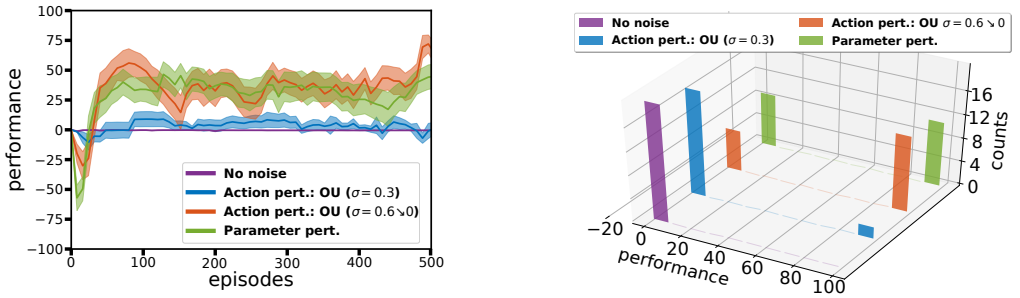
GEP policies are either linear with *tanh* activations or have two hidden layers of size (64, 64), ReLU activations in the hidden layers and *tanh* activations at the output. In a bootstrap stage, we sample GEP policy parameters in $[-1, 1]^P$ where P is the number of parameters. We use 10 episodes of bootstrap for CMC, 50 for HC. Then 90 (respectively 450) random goals are drawn in a custom goal space which is 3D for CMC (range of positions, maximum position and total energy spent) and 2D for HC (mean velocity and minimum head position). The goal spaces are normalized by hand to $[-1, 1]^3$ and $[-1, 1]^2$ respectively. Given a goal, the algorithm finds in its memory the closest previously-experienced outcome using a 1-nearest neighbor algorithm (Euclidean distance) and samples a Gaussian noise to act as policy mutation with $\sigma = 0.01$. Further details and the full implementation can be found at https://github.com/flowersteam/geppg.

2.1.3 Is Undirected Exploration Enough?

Before looking at the impact of autotelic exploration on performance, let us look at the impact of undirected exploration methods. Is it true that parameter-perturbations work better than action-perturbations, as argued in Plappert et al. (2018b)?

In CMC and HC, we compare four variants of DDPG characterized by different undirected exploration mechanisms: 1) a DDPG without any exploration; 2) the standard DDPG from the original paper (Lillicrap et al., 2016) with an OU noise ($\mu = 0$, $\sigma = 0.3$); 3) a variant with a linearly decreasing OU noise ($\sigma = 0.6 \searrow 0$); 4) the parameter-perturbation variant from Plappert et al. (2018b).

During training, we regularly evaluate learning agents offline by reporting their average return computed over 20 episodes without exploration noise. The global performance is measured as the average return over 100 episodes of the best policy found across learning. Whereas standard methodologies usually report the performance of the last policy, we found that learning could be quite unstable in CMC and rather use the best policy. We run each algorithm 20 times with 20 random seeds and comment on comparisons using Welch's t-test with significance level $\alpha = 0.05$. The null hypothesis assumes equal means of the distributions of performance for the two algorithms being compared. We report the mean and standard error of the mean. For further details on methodology, readers can refer to Appendix A.

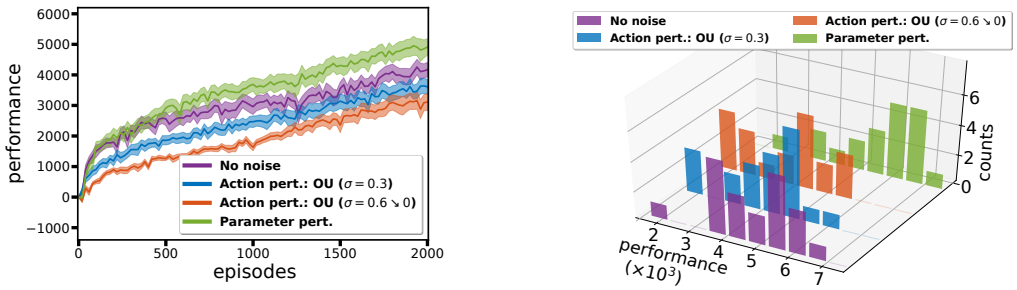


(a) (b)

Figure 2.3: Effect of various undirected exploration methods in CMC (a): learning curves. (b): histogram of global performance measures.

Consistent with Plappert et al. (2018b), we found that parameter-perturbation outperforms action-perturbation in HC and CMC ($p < 6.0 \times 10^{-4}$), see Figure 2.3 and 2.4. In CMC, DDPG without noise performs poorly and DDPG with noise annihilation outperforms the static noise version, see Figure 2.3. In HC, it seems that having no noise, a standard OU(0.3) or a decreasing noise do not lead to statistically different performance measures at 2M steps, although differences could be noted earlier on, see Figure 2.4. We conclude that increasing noise in the first stage is beneficial in CMC but might be detrimental in HC.

Several phenomena might explain these contrasting results. First, there is probably less need for exploration in HC than in CMC, as reward information is available anytime (dense reward). Second, as explained in Section 2.1.1, reaching the goal early is crucial



(a) (b)

Figure 2.4: Effect of various undirected exploration methods in HC (a): learning curves. (b): histogram of global performance measures.

in CMC—not in HC. Using a more substantial exploration noise in the first stage might therefore bring a significant advantage in CMC. Third, it might be the case that too much noise is more detrimental to performance in HC than in CMC because, in HC, more noise may result in more falls.

However, the information depicted in Figure 2.3a must be taken with a grain of salt. Indeed, the curves depict averages over 20 runs, but the individual performance measures are not normally distributed. To present complementary evaluations, we choose to report the histogram of global performance across seeds in Figure 2.3b.

In conclusion, parameter perturbations perform better than action perturbations, whereas increasing exploration early might be either advantageous or detrimental to the performance depending on the environment properties. In CMC, it seems these undirected exploration methods are not sufficient to solve the task reliably. In HC, they were sufficient to achieve state-of-the-art performance when this study was conducted (Plappert et al., 2018b).

2.1.4 Can Autotelic Exploration Help?

Undirected exploration seems insufficient in CMC. Can we do better with autotelic exploration? Undirected exploration seems to be sufficient in HC. Can we do even better with autotelic exploration? Before answering these questions, let us first conduct a quick sanity check. In the small world of CMC, does autotelic exploration explore better than DDPG—i.e. does it reach the flag earlier and more often?

Reaching the Flag

As explained above, the CMC benchmark raises a specific exploration issue because the time at which the agent first reaches the flag is crucial. If DDPG does not find the flag early, it quickly learns to avoid energy costs and stops moving, ruining any hope to reach it.

Let us look at the number of steps required before the agent reaches the flag for the first time. We compare four algorithms: 1) GEP applied to the linear policy we use in

GEP-PG; 2) GEP applied to the 'deep' policy (64, 64); 3) DDPG with parameter noise and 4) DDPG with action noise OU($\sigma = 0.3$). For each condition, we perform 500 trials and record the mean and the histogram of the number of steps necessary to reach the flag—if it happens in the first 50.000 steps. We present the results in Figure 2.5.

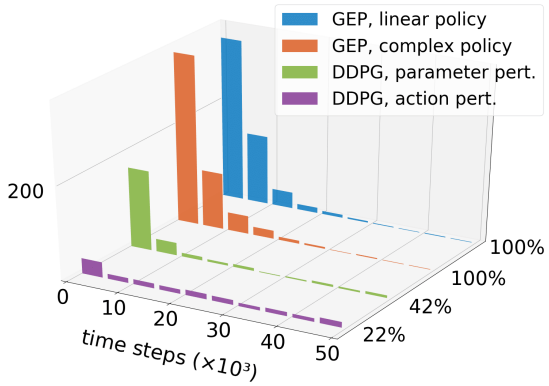


Figure 2.5: Histogram of the number of steps required to reach the flag for the first time with 1) GEP applied to a linear policy; 2) GEP applied to a deep policy; 3) DDPG with parameter noise and 4) DDPG with action noise OU($\sigma = 0.3$). The percentages of times the goal was found before 5×10^4 steps are indicated on the y-axis.

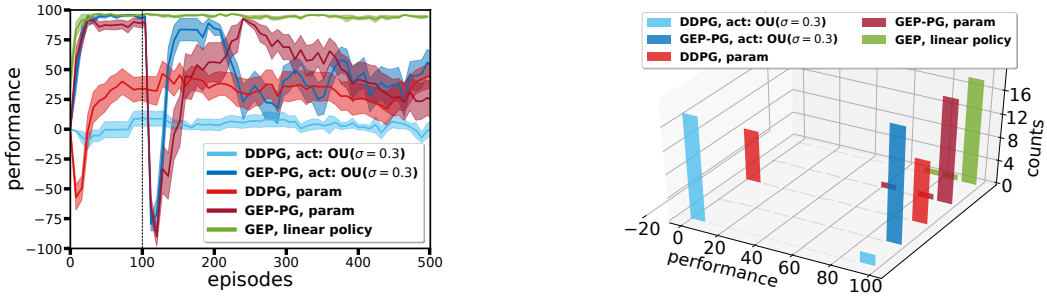
Linear and deep versions of GEP reach the flag after an average of 3875 and 3773 steps, respectively. DDPG with parameter perturbations (respectively action perturbations) reaches the flag before 50.000 steps in only 42% (respectively 22%) of the runs. Thus, GEP finds the goal faster than the DDPG variants. This effect can be attributed to the deceptive gradient issue: before experiencing positive rewards, RL updates drive the policy search to a local optimum in which the agent stops moving. We also tested a random action baseline and found it reached the flag in the first 50.000 steps only 6% of the time. The complexity of the GEP policy does not seem to impact exploration.

In fact, GEP variants reach the goal as early as the bootstrap phase, where the policy is essentially random. Random policy search, because it avoids the deceptive gradient, is thus enough to solve CMC. This confirms the idea that CMC can be considered a simple problem when the algorithm is not blinded by deceptive rewards.

GEP–PG

Let us now answer the central question: can autotelic exploration help RL solve external tasks? To answer this question, we compare the performance of DDPG with action or parameter perturbation to their corresponding GEP-PG variants and GEP alone. The vertical dotted lines in Figure 2.7a and 2.6a signal the transition from the exploration phase to the exploitation phase. For GEP-PG, we report the performance of the best policy found by GEP before the dotted line and report the performance of the current DDPG policy after the line. In both cases, performance measures are evaluated offline by testing the best policy over 20 episodes.

In CMC, GEP alone performs best. CMC is a simple task when the deceptive reward

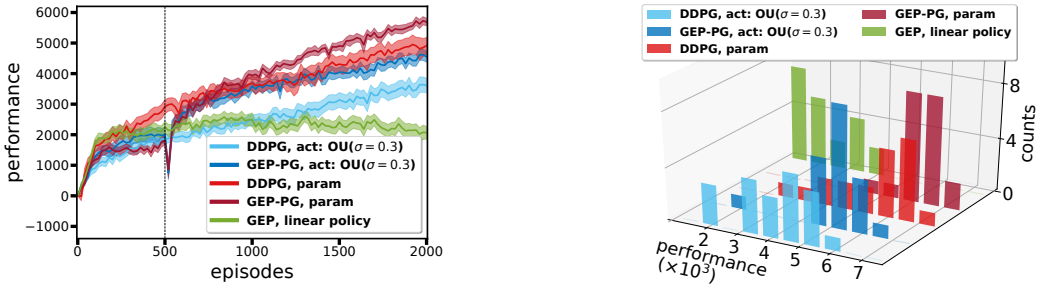


(a) (b)

Figure 2.6: (a): Learning performance in CMC of GEP, DDPG with action noise OU($\sigma = 0.3$), DDPG with parameter noise and GEP-PG variants with the same noise configurations. (b): Histogram of global performance measures in CMC.

is not considered. As we saw before, DDPG alone cannot reliably solve the task, no matter its exploration setting. Looking at GEP-PG's performance curves, we can see that performance drops when switching from GEP to DDPG. This is expected because GEP-PG only transfers collected transitions from GEP to DDPG; not the policies themselves.

After this drop, GEP-PG policies quickly learn reliable policies solving the task. However, learning seems to be highly unstable and this performance quickly drops to performance measures similar to the ones achieved by DDPG. As before, the learning performance depicted in Figure 2.6a does reflect the ratio of trials that reliably find the goal rather than the average performance. The distribution of performance measures can be better appreciated in the histogram, Figure 2.6b.



(a) (b)

Figure 2.7: (a): Learning performance in HC of GEP, DDPG with OU noise ($\mu = 0, \sigma = 0.3$) on actions, DDPG with noise on policy parameters and GEP-PG with the same noise configurations. (b): Histogram of global performance measures in HC.

In HC, GEP alone performs statistically worse than others ($p < 10^{-5}$). We found that the deep policy did not perform better ($p = 0.7$). Whereas GEP was efficient enough to master CMC, it does not seem to scale to the higher dimensional HC. More importantly, the GEP-PG versions significantly outperform their DDPG counterparts on this benchmark ($p = 0.0046$ and $p = 0.023$ for action and parameter perturbations respectively). At the time of this study, this made GEP-PG the new state-of-the-art on Half-Cheetah—though

not for long (Haarnoja et al., 2018b; Fujimoto et al., 2018). Average global performance metrics are reported in Table 2.1. Note that we do not represent the learning curves of DDPG without exploration noise and its GEP-PG counterpart. They were found to match the corresponding versions using OU(0.3) noise closely.

Table 2.1: Final and global performance measures in HC, mean ± std.

Algorithm	final performance	global performance
DDPG action pert.	3596 ± 1102	3943 ± 1209
DDPG param. pert.	4961 ± 1288	5445 ± 1265
GEP-PG action pert.	4521 ± 738	5033 ± 833
GEP-PG param pert.	**5740 ± 573**	**6118 ± 634**
GEP param pert.	2105 ± 881	2140 ± 881

We found GEP-PG to be robust to the number of GEP episodes used to fill the DDPG replay buffer. The performance remains stable from 100 to 1000 GEP episodes. Finally, the GEP-PG versions seem to generate less variability than their DDPG counterparts (see standard deviations in Table 2.1, statistically significant for the parameter perturbation comparison with Levene and Bartlett tests). This means that an efficient policy can be found more consistently and can matter in risk-sensitive applications were performance should not fall below a specific level.

Looking at the Buffer's Content

There is still one thing we need to check. It might be that the performance boost is not due to the quality of the exploration but to the pre-loading of DDPG's replay buffer. To answer this question, we compared several ways to fill DDPG's buffer: 1) empty buffer; 2) buffer filled with GEP; 3) buffer filled by running an untrained DDPG policy with parameter noise exploration; 4) buffer filled by policies sampled uniformly from the parameter space Θ. Figure 2.8 presents the results. Of all the three methods, only GEP-PG significantly outperforms the DDPG baseline. However, it seems that the variance of the performance decreases for all pre-filing strategies.

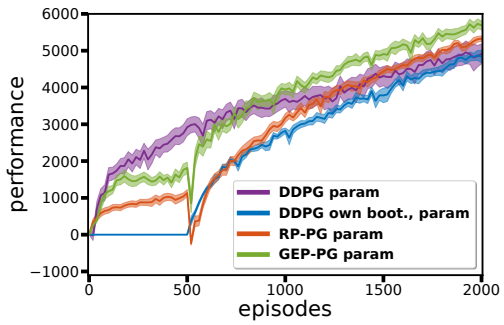


Figure 2.8: Comparing different methods to pre-fill the replay buffer in GEP-PG.

Further analyses of the impact of the content of a buffer filled with GEP on GEP-PG performance in HC led to the following conclusions: 1) the method is robust to variations of the number of GEP transitions transferred to DDPG in the range 100–1000; 2) GEP-PG's performance positively correlates with GEP best performance ($p < 10^{-5}$) and average performance ($p < 10^{-7}$); 3) GEP-PG's performance positively correlates to diversity metrics. In particular, it correlates to:

- the standard deviation of the performance of GEP policies (performance diversity, $p < 10^{-9}$);
- the average standard deviation of features distributions (perceptual diversity, $p < 10^{-7}$);
- the average distance between outcomes and their k nearest neighbors in outcome space, for various k (outcome diversity, $p < 10^{-9}$);
- the entropy of outcome distributions for various discretization step sizes (outcome diversity, $p < 10^{-6}$).

Overall, these results indicate that the quality of the autotelic exploration phase is correlated to increased performance down the line in the exploitation phase.

Repertoires of Skills as a Side Effect

During exploration, GEP fills an archive of skills as $\langle \theta, o \rangle$ pairs. Skills can easily be retrieved by selecting a goal in the outcome space and retrieving the associated policy with the nearest neighbor strategy or from a more sophisticated inverse model trained locally from the memory. The set of skills is conditioned on the outcome representation function. In HC, we consider a 2D space with the average velocity and the minimum head position. This results in skills like running forward, running backward, not moving, or flipping on the back. In CMC, the 3D outcome space is (range of positions, rightest position and total energy spent). This results in different skills such as not moving, covering the widest range with bang-bang policies, covering a short range with large accelerations, etc. These skills are not used in GEP-PG but are interesting for our main objective: the autonomous acquisition of behavioral repertoires.

2.1.5 Discussion

In this study, we asked whether organizing exploration as the acquisition of behavioral repertoires could benefit learning in an external pre-defined task. To answer this question, we presented GEP-PG, a simple algorithm that sequentially combines an autotelic exploration phase and a traditional exploitation phase with DDPG. We showed that seemingly simple problems like CMC could hide both deceptive and sparse reward functions and, for this reason, be difficult to solve with traditional DRL algorithms. Indeed, such algorithms usually implement crude exploration mechanisms always coupled with exploitation, and can be temporally unstructured (e.g. action perturbations) or temporally structured but undirected (e.g. parameter perturbation). We showed that these exploration mechanisms

do not allow efficient exploration in the CMC benchmark and proved that replacing them with a decoupled, temporally structured and goal-directed mechanism such as GEP removes the difficulty. GEP-PG proposes a simple idea to generate efficient exploration in these cases: leveraging an efficient exploration phase decoupled from exploitation. We showed this approach helped in the deceptive and sparse CMC benchmark and in HC, even though the latter is not considered to be a hard-exploration problem. Let us now discuss the limitations of this work and give hints about how they could be overcome. This discussion is re-written three years after the study, it includes references to posterior works.

The proposed implementation of GEP-PG currently uses one of the simplest mechanisms compatible with the general method of decoupling a pure exploration phase with an exploitation phase. GEP stores past policies, thus is less sensitive to the *catastrophic forgetting* issues encountered by sequential learning in goal-conditioned policies based on neural networks. The counterpart is that each policy might just be "lucky" to lead to a certain outcome as it is not trained to be robust to a variety of initial conditions or to stochastic dynamics. In CMC and HC, however, the effect is minimal because the dynamics are deterministic and the distribution of initial states has low variance. Our simple GEP implementation also seems to show limited performance in HC, which is probably explained by our crude mutation mechanism.

GA-based optimization methods can benefit from more sophisticated mutation operators. The NEAT algorithm, for example, uses GA to evolve the architecture of the network as well as its weights using advanced mutation operators and diversity-preserving techniques (Stanley & Miikkulainen, 2002). It is widely used in NS implementations (Lehman & Stanley, 2011a). GA can also be scaled to evolve deeper neural networks, although this does not seem to work well in high-dimensional control tasks (Such et al., 2017). *Safe mutations through gradients* can facilitate the training of high-dimensional policies by automatically tuning parameter-specific mutation strengths to maintain a pre-defined level of divergence in the network's outputs (Lehman et al., 2018b). In the context of IMGEPs, previous works have also investigated more efficient methods to build inverse models ($\mathcal{O} \rightarrow \Theta$), see Forestier et al. (2017).

Another limitation of our GEP-PG implementation is the simplicity of the transfer mechanism from the exploration to the exploitation phase. To replace the pre-loading of the replay buffer, we could imagine using *imitation learning* methods: to train DDPG's actor to reproduce the behavior of GEP's policies. One could also pre-train both the actor and the critic using methods from *off-line* RL, see a review in Levine et al. (2020). Finally, our implementation involves a sequential combination of exploration and exploitation. We could imagine more than two phases, where a controller—e.g. a two-arm bandit algorithm—would decide whether to explore or to exploit based on high-level objectives such as learning progress (Lopes & Oudeyer, 2012).

One striking limitation of GEP-PG seems to be its potential for instability, as revealed in the CMC benchmark. We hypothesize it might be due to the crude transfer of transitions from exploration to exploitation. Indeed, DDPG and other so-called *off-policy* algorithms are known not to be *truly* off-policy and can have trouble handling data coming from drastically different policies (Levine et al., 2020). In recent years, many approaches have been developed under the name of *off-line* RL: methods that aim to learn from datasets

of diverse experience collected with different policies. We think GEP-PG could benefit from these new techniques and that it would reduce its instabilities.

Evolutionary and developmental exploration methods like GEP, NS, or QD require the pre-definition of an outcome representation characterizing interesting features of the state or state-action trajectories. In some environments, defining these might be straightforward (e.g. final x-y position in a maze), but it might be non-trivial in others (e.g. in the Hopper benchmark). How to automatically learn useful outcome representations without explicit supervision remains an open question. Existing approaches often train use VAE (Kingma & Welling, 2014) to learn low-dimensional representations of high-dimensional states such as images, but are often limited to simple setups (e.g. with few objects Péré et al., 2018; Laversanne-Finot et al., 2018; Nair et al., 2018b; Cully, 2019).

Finally, let us take a step back. This study was conducted in late 2017 and both the field of RL and myself have considerably learned since then. One thing that is perhaps missing in this study is an explicit comparison to exploration methods relying on *knowledge-based intrinsic motivations*, a few of which already existed at the time (Bellemare et al., 2016; Pathak et al., 2017). KB-IMs offer temporally structured and directed exploration but are often used in a coupled way with exploitation as an *exploration bonus*. However, they could easily replace the autotelic exploration of GEP in GEP-PG to perform a decoupled exploration. This would have been an interesting baseline to compare to, although KB-IMs are not concerned with the acquisition of skill repertoires—which is an important feature of this research. The DIAYN approach might also be an interesting exploration algorithm to replace GEP in GEP-PG, as it allows the acquisition of skills repertoires without pre-defined outcome spaces (Eysenbach et al., 2019).

The idea of combining evolutionary algorithms and RL to help exploration has come a long way, and many approaches have built on GEP-PG to achieve state-of-the-art results (Pourchot & Sigaud, 2019; Maheswaranathan et al., 2019; Khadka et al., 2019; Majumdar et al., 2020; Jung et al., 2020; Shi et al., 2020; Doan et al., 2019). In evolutionary RL, algorithms train a population of policies with evolutionary methods, use the generated data to train one or several RL policies and sometimes transfer a copy of RL actors back to the evolutionary population. This symbiosis enables the algorithm to leverage the good exploration properties of evolutionary algorithms—exploration is temporally structured because of parameter-based mutations and is conducted in parallel by a population of policies—and the sample efficiency and fine-tuning abilities of RL. We observe a similar interest in the idea of effective decoupling of exploration and exploitation (Eysenbach et al., 2019; Beyer et al., 2019; Zhang et al., 2019; Ecoffet et al., 2021; Matheron et al., 2020; Cideron et al., 2020a; Badia et al., 2020a,b). In particular, recent progress in the Atari benchmark was associated with such decoupling. Go-Explore, for example, decouples exploration and exploitation in the same trajectory: the agent first goes back to a previously experienced outcome (*first go*), then starts exploring when that outcome is reached (*then explore*) (Ecoffet et al., 2021; Matheron et al., 2020). The first agent to solve all 57 Atari games at super-human level also uses decoupling by training a policy to maximize all mixtures of extrinsic and intrinsic motivations, thus sometimes performing decoupled behaviors (Badia et al., 2020a,b).

2.2 Decoupling Autotelic Exploration and Exploitation with Evolution Strategies

In this study, we look at quality-diversity, a family of algorithms that grow a population of solutions—a set of skills—that is both *behaviorally diverse* and *locally high-performing* in some external task (Cully & Demiris, 2017). As discussed in Section 1.4, QD algorithms—just like population-based IMGEPs—often rely on genetic algorithms (GA) and have trouble scaling to high-dimensional control tasks (Such et al., 2017).

This study proposes to scale a famous QD algorithm called MAP-ELITES (Mouret & Clune, 2015) to solve high-dimensional control tasks using high-dimensional policies. Like in the previous study, we demonstrate how decoupling autotelic exploration and task-oriented exploitation facilitates the resolution of hard-exploration problems. This work was conducted during a summer internship at UberAI Labs in collaboration with Joost Huizinga and Kenneth Stanley, under the supervision of Vashisht Madhavan and Jeff Clune.

2.2.1 Motivations and Related Work

In recent years, neuroevolution has emerged as a powerful competitor to back-propagation for training deep neural networks (DNNs). Salimans and colleagues, in particular, present a scalable version of the evolution strategy algorithm (ES) similar to the one introduced in 1988 by Salomon (Salomon, 1998; Salimans et al., 2017). This scaled version performs comparably to state-of-the-art DRL algorithms on Mujoco's high-dimensional control tasks (Todorov et al., 2012; Brockman et al., 2016) and the Atari suite (Bellemare et al., 2013). In contrast with GAs, ES combines information from many perturbed versions of the parent policy to generate a new one. Recent implementations efficiently leverage resources of computing clusters by parallelizing policy evaluations, making ES algorithms competitive with DRL in terms of training time (Salimans et al., 2017). A similar approach was undertaken to scale GA to deep neural networks (Such et al., 2017). It achieves impressive results in Atari games but fails to efficiently solve continuous control tasks from the Mujoco suite (Todorov et al., 2012).

ES thus emerged as a powerful alternative to GAs. We can leverage these results to scale novelty-search and quality-diversity algorithms. Novelty-search with ES (NS-ES) made the first step in this direction: replacing the performance objective of ES with a novelty objective (Conti et al., 2018). They proposed two variants to incorporate the performance objective: NSR-ES, which mixes performance and novelty objectives evenly and NSRA-ES, which adaptively tunes the ratio between the two. Like most RL exploration mechanisms, these methods are coupled. They use a mixture of exploitation and exploration objectives to deal with the *exploitation-exploration tradeoff*. While this might work when the two objectives are somewhat aligned, it may be inefficient when they are not (Pugh et al., 2016).

Quality-diversity algorithms (QD) present a natural way of *decoupling* the optimization of exploitation (quality) and exploration (diversity) by looking for high-performing solutions in local niches of the behavioral space, leading to local competition between solutions instead of global competition (Lehman & Stanley, 2011b; Mouret & Clune, 2015;

Cully et al., 2015; Cully & Demiris, 2017). In this study, we propose ME-ES, a new QD algorithm based on MAP-ELITES that leverages ES as its underlying algorithm. Unlike NS-ES and its variants, which optimize a small population of DNN policies (5 in Conti et al., 2018), we want to build a large repertoire of diverse and high-performing policies. As we will see, such a repertoire is useful for several reasons: 1) it organizes an efficient exploration to facilitate the resolution of external tasks; 2) it produces a rich repertoire of skills in high-dimensional control problems and 3) it offers the algorithm the ability to adapt to perturbations by searching recovery policies within the behavioral archive.

Table 2.2 presents a classification of ME-ES and the related algorithms presented in this study along three dimensions: 1) whether they use pure exploration, pure exploitation or both; 2) whether they rely on ES-based or GA-based learning algorithms and 3) whether they couple or decouple the trade-off between exploration and exploitation (if applicable).

Table 2.2: Classification of related algorithms. ME-GA is the traditional implementations of MAP-ELITES based on GAs.

	based on ES	based on GA
pure exploitation	ES	GA
pure exploration	NS-ES	NS
exploration & exploitation	coupled: NSR-ES, NSRA-ES	ME-GA
	decoupled: ME-ES	

2.2.2 Background on MAP-Elites and Evolution Strategies

This section provides a bit of background on the original MAP-ELITES and ES algorithms.

MAP-Elites

MAP-ELITES was first introduced in Mouret & Clune (2015). It requires the definition of a *fitness* function (F) to measure the performance of the agent and a behavioral characterization (BC) to map the state-action trajectories to low-dimensional embeddings lying in a *behavioral space*.[1] MAP-ELITES keeps track of an archive of triplets (θ, f, BC) where θ is a set of controller parameters, f is its fitness and BC its behavioral characterization—i.e. its outcome. The objective of MAP-ELITES is to curate a repertoire of behaviorally diverse and locally high-performing policies. To this end, the behavioral space is discretized into *cells* representing *behavioral niches*. In each cell, the algorithm keeps the highest performing policy—highest f—whose BC felt in the niche. Individuals in each niche optimize for the fitness objective (local competition), yet are implicitly driven towards empty and under-optimized cells to avoid the selection pressure of cells filled with highly optimized individuals.

[1] These are equivalent to the outcome o and outcome space $\mathcal{O}$ presented in the previous study, but I keep the usual QD terminology here.

After initializing the archive with a few randomly initialized policies, MAP-ELITES repeats the following steps:

1. Select a populated cell,
2. Mutate the cell's policy with GA to obtain a new policy,
3. Gather one or several trajectories of agent-environment interactions with the new policy and compute its fitness and BC,
4. Update the archive: add the policy to the cell where the BC falls if at least one of two rules is satisfied.
    - *Rule 1*: the cell is empty,
    - *Rule 2*: the fitness is higher than that of the policy currently in the cell.

The two rules guiding additions to the archive implement a decoupling between exploitation (quality) and exploration (diversity). *Rule 1* implements exploration, as it ensures the preservation of policies that exhibit novel behavior (i.e. mapped to empty cells). *Rule 2* implements exploitation, as it enforces local competition and retains only the highest performing solution in each behavioral niche. In this manner, the exploration and exploitation objectives cannot contradict each other. Note that exploration and exploitation are not directly optimized. The undirected mutation of the policy might eventually serve either exploration or exploitation; we can only tell in hindsight. We call ME-GA the traditional implementation of MAP-ELITES based on GA while MAP-ELITES encompasses both ME-GA and ME-ES variants.

Evolution Strategies

Evolution Strategies (ES) is a class of black-box optimization algorithms inspired by natural evolution (Rechenberg, 1973; Back et al., 1991; Salimans & Chen, 2018). At each generation, an initial parameter vector (the *parent*) is mutated to generate a population of parameter vectors (the *offspring*). The *fitness* of each child is evaluated and their parameters are combined to produce the new parent, such that individuals with high fitness have a stronger influence than others. As this process is repeated, the population tends towards regions of the parameter space with higher fitness until a convergence point is reached.

We use a version of ES introduced in Salimans et al. (2017) and similar to Salomon (1998) that belongs to the subcategory of Natural Evolution Strategies (NES) (Wierstra et al., 2008; Sehnke et al., 2010). In NES, the population of parameter vectors θ is represented by a distribution $p_\psi(\theta)$, parameterized by ψ. Given a fitness function F, NES algorithms optimize the expected fitness $\mathbb{E}_{\theta\sim p_\psi} F(\theta)$ using stochastic gradient ascent. Recently, Salimans and colleagues proposed a version of the NES algorithm able to scale to the optimization of high-dimensional parameters ($\approx 10^5$) (Salimans et al., 2017). In their work, they address the RL problem. θ are the parameters of a policy while the fitness is computed from the return obtained by the corresponding policy over an episode of interactions with the environment. Here we use the ranks of the offspring returns as

a measure of fitness. Given the parent policy parameters θ, the offspring population $(\theta_i)_{i\in[\![1,\,n]\!]}$ is sampled from the isotropic multivariate Gaussian distribution $\theta_i \sim \mathcal{N}(\theta,\, \sigma^2 I)$ with fixed variance σ^2. Like in REINFORCE (Williams, 1992), θ is updated according to the following gradient approximation:

$$\nabla_\psi \, \mathbb{E}_{\theta\sim p_\psi}[F(\theta)] \approx \frac{1}{n}\sum_{i=1}^{n} F(\theta_i)\, \nabla_\psi \, log\, p_\psi(\theta),$$

where n is the offspring population size, usually large to compensate for the high variance of this estimate. In practice, any θ_i can be decomposed as $\theta_i = \theta + \sigma\, \epsilon_i$, where ϵ_i is a standard Gaussian noise. The gradient is then estimated by:

$$\nabla_\psi \, \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\,I)}[F(\theta+\sigma\epsilon)] \approx \frac{1}{n\sigma}\sum_{i=1}^{n} F(\theta_i)\, \epsilon_i.$$

Our version of ES—that we simply call ES—is equivalent to the one of Salimans and colleagues (Salimans et al., 2017). Like them, we use a virtual batch normalization of the policy's inputs and rank-normalize the fitness $F(\theta^i)$.

2.2.3 ME-ES: MAP-Elites Powered by Evolution Strategies

We evaluate ME-ES in two setups:

- **Damage Adaptation**: we first build a repertoire of diverse walking gaits for a quadruped robot in a high-dimensional environment, then evaluate its quality in a downstream task called *damage adaptation*. In damage adaptation, the agent undertakes damage (e.g. disabled joints) and must find recovery policies from the behavioral repertoire to succeed. This is similar to the setup introduced in Cully et al. (2015).
- **Deep Exploration**: we use behavioral repertoires to organize exploration in a deceptive maze task where the agent is a multi-joint simulated robot.

The ME-ES algorithm reuses the founding principles of MAP-ELITES underlying damage robustness and efficient exploration while leveraging the optimization performance of ES. ME-ES curates a *behavioral map* (BM), an archive of neural policies parameterized by θ. Every N_g generations, it samples a populated cell and its associated policy θ_{cell} from the archive. It copies this policy to θ_g, which is subsequently evolved for N_g generations using ES. At each generation, offspring parameters $\theta_g^i \sim \mathcal{N}(\theta_g,\, \sigma^2\, I)$ are used to compute an update for θ_g. Just like in NS-ES, we only consider the $\theta_{[g,\,..,g+N_g]}$ for addition to the BM (Conti et al., 2018). Doing so, we ensure robust evaluations of our candidate solutions (θ_g are evaluated over 30 episodes, θ_g^i only in one) and a fair comparison with NS-ES and its variants. Algorithm 1 provides a detailed outline of the way ME-ES combines principles from MAP-ELITES and ES.

Algorithm 1 ME-ES algorithm

Input: number of generations n_gens, population size n, mutation strength σ, number of generation per parent N_g, empty behavioral map BM, number of evaluations n_e.
Initialize: $BM \leftarrow (\theta_0 \sim init_{normc}(), F(\theta_0), BC(\theta_0))$
for $g = [0, \text{n\_gens}]$ **do**
  if g % N_g == 0 **then**
  ▷ Sample a cell and objective
    mode ← explore_or_exploit()
    if mode is 'explore' **then**
      θ_{cell} ← sample_explore_cell()
    else if mode is 'exploit' **then**
      θ_{cell} ← sample_exploit_cell()
    $\theta_g \leftarrow \theta_{cell}$
  ▷ Run ES update
  θ_g ← ES_optim(θ_g, n, σ, objective=mode)
  f, BC ← Evaluate(θ_g, n_e)
  Add (θ_g, f, BC) to BM.

ME-ES Variants

We define three variants of ME-ES that differ in the objectives optimized by ES:

- In ME-ES EXPLOIT, the objective is the fitness function F. Offspring solutions are ranked according to their cumulative reward computed over one episode. The fitness $F(\theta)$ of a solution is then computed as its normalized rank (*directed exploitation*).
- In ME-ES EXPLORE, the objective is the *novelty* function $N(\theta, k)$. We define it as the average Euclidean distance between a policy's BC and the BCs of its k nearest neighbors in an *archive* that contains all previous θ_g, regardless of their addition to the BM (one per generation). Because the novelty objective explicitly incentivizes agents to explore their environment, we call it a *directed exploration* objective.
- Finally, ME-ES EXPLORE-EXPLOIT alternates between both objectives. It implements a decoupled exploration and exploitation procedure. Because it explicitly optimizes for both objectives, ME-ES EXPLORE-EXPLOIT conducts both *directed exploration* and *directed exploitation*.

Note that ME-ES EXPLORE performs undirected exploitation; while the ES steps do not directly optimize for fitness, performance measures are still used to update the BM (Rule 2 of the map updates). In the same way, ME-ES EXPLOIT also performs undirected exploration, as the BM is updated with novel policies (Rule 1 of map updates). Like ME-GA, all versions of ME-ES thus perform forms of exploration and exploitation. However, only ES optimization steps enable agents to perform the *directed* optimization of exploitation and exploration objectives.

Cell Sampling

The number of generations performed by ME-ES is typically orders of magnitude lower than MAP-ELITES because each generation of ME-ES involves a greater number of training episodes (10^4 instead of 1 for ME-GA). Thus, the sampling of the cell from which to initiate the next N_g generations is crucial (see Algorithm 1). Here, we move away from the uniform cell selection of MAP-ELITES towards a biased cell sampling. We propose two distinct strategies.

For exploitation steps, we select from cells with high-fitness policies, under the assumption that high fitness cells may lead to even higher fitness cells. For exploration steps, we select from cells with high novelty scores. By definition, novel policies are in under-explored areas of the search space and mutating these policies should lead to novel areas of the behavior space (Lehman & Stanley, 2011a,b). In practice, as the number of populated cells increases with each generation, the bias towards selecting cells with higher performance or novelty scores diminishes. For instance, if cells are selected in proportions of their novelty, the best policy of two that have novelty 2 and 1 will have a 2/3 chance of being selected, while the same policy in a group of 3 policies with novelties $[2, 1, 1]$ will only have probability 1/2 of being selected. To fight this phenomenon, we propose to restrict the novelty-proportional selection to the five most novel cells. Similarly, for exploitation steps, we sample either uniformly from the two highest fitness cells ($p = 0.5$), which promotes exploitation from the current best cells, or uniformly from the two highest fitness cells from the last five updated ($p = 0.5$), which promotes exploitation from newly discovered cells. We also considered sampling cells with high performance progress. However, this often led to the selection of sub-optimal policies because starting to make progress from a low-performing policy is often the easiest way to make progress.

Hyperparameters

We use fully connected neural network controllers with two hidden layers of size 256, *tanh* non-linearities, Xavier initialization (He et al., 2015), an Adam optimizer (Kingma & Ba, 2015) with learning rate 0.01 and a l2-coefficient of 0.005 for regularization. We run ES for $N_g = 10$ consecutive generations with a population size of $n = 10^4$ and a noise parameter $\sigma = 0.02$. After each generation, we evaluate the average fitness $\overline{f}$, average behavioral characterization $\overline{\text{BC}}$ and associated novelty score $N(\theta_g)$ of θ_g over $n_e = 30$ episodes. Novelty is computed as the average distance between the controller's $\overline{\text{BC}}$ and the one of its $k = 10$ nearest neighbors in the archive of all past BCs, regardless of whether they were added to the BM. The code and environment are provided at `https://github.com/uber-research/Map-Elites-Evolutionary`.

Damage Recovery

The *intelligent trial and error* algorithm (IT&E) integrates the evolution of a behavioral map using MAP-ELITES and a subsequent procedure enabling agents to recover from damage by searching the BM for efficient recovery policies (Cully et al., 2015). Here, we reuse this downstream task to compare the traditional implementation of MAP-ELITES

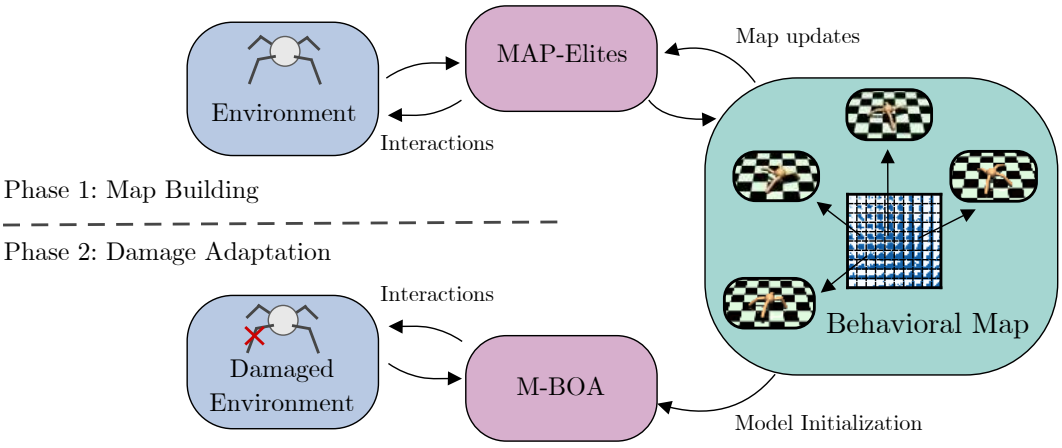


Figure 2.9: Building repertoires for damage adaptation. Phase 1: MAP-ELITES builds a repertoire of diverse and high-performing behaviors. Phase 2: M-BOA builds a model of the objective function under the perturbed conditions to find a policy robust to the damage.

based on GA (ME-GA) and our proposed ME-ES variants (see Figure 2.9). Once the behavioral map has been filled by ME-GA or ME-ES, we can use it for damage adaptation (e.g. loss of control for one or several joints). For the adaptation procedure, we use the map-based Bayesian optimization algorithm (M-BOA) (part of IT&E, Cully et al., 2015). Bayesian optimization is especially suitable to find the maximum of an unknown objective function for which it is costly to obtain samples. We define the objective function as $\mathcal{F} : \text{BC} \rightarrow \text{F}$. M-BOA initializes a model of $\mathcal{F}$ using the behavioral map filled by MAP-ELITES and updates the model using Gaussian process regression (GP). Data acquisition boils down to the evaluation of a policy θ in the perturbed environment, providing a new pair $(\text{BC}(\theta), \text{F}(\theta))$ to update the model $\mathcal{F}$. We follow the implementation of Cully et al. (2015) and use a Matérn kernel function with parameters $\nu = 5/2$ and $\rho = 0.03$ as well as hyperparameters $\kappa = 0.3$ and $\sigma^2_{\text{noise}} = 0.01$ for the upper confidence bound exploration parameter and noise prior respectively (see code).

The Autotelic Exploration of ME-ES

At first sight, it is not apparent that ME-ES is conducting an autotelic exploration. Where are goals here? In Section 1.2, we said that *goals* were parametric problems the agent could represent, target and self-evaluate on. Here the ME-ES agent selects its own problems: improving diversity (novelty objective) or improving performance (fitness objective) starting from a given policy characterized by a given BC. The goal representation is composed of both the average BC representing the sampled cell and the choice of the objective (novelty or fitness). As it samples an objective and a starting cell, the agent conditions its behavior on the selected goal by running the policy found in that cell. Just as in IMGEPs, ME-ES leverages cross-goal learning. It does not matter what the initial goal was—improving diversity or performance from this or that cell—the policy that results from the ES update can be kept either because it improved diversity (Rule 1) or because it improved (local) performance (Rule 2). ME-ES samples goals as a way to

organize its exploration and uses self-generated rewards (novelty objective) to conduct an autotelic exploration. Here, however, the external reward is part of the process.

2.2.4 Baselines and Controls

In this study, we test ME-ES in the *damage adaptation* and the *deep exploration* settings. The two applications have different baselines and ablations conditions, which we describe now.

MAP-Elites with Genetic Algorithm

We use the traditional implementation of MAP-ELITES based on GA (ME-GA) as a baseline in the *damage adaptation* experiments to highlight the lift from using ES instead of GA to power MAP-ELITES in high-dimensional control tasks. To ensure a fair comparison, it is important to keep the number of learning episodes constant between ME-GA and ME-ES. Here, ME-ES requires 10.030 episodes for each new policy (10.000 evaluations for the offsprings and 30 evaluations of the updated parent policy). As such, we enable ME-GA to add up to 334 new policies per generation, each being evaluated 30 times (10.020 episodes per generation).

We do not use ME-GA as a baseline in the *deep exploration* experiments because the task builds on the Humanoid-v1 environment where GAs were found not to be competitive with ES, even with 150x more environment interactions (Such et al., 2017).

Novelty Search and Evolution Strategies

We compare ME-ES to NS-ES and its variants (Conti et al., 2018) in the *deep exploration* experiments. Since these baselines do not build behavioral repertoires, they are not used for the *damage adaptation* application. NS-ES replaces the performance objective of ES with the same novelty objective as ME-ES EXPLORE. Instead of a behavioral map, NS-ES and its variants keep a population of five parent policies optimized in parallel via ES. NS-ES never uses fitness for optimization and thus is a pure exploration algorithm. We also compare against two variants of NS-ES that include a fitness objective: 1) NSR-ES optimizes the average of the fitness and novelty objectives, while 2) NSRA-ES implements an adaptive mixture. In NSRA-ES, the mixing weight leans towards exploration (novelty objective) when the best fitness stagnates and shifts back towards exploitation (fitness objective) when new behaviors are discovered that lead to higher performance.

The weight adaptation strategy of NSRA-ES was modified in this paper. In the original work, the weight starts at 1 (pure exploitation), decreases by 0.05 every 50 generations if fitness does not improve and increases by 0.05 at every improvement. In practice, we found this cadence too slow to adapt appropriately, as the mixing weight can only go to 0 after 1000 generations in the best case. Thus, we reduce the update period to 20 generations and remove the bias towards exploitation at the start by setting the mixing weight to 0.5 initially.

We present two experiments in the following sections. Recall that we implement

policies by fully connected neural networks with two hidden layers of size 256 (about 10^5 parameters). This results in search spaces that have orders of magnitude more dimensions than those used in traditional QD studies (Lehman & Stanley, 2011a; Mouret & Clune, 2015; Lehman & Stanley, 2011b; Cully & Demiris, 2017). For each treatment, we report the mean and standard deviations across five seeds, except for ME-GA, for which we present three seeds. ME-GA is about three times slower to run (> 3 days) and does not manage to learn recovery policies in the *damage adaptation* task.

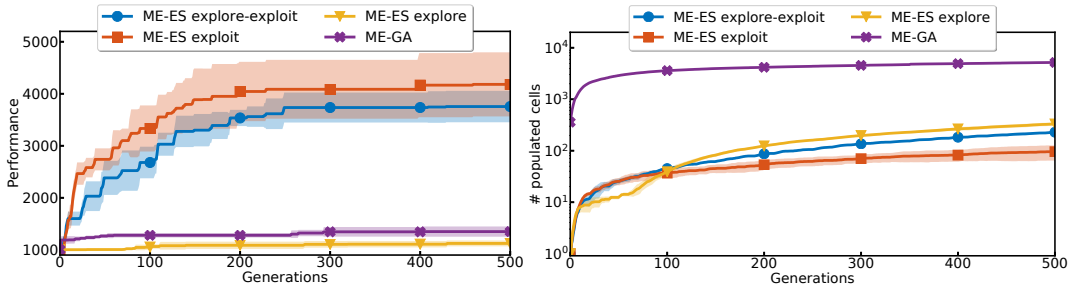
2.2.5 Application to Damage Recovery

We compare the quality of the BM generated by the different versions of MAP-ELITES for damage recovery. In the first phase, we build a behavioral map with each of the MAP-ELITES algorithms and, in the second phase, we use the behavioral map created in phase 1 to help the agent recover from damage to its joints via M-BOA (Figure 2.9). The damage applied to the joints, indexed by $\mathcal{J}_i$, include:

- One joint cannot be controlled ($\mathcal{J}_{i \in [\![0,7]\!]}$).
- One full leg cannot be controlled ($\mathcal{J}_{0,1}, \mathcal{J}_{2,3}, \mathcal{J}_{4,5}, \mathcal{J}_{6,7}$).

Domain

We evolve a repertoire of walking gaits in the Mujoco Ant-v2 environment (Brockman et al., 2016). The BC is a 4D vector where each dimension represents the proportion of episode steps where each leg is in contact with the floor ($[0, 1]^4$). Different walking gaits would be mapped to different BC, thus exploring the BC space might lead to different walking gaits. We discretize the behavioral space into 10 bins along each dimension for a total of 10^4 cells. The fitness is a mixture of the final x position and a cost on the torque for each joint. This incentivizes the agent to move as far as possible along the x-axis as possible without exerting too much energy.

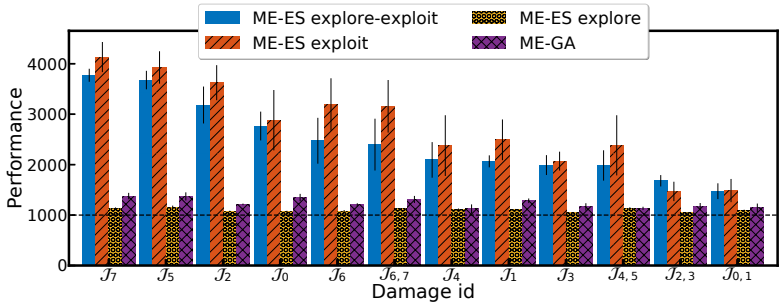


(a) (b)

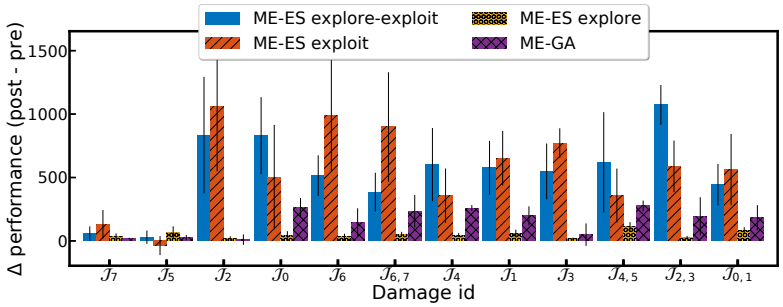
Figure 2.10: *Ant* behavioral map. a) Map performance (best performance in map). b) Map coverage (# populated cells), log scale on y-axis.

Results – Map Building

Figure 2.10a shows the evolution of the map's best performance and Figure 2.10b shows the map's coverage. As ME-GA adds up to 334 times more policies to the map at each generation, its coverage is orders of magnitude higher than ME-ES variants. However, the best performance found in the map is relatively low for ME-GA. ME-ES EXPLORE also shows poor performance across its map. In contrast, ME-ES EXPLOIT and ME-ES EXPLORE-EXPLOIT show high performance, on par with Twin-Delayed Actor-Critic (≈ 4200) (Fujimoto et al., 2018) but below Soft-Actor Critic (≈ 6000) (Haarnoja et al., 2018b), two state-of-the-art DRL algorithms for this task.



(a)



(b)

Figure 2.11: Damage adaptation. a) Performance of the best recovery policy. The joint damage is represented as $\mathcal{J}_i$ where i is the identifier of the broken joint or joints ($i \in [\![0, 7]\!]$). The dashed line represent the score of an agent that does not move forward but does not fall. b) Difference in performance, post- and pre-behavioral adaptation.

Results – Damage Adaptation

M-BOA is run on the behavioral maps obtained in each condition. Recovery policies extracted from ME-GA and ME-ES EXPLORE maps do not achieve high scores (Figure 2.11a). The scores are around 1000, which corresponds to a motionless Ant (not moving forward, but not falling either). In contrast, ME-ES EXPLORE-EXPLOIT and ME-ES EXPLOIT can recover from joint damage and the Ant can move in all damage scenarios (Figure 2.11a). In Figure 2.11b, we highlight the difference in performance before and after adaptation. We compute pre-adaptation performance by evaluating the highest performing pre-damage policy in each of the post-damage environments. The post-adaptation performance is

computed by evaluating the recovery policy in each of the post-damage environments. In most cases, ME-ES EXPLORE-EXPLOIT and ME-ES EXPLOIT are able to significantly improve upon the pre-damage policy (Figure 2.11b). Averages and standard errors are computed over different runs of MAP-ELITES + M-BOA (different seeds) in both figures. We average performance measures over 30 episodes.

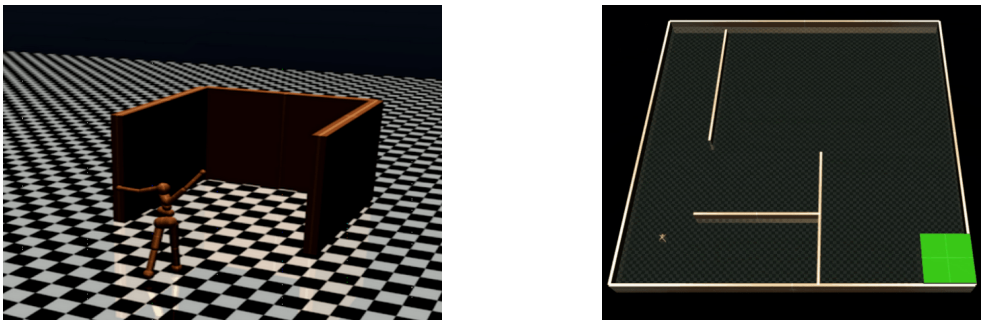
Interpretation

ME-GA does not scale to the high-dimensional Ant-v2 task, probably because it relies on random parameter perturbations (i.e. a GA) for optimization, which does not scale to high-dimensional policies (Such et al., 2017). ME-ES EXPLORE performs poorly, as it is not interested in performance, but only in exploring the behavioral space, most of which is not useful to solve the task. The exploitation ability of ME-ES EXPLORE only relies on its directed exploration component to find higher-performing solutions by chance. Indeed, neither ME-GA nor ME-ES EXPLORE leverages directed exploitation. ME-ES EXPLOIT only focuses on exploitation but still performs undirected exploration by retaining novel solutions that fill new cells. While ME-ES EXPLORE-EXPLOIT targets performance only half of the time, it still finds a BM that is good for damage recovery. Its exploration component enables better coverage of the behavioral space, while its exploitation component ensures high-performing policies are in the map. A good BC space coverage is a poor indicator of adaptation performance, whereas having a high-performing, somewhat populated map is a good indicator. Directed exploitation seems to be required to achieve good performance in the *Ant-v2* task. Undirected exploration—as performed by ME-ES EXPLOIT—seems sufficient to build a behavioral map useful for damage recovery, as its recovery performance is similar to that of variants using directed exploration and exploitation (ME-ES EXPLORE-EXPLOIT).

2.2.6 Application to Deep Exploration

Domains

We use two modified domains from OpenAI Gym based on *Humanoid-v2* and *Ant-v2* (Figure 2.12) (Brockman et al., 2016). In *Deceptive Humanoid*, the humanoid robot faces a U-shaped wall (like in Conti et al., 2018), while in *Ant Maze* the Ant is placed in a maze similar to the one used in Frans et al. (2018). Both environments possess a strongly deceptive reward, whose gradient leads the agent directly into a trap. For both environments, we use the final (x, y) position of the agent as the BC. In *Deceptive Humanoid*, fitness is defined as the cumulative reward, a mixture of velocity along the x-axis and costs for joint torques (for more details, see Brockman et al., 2016). In *Ant Maze*, the fitness is the Euclidean distance to the goal (green area). We run policies for 3.000 timesteps in *Ant Maze* and up to 1.000 timesteps in *Deceptive Humanoid*, less when the agent falls over.

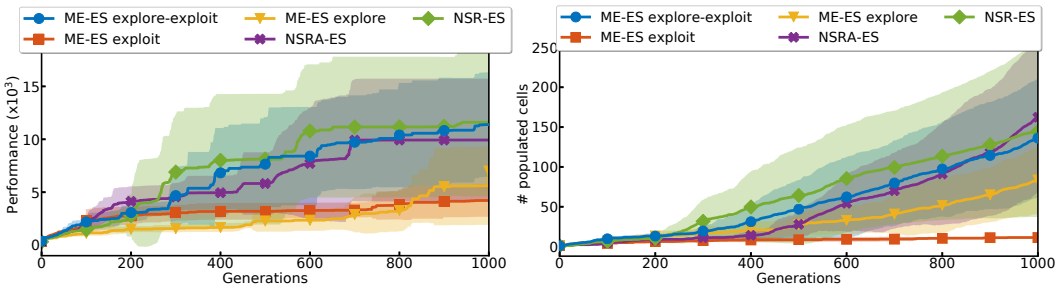


(a) (b)

Figure 2.12: Domains for the deep exploration study. a) *Deceptive Humanoid* domain. b) *Ant Maze* domain. Here the goal is the green area and the trap is in the cul-de-sac to the right of the agent's starting position.

Results – *Deceptive Humanoid*

Undirected exploration (ME-ES EXPLOIT) is insufficient to get out of the trap and directed exploration alone (ME-ES EXPLORE) falls short as well (Figure 2.13a). That said, like NS-ES, ME-ES EXPLORE eventually explores around the wall, as indicated by its performance above 3.000. Algorithms implementing directed exploration and directed exploitation manage to go around the wall and achieve high performance (Figure 2.13a), regardless of whether they use decoupled (ME-ES EXPLORE-EXPLOIT) or coupled (NSR-ES, NSRA-ES) exploration-exploitation. Surprisingly, ME-ES EXPLORE displays poor map coverage despite its focus on exploration (Figure 2.13b).



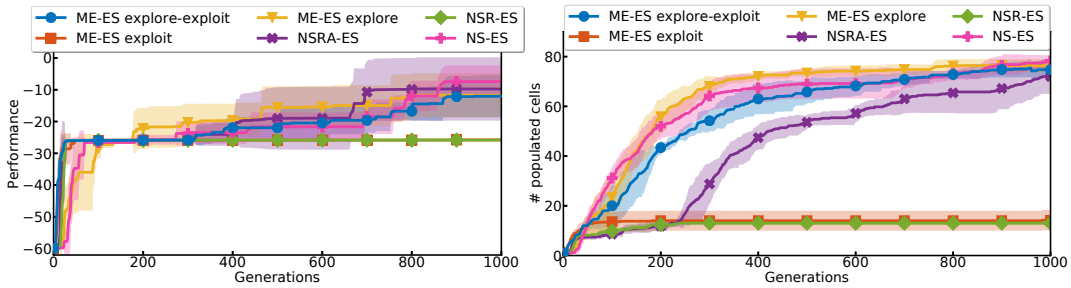
(a) (b)

Figure 2.13: *Deceptive Humanoid* behavioral map. a) Map performance (best in map). Being stuck in the trap corresponds to around 3000. b) Map coverage (# populated cells).

Results – *Ant Maze*

In the *Ant Maze*, both ME-ES EXPLOIT and NSR-ES get stuck in the deceptive trap, as indicated by a score of -27. All other methods (ME-ES EXPLORE, NS-ES, ME-ES EXPLORE-EXPLOIT, and NSRA-ES) can avoid the trap and obtain a score closer to 0. Examining the exploitation ratio of NSRA-ES, we observe that all runs quickly move towards mainly

performing exploration. That said, some runs have a ratio that falls to pure exploration and stays there whereas, in others, the algorithm manages to find a high-performing policy triggering an increase in preference towards performance (Figure 2.15)



(a) (b)

Figure 2.14: *Ant Maze* behavioral map. a) Map performance (negative distance to the goal area in map). The trap corresponds to −27. b) Map coverage (# populated cells).

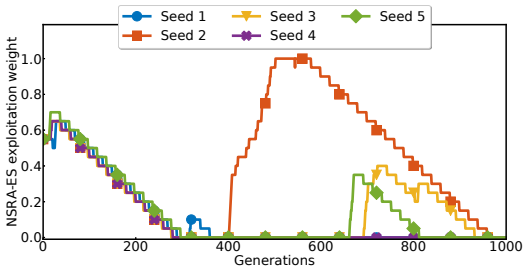


Figure 2.15: *NSRA-ES weight adaptation.* Evolution of the adaptive weight of NSRA-ES that tunes the ratio between the performance (exploitation) and the novelty (exploration) objectives for five runs. Higher values correspond to higher levels of exploitation. Seeds 1 and 4 get stuck in pure exploration and, thus, lose sight of the goal.

Interpretation

Deceptive Humanoid. In this environment, simply following the fitness objective can only lead the agent into the U-shaped trap. Because the environment is open, pure exploration does not guarantee the discovery of a high-performing behavior either, as it is possible to explore in directions misaligned with performance endlessly, a phenomenon previously encountered by NS-ES in Conti et al. (2018). This explains the poor performance of both ME-ES EXPLORE and ME-ES EXPLOIT. However, combining exploration and exploitation provides the best of both worlds. We observe that all algorithms that do so navigate around the wall (NSR-ES, NSRA-ES, ME-ES EXPLORE-EXPLOIT). As there is no need to follow gradients orthogonal to those of the performance objective to succeed, coupling exploration and exploitation (e.g. NSR-ES) is sufficient to achieve high fitness. The poor map coverage of ME-ES EXPLORE may be a result of it only optimizing for

performance indirectly via MAP-ELITES map updates. As a result, it is never encouraged to learn to walk efficiently, and thus unable to fill large portions of the map. In the NS-ES paper, however, NS-ES manages to make the humanoid walk using pure directed exploration (Conti et al., 2018). This may be because NS-ES only updates a population of five policies, whereas any policy in the behavioral map of ME-ES EXPLORE can be updated. A given ME-ES EXPLORE policy will thus be updated less frequently than any of the five NS-ES policies. This dilution of the number of updates of each policy might explain why ME-ES EXPLORE is less efficient at exploring than NS-ES in this environment. Because the environment is unbounded, different runs may explore different directions of the BC space, resulting in high standard deviations.

Ant Maze. In the Ant Maze task, pure exploitation will lead the agent to the deceptive trap (Figure 2.12a). Pure exploration, because it is restricted by the enclosed maze, eventually leads to the goal. This explains the success of directed exploration algorithms (NS-ES, ME-ES EXPLORE) and the poor performance of directed exploitation algorithms (ME-ES EXPLOIT). This environment requires a more extensive exploration procedure to achieve the goal than does *Deceptive Humanoid*. In *Ant Maze*, the agent needs to avoid two traps and needs to make turns to achieve its goal, while in *Humanoid Deceptive*, the agent can simply walk at an angle of 45 degrees to achieve high scores. As such, an even mix of the performance and novelty objectives (NSR-ES) fails, as we try to average two contradicting forces. NSRA-ES has a better chance of getting out of the trap by adding more and more exploration to the mix until the agent escapes. In doing so, however, the mixing weight can go down to $w = 0$ and only consider the novelty objective, thus losing sight of the goal (Figure 2.15). ME-ES EXPLORE-EXPLOIT can escape the trap because of its decoupled objectives; exploration steps will lead it to explore the environment until it reaches a point where exploitation steps allow it to reach the goal.

2.2.7 Discussion

This study presents ME-ES, a new QD algorithm scaling the powerful MAP-ELITES algorithm to deep neural networks with ES. We present three variants of this algorithm: 1) ME-ES EXPLOIT, where ES targets a performance objective; 2) ME-ES EXPLORE, where ES targets a novelty objective and 3) ME-ES EXPLORE-EXPLOIT, where ES targets performance and novelty objectives in an alternating fashion. Because the behavioral map of MAP-ELITES implicitly implements undirected exploration (update rule 1) and undirected exploitation (update rule 2), these variants implement directed exploitation with undirected exploration, directed exploration with undirected exploitation, and decoupled directed exploration and exploitation, respectively.

In a first application, ME-ES curates an archive of diverse and high-performing policies that we use for damage adaptation. We show that ME-ES manages to recover from joint damage in a high-dimensional control task while ME-GA does not. In a second application, we use ME-ES to solve strongly deceptive exploration tasks. Here, we show that ME-ES EXPLORE-EXPLOIT performs well in both open and closed domains, on par with the state-of-the-art exploration algorithm NSRA-ES. As in most DRL control tasks, the policies optimized in this work contain about 10^5 parameters (Haarnoja et al., 2018b). Previous

work showed ES could scale to even larger policies (e.g. $\approx 10^6$ in Atari games) (Salimans et al., 2017), which suggests ME-ES could as well.

Others have conducted work similar to ours. Frans and colleagues tackle an environment similar to Ant Maze with hierarchical RL (Frans et al., 2018). Their method pre-trains independent low-level policies to crawl in the four cardinal directions and trains a high-level controller to pick directions to find the goal. On top of requiring domain knowledge to train low-level policies, hierarchical RL has trouble composing sub-policies. They solve this problem by resetting the agent to a default position between sub-sequences. With ME-ES, we learn to solve the deep exploration problem by controlling joints directly.

Concurrently to our work, Fontaine and colleagues proposed CMA-ME, a similar algorithm combining MAP-ELITES with CMA-ES (Hansen, 2016), another algorithm from the ES family (Fontaine et al., 2020). In addition to a performance objective, CMA-ME considers two others. The *improvement* objective rewards policies that discovered new cells or improved over already populated cells. The *random direction* objective rewards policies for minimizing the distance between their BC and a behavioral target vector randomly sampled from the behavioral space. Although the last objective does encourage exploration, it is not explicitly directed towards novelty as in ME-ES EXPLORE. Additionally, although CMA-ES is generally recognized as a very powerful algorithm, it is in practice limited to comparatively low-dimensional parameter spaces due to the algorithmic complexity induced by its covariance matrix updates ($O(n^2)$). Note that it might not be a limiting factor in our case, as lower-dimensional controllers might be able to solve our tasks.

In episode-based algorithms, each policy is associated with a single outcome description. This works fine in deterministic environments but can fall short in stochastic ones (e.g. various initial states, stochastic opponents, procedurally-generated worlds); when each new rollout of the policy leads to a different outcome. The standard way to handle this problem is to report the average outcome computed over several episodes (here, 30), i.e. to represent the distribution of outcomes by its empirical mean. This can be a problem when the distribution has high variance, and might even lead to unrealistic outcomes in the case of multi-modal distributions—if the outcome is either 1 or 0, then averaging several of them leads to a mean in]0, 1[, an impossible outcome. We could fight this phenomenon by adding a cost on the outcome distribution's variance to the fitness measure, thus pushing the QD algorithm to evolve policies with low variance. Instead of fighting the problem, we could also turn it to our advantage by pairing policies with a representation of the distribution of outcomes they lead to. In addition, to evolve policies to reach various outcomes, these algorithms could evolve policies that achieve low-variance outcomes, high-variance outcomes, bi-modal outcomes, low-risk outcomes, etc.

Because ME-ES only affects the optimization procedure of MAP-ELITES, any application leveraging behavioral maps could benefit from its improved optimization efficiency. In addition to the demonstrated applications for exploration (Section 2.2.6) or damage adaptation (Section 2.2.5), previous works have used MAP-ELITES for maze navigation (Pugh et al., 2016) or to explore the behavioral space of soft robotic arms and the design space of soft robots (Mouret & Clune, 2015). It can also co-evolve a repertoire of diverse images matching each of the ImageNet categories (Nguyen et al., 2015) or generate diverse collections of 3D objects (Lehman et al., 2016).

ME-ES could also leverage advances proposed for either MAP-ELITES or ES. An interesting potential improvement is the use of centroidal Voronoi tessellation to partition high-dimensional behavioral spaces into a tractable number of cells (Vassiliades et al., 2017). The behavioral adaptation phase can also be modified to allow adaptation without the need for resets or for human interventions in the case of real robots (Chatzilygeroudis et al., 2018). Another interesting lead is the *evolvability* objective proposed in Lehman et al. (2018a). This objective orients the search to *evolvable* areas of the parameter space—areas from which random perturbations lead to high behavioral diversity. This seems like a very interesting idea to favor exploration of outcome spaces.

More generally, ME-ES could leverage a set of various objectives in parallel. As the bottleneck lies in the computation of the 10^4 offspring, various objectives can be computed and optimized simultaneously (novelty, performance, combinations of them as in NSR-ES, evolvability, etc). Each resulting policy can be tested 30 times and considered a candidate for the behavioral map updates, with a negligible increase in the cost $N_{\text{objectives}} \times 30 \ll 10^4$.

In the *deep exploration* experiment, ME-ES could also be extended to perform hierarchical evolution. In such an algorithm, solutions are *chains* of policies, where each link is a policy run for a given number of time steps. After sampling a cell and retrieving the associated chain, the algorithm can either mutate the last policy of the chain or add a new policy to it. This would implement the *go-explore* strategy proposed in Ecoffet et al. (2021), where the agent first returns to a behavioral cell (first policies are fixed) before exploring further (mutating the last policy or chaining a new one).

These experiments were run in large computing clusters (with 1000 CPUs), which makes ME-ES a *very sample-inefficient* method. Since this study was published, a similar approach aimed at scaling QD algorithms via a population of policies trained with DRL (Cideron et al., 2020a). This approach achieves large gains in sample efficiency in the *AntMaze* but does not seem to scale to the higher dimensional *HumanoidDeceptive* yet. In addition, DRL methods are so far limited to behavioral descriptions that are time-specific—i.e. computed from a state, not a trajectory. Because ME-ES relies on the *episode-based* ES algorithm, it can handle BCs that characterize complete trajectories, like the one used in the damage adaptation application to characterize the Ant's walking gaits.

The addition of ME-ES and its variants to the QD family opens new avenues, as the exploration and diversity properties of QD algorithms can now be scaled to higher-dimensional problems. Sample inefficiency put aside, ME-ES is a powerful algorithm to implement autotelic exploration in the context of an external task. The curated repertoire of skills helps solve the external tasks, makes the agent robust to subsequent damage and, more importantly, contains a rich repertoire of diverse and locally high-performing skills.

It is now time to close this chapter. Let us review what we learned. Here, we focused on the standard RL setting: an agent must learn how to solve an external task using learning signals generated by a hand-defined reward function. We looked at problems that required agents to demonstrate strong exploration abilities. In the first study (2.1), we showed that even problems that appeared simple might hide deceptive and sparse reward signals, thus require strong exploration. In these cases, crude exploration methods used by standard implementations of DRL algorithms fall short. Strong exploration needs to be temporally structured, directed and decoupled.

In the first study (Section 2.1), we proposed GEP-PG and showed that a simple exploration method showing these characteristics (GEP) could already boost the exploration abilities of standard DRL algorithms significantly. In the second study (Section 2.2), we showed that the family of autotelic exploration mechanisms that rely on genetic algorithms (NS, QD, POP-IMGEPs) could have limits in high-dimensional tasks. To fight this issue, we proposed a new QD variant called MAP-ELITES with ES (ME-ES). In these two studies, we demonstrated that autotelic exploration methods could be leveraged to solve pre-defined high-dimensional control tasks that require strong exploration.

The exploration mechanisms of GEP-PG and ME-ES share an essential similarity contrasting them from others. Both are organized around an objective that is orthogonal to the pre-defined task—*the curation of a repertoire of diverse skills*. These skills are represented by *goals* in GEP-PG and *behavioral characterizations* in ME-ES. In addition to solving the pre-defined task more efficiently, the trained agents can also call upon any of these learned skills. This is the first step towards our objective of training autonomous agents to explore open-ended environments and grow repertoires of skills.

However, our agents are still far from being autonomous. GEP-PG and ME-ES both require the pre-definition of the outcome/behavioral space that the autotelic exploration tries to cover. Associated reward functions (based on distance metrics in outcome space) are also provided. In GEP-PG, the autotelic exploration can only find coarse policies (see the walking-forward behavior in Half-Cheetah) and requires DRL to fine-tune them. In the remaining of this research, we move on towards more autonomous agents that learn in reward-free environments, target a diversity of goals, organize their learning trajectories (Chapter 3) and, eventually, learn their own outcome representations and generate their own rewards (Chapters 5 and 6). To enable efficient autotelic exploration in high-dimensional tasks, we turn towards *goal-conditioned* RL methods and introduce the first RL-based IMGEP algorithms (RL-IMGEP).

Chapter 3

Autotelic Acquisition of Diverse Repertoires of Skills

The previous chapter introduced ideas from the fields of developmental robotics (IMGEPs) and evolutionary robotics (MAP-ELITES) to help the resolution of pre-defined RL tasks. GEP-PG and ME-ES agents efficiently organize their exploration as the acquisition of repertoires of skills but do so as an auxiliary task to help solve the main external task. These studies helped us get a foot in the door: concepts from developmental robotics are relevant to RL.

From this chapter, we pass through the door entirely and set one of the main targets of developmental robotics as our main objective: to design autonomous learning agents to tackle the acquisition of open-ended skills repertoires. With this objective, we depart from the standard RL setting: there is no external task to solve anymore. Agents are set in a reward-free environment and need to grow sets of skills on their own.

To tackle this problem, we rely on our RL-IMGEP framework, the set of methods training artificial agents to represent, generate, pursue and master their own goals via goal-conditioned RL methods (see formal definition in Section 1.2). RL-IMGEPs differ from previous population-based implementations of IMGEPs as they train a single goal-conditioned policy and benefit from the advantages of state-of-the-art DRL methods: stronger generalization abilities, a higher capacity to handle high-dimensional inputs (e.g. visual or linguistic inputs), higher robustness to initial conditions and an increased sample efficiency (by leveraging step-based feedback). RL-IMGEPs also differ from goal-conditioned RL methods because they do not solve the same problem. Indeed, RL-IMGEPs solve the *autotelic RL problem*: they learn repertoires of skills by generating their own goals and rewards. This increased autonomy, however, comes with costs. RL-IMGEP agents can neither rely on engineers to provide well-defined goal definitions and rewards nor know in advance which goals can be mastered, which are trivial or impossible. They also need to sculpt their own learning trajectories, decide on *what* and *when to learn*.

This chapter introduces CURIOUS, the first RL-IMGEP agent. CURIOUS does not answer all of our questions just yet but proposes an answer to two of these. First, we ask how agents can efficiently target a wide diversity of goals that extends to different types of affordances. Second, we ask how to organize a learning trajectory when the set of goals demonstrates realistic properties—some goals can be easy, others impossible, some goals might require to master others first, etc. In the next two studies, we answer the

first question with a new learning architecture called *modular universal value function approximators* (M-UVFA) (Section 3.1) and answer the second with a mechanism that automatically organizes a curriculum with intrinsic motivations oriented towards learning progress (Section 3.2). The combination of these two methods forms the CURIOUS algorithm.

3.1 From Multi–Goal to Multi–Goal–Types: Modular-UVFA

Goal-conditioned reinforcement learning (GC-RL) is the set of RL methods training agents to target distributions of goals (Schaul et al., 2015). In Section 1.5, we defined a *goal* as a pair made of 1) a compact goal representation (goal encoding) and 2) an associated reward function to measure progress towards it. *Universal value function approximators* (UVFA) first proposed to train a unique RL policy to target a—possibly infinite—set of goals (Schaul et al., 2015). The idea is simple: the policy should take as input both the current state and goal. The general intuition behind UVFAs is that representations acquired to solve one goal might transfer well to other goals. In other words, sharing representations across goals should enable efficient transfer between skills and improve generalization to new goals.

The parallel with methods introduced in the previous chapter is striking. The space of goal encodings in UVFAs—i.e. the *goal space*—is similar to the *outcome space* of GEP-PG and the *behavioral space* of ME-ES. In all cases, the agent's trajectory is projected into a low-dimensional representation space characterizing its behavior. One might consider that defining a single goal space restricts the set of behaviors an agent can acquire. Suppose the outcome representation characterizes the 2D position of an agent in a maze. In that case, the agent can only learn to explore the 2D space but cannot explore walking gaits or learn to find walls of specific colors. In short, the behavioral exploration is limited to a single low-dimensional outcome space.

In this study, we propose to extend the idea behind UVFAs. If agents can target multiple goals and transfer knowledge between them, could they also target multiple goals of different types involving different affordances and still benefit from skill transfer? As we said, goals are nothing else than reward functions and associated embeddings. If we could find a way to encode different goal types, we could train a unique policy to target them all. This study proposes such an encoding and tests its benefit on skill transfer.

3.1.1 Problem Definition and Related Work

Autonomous agents should be able to target many goals of many different types. A child, for example, can reach a given coordinate x with her hand to hide the sun or push the fork to position y to indicate that she does not want to eat anymore. Current GC-RL approaches only allow targeting one of these: agents can be trained to reach any target coordinates with their hand *or* push a block to any target coordinates on a table, not both (Plappert et al., 2018a). Indeed, standard implementations only allow the definition of one unique continuous goal space characterized by one goal-parameterized reward function—e.g. a binary function asserting that the hand is within a distance ϵ of its target.

Alternative approaches can learn about a discrete set of goals (e.g. Mankowitz et al., 2018).

We can see *reaching* and *pushing* as two *goal-modules*. Each module describes a particular affordance, has its own continuous goal space (3D target for the hand; 2D target for the fork), and its own target-parameterized reward function (judging the distance hand–target or fork–target). We thus define *modules* as a pair of a goal-parameterized reward function R_{M_i} and goal space $\mathcal{G}_{\mathcal{M}_i}$):

$$M_i = (R_{M_i}(\cdot \mid g_i \in \mathcal{G}_{\mathcal{M}_i}), \mathcal{G}_{\mathcal{M}_i}).$$

At each episode, the agent can sample a goal from one of the modules and target it.

Modular Goal Spaces in Manipulation Task

We implement an environment with modular goal spaces called the *Modular Goal Fetch Arm* (Figure 3.1). *Modular Goal Fetch Arm* is a simulated environment adapted from the OpenAI Gym suite (Brockman et al., 2016; Plappert et al., 2018a). The agent is embodied by a robotic arm facing two blocks randomly positioned on a table. It can target a set of modular goals:

- (M_1) *Reach* a 3D target with the gripper,
- (M_2) *Push* block 1 onto a 2D target on the table,
- (M_3) *Pick and Place* block 1 on a 3D target,
- (M_4) *Stack* block 1 over block 2.

In some of the experiments, we also consider distracting modules: Push modules related to moving out-of-reach blocks (purple blocks in Figure 3.1). The agent controls the closing and the 3D Cartesian velocity of its gripper (4D continuous action space). The observation space has 40 dimensions (see Plappert et al., 2018a), plus 3 extra dimensions per additional distracting block.

Related Work

Let us discuss the most relevant related work and organize it along two dimensions: 1) whether learning occurs within one or several policies and 2) whether the goal spaces are modular or flat—see Table 3.1. Algorithms that train agents to learn about multiple goals can train one policy per goal (*goal-experts* approach). This is the case in Kaelbling (1993), where the algorithm trains one value function per goal and uses experience to update all of them in parallel (cross-goal learning). More recently, SAC-X proposed to train one policy per task in a multi-task RL setting designed to provide auxiliary tasks to help solve the most challenging task (placing blocks inside a closed box) (Riedmiller et al., 2018). In the IMGEP family, SAGG-RIAC also memorizes one policy for each goal (Baranes & Oudeyer, 2013). The goal-expert strategy might prevent catastrophic forgetting and goal interference because learning about a skill does not affect other skills. However, it

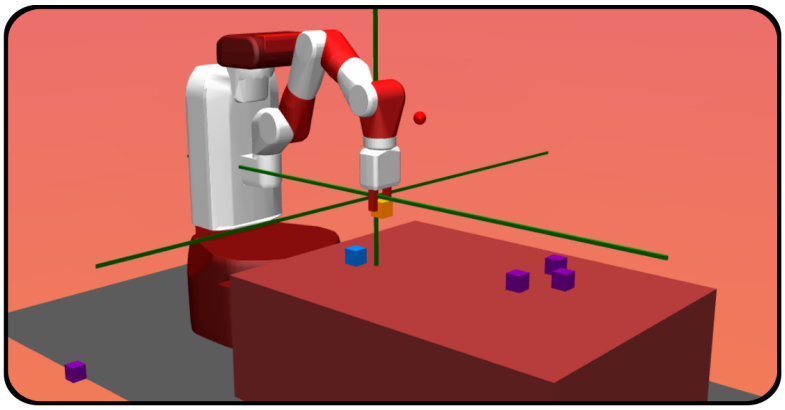


Figure 3.1: The *Modular Goal Fetch Arm* environment. The autotelic agent can set its own (modular) goals (Reach, Push, Pick and Place, Stack) and faces objects and distractors (out-of-reach purple blocks).

cannot leverage shared representations, thus may fail to perform efficient skill transfer. An alternative—and now the standard approach—is to train a single goal-conditioned policy and value function to target all goals with UVFAs (Schaul et al., 2015). All the methods described up to this point target *flat* goal spaces. In contrast, a population-based algorithm called MACOB proposes to target modular goals in a *modular goal-experts* approach: each policy is mapped to an outcome in each of the modules (Forestier & Oudeyer, 2016b), see Nguyen & Oudeyer (2012) for a similar approach.

Multi-goal approaches prove better than simply training a policy per goal because knowledge can be transferred between different goals using off-policy learning and hindsight learning (Andrychowicz et al., 2017). Off-policy learning reuses any transition to improve the current policy: transitions collected from an older version of the current policy (Lillicrap et al., 2016), from a population of exploratory policies (previous chapter, Section 2.1), or even from demonstrations (Vecerík et al., 2017). Hindsight learning allows reusing transitions collected while aiming at a particular goal to learn about any other (Andrychowicz et al., 2017). In the case of UVFA policies, the original goal is substituted by an imaginary one, a technique called *goal replay* or *goal substitution*. In HER, the substitute imaginary goal is sampled from the outcome achieved later in the trajectory (Andrychowicz et al., 2017). With discrete and finite sets of goals, one can sample imaginary goals uniformly (Kaelbling, 1993; Mankowitz et al., 2018). These mechanisms facilitate skill transfer and help increase the probability to observe rewards in reward-sparse environments. More complete reviews of relevant population-based IMGEPs and goal-conditioned RL methods are provided in Sections 1.2 and 1.4 respectively.

This chapter introduces three algorithms. The M-UVFA algorithm (this study) trains a unique policy to target modular goals. The CURIOUS algorithm (study in Section 3.2) builds on M-UVFA and adds an intrinsic motivation to organize learning trajectories. Finally, the UVFA-ME trains one goal-conditioned policy per module and will be used as a baseline—see Table 3.1.

Table 3.1: Classification of multi-goal algorithms. Underlined: autotelic algorithms (IMGEPs), (*): using intrinsic motivations for goal selection. *Italic*: population-based algorithms. **Bold**: algorithms introduced in this paper.

	n GOALS, n POLICIES	n GOALS, 1 POLICY
FLAT REPR.	GOAL-EXPERTS: (Kaelbling, 1993)	GOAL-CONDITIONED: UVFA (Schaul et al., 2015)
	SAC-X (Riedmiller et al., 2018)	HER (Andrychowicz et al., 2017)
	*SAGG-RIAC** (Baranes & Oudeyer, 2013)	UNICORN (Mankowitz et al., 2018)
MODULAR REPR.	MOD-GOAL-EXPERTS: *MACOB** (Forestier & Oudeyer, 2016b)	MODULAR-GOAL-CONDITIONED:
	GOAL-CONDITIONED MODULE-EXPERTS: **UVFA-ME**	**M-UVFA**, **CURIOUS***

3.1.2 A Modular UVFA Architecture

To train agents to master modular goal spaces, the first thing that comes to mind is to train multiple goal-conditioned UVFA policies. We call this approach the *multi-goal module experts* (UVFA-ME). It shares representations between goals inside any given module thanks to the UVFA architecture but does not share them *across* modules. In our environment, however, different modules might require the same type of representations. The agent always needs to learn how to move its gripper and often needs to interact with blocks. In real life as well, most of our behaviors rely on compositions of lower-level behaviors including basic sensorimotor skills. To enable this transfer across modules, we introduce the *modular* UVFA architecture (M-UVFA) derived from UVFAs.

UVFAs aggregate the goal of the agent with its current state to form the input of a policy and critic implemented by deep neural networks (Schaul et al., 2015). With M-UVFA, we propose a new modular goal encoding enabling agents to target a rich diversity of modular goals within a single policy (*modular goal-conditioned* approach), see Figure 3.2.

Let us call $\mathcal{G}_{M_i}$ the goal space of module M_i and cardinal $c_i = |\mathcal{G}_{M_i}|$. The goal encoding is the concatenation of a module encoding m and target encoding g. The module encoding is a simple one-hot encoding of size N, where N is the number of modules. The target encoding g is of size $|\mathcal{G}| = \sum_{i=1}^{N} c_i$. It is the concatenation of the targets for all modules. When the agent selects a module M_i and a target $g_i \in \mathcal{G}_{M_i}$, all the other modules' targets are masked by zeros. Zero weights effectively freeze backpropagation so that, in the first layer, learning only affects the weights relevant to the selected module.

Figure 3.2 depicts a simple example of the M-UVFA actor-critic architecture with two modules of cardinals 2 and 1, respectively. The actor implements the policy and maps the input $[s_t, g, m]$ to the following action a_t. The action is then concatenated to a copy of the actor's input to feed the critic $[s_t, g, m, a_t]$. The critic provides an estimate $Q(s_t, g, m, a_t)$ of the Q-value. The critic and the actor are then trained using DDPG (Lillicrap et al., 2016), although any other off-policy learning method could be used (e.g. TD3 (Fujimoto et al., 2018), or DQN for the discrete case (Mnih et al., 2015)). More details about DDPG can be found in Section 1.1 or the original paper (Lillicrap et al., 2016).

Learning about all modules and goals within the same network helps generalization via *weight sharing*. Indeed, weights used to process goal x from module y are also used to handle goal x' from module y'. Because these parameters are shared, training on

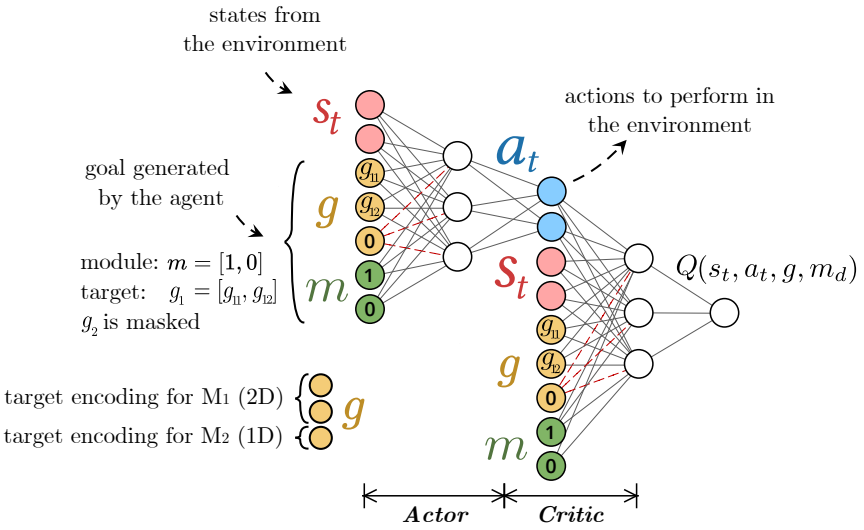


Figure 3.2: Modular goal-conditioned actor-critic architecture (M-UVFA). Toy example with two modules parameterized by targets g_1 (2D) and g_2 (1D), respectively. Here, the agent targets g_1 in module M_1, as specified by the one-hot module encoding $m = \langle 1, 0 \rangle$. The target g_2 in M_2 is masked by zeros, which prevents learning in the weights depicted with dashed red lines. The actor (left) computes the action a_t. The critic (right) computes the Q-value.

one goal helps crafting representations that can be leveraged to solve others. In other words, knowledge is transferred across modules and goals. But, it is not all. Transfer also occurs via *hindsight learning*. Hindsight learning increases transfer across goals from the same module by replacing the initial goal with one of the outcomes achieved later in the trajectory (Andrychowicz et al., 2017). We use the same mechanism to transfer knowledge across modules. When a trajectory is collected while aiming at a given target from a given module, we reuse it to learn about targets from other modules as well. To this end, we modify the modular goal encoding (target encoding + module encoding): 1) the module encoding is sampled at random from the set of N modules (random module replay); 2) the target encoding is sampled from the outcomes *corresponding to the sampled module* achieved later in the trajectory (standard goal replay).

3.1.3 Efficient Transfer Across Modules and Goals

We went over good reasons to expect a shared modular policy to outperform a set of independent goal-conditioned policies. This section tests this hypothesis empirically. Let us consider a setup with the four achievable modules and four distracting modules (pushing out-of-reach blocks). We test the M-UVFA algorithm against two baselines:

1. A *flat multi-goal* architecture (HER). This algorithm does not represent goals in a modular fashion but with flat encodings. The corresponding goal is selected uniformly inside $\mathcal{G}$, a holistic goal space such that $\mathcal{G} = \prod_{i=1}^{N} \mathcal{G}_{M_i}$. To generate a reward, the agent needs to satisfy the constraints described by all the modules at once. This goal-parameterized architecture is equivalent to UVFA and is implemented by a goal-conditioned DDPG with the HER mechanism.

2. A *goal-conditioned module-experts* architecture (UVFA-ME) where an expert UVFA policy is trained for each of the N modules. Each policy is trained with DDPG and HER one epoch every N with transitions collected in a buffer shared across module-experts. Thus, this baseline uses a form of transfer across modules and goals via HER applied on transitions collected while targeting other modules. However, it does not share representations across modules, only across goals from the same module. When evaluated on a particular module, the algorithm uses the corresponding module-expert.

We evaluate agents offline periodically and report their success rate over 95 episodes. During training and evaluations, the agent samples goals uniformly: it samples a module first, then samples a reachable target from the corresponding goal space.

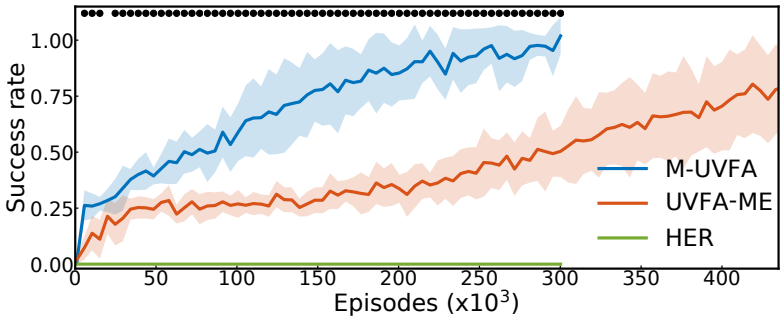


Figure 3.3: Impact of the policy and value function architecture. Average success rates computed over sets of goals uniformly sampled from the agents' goal spaces. This excludes goals from distracting modules, where the success rate is null by design. Mean ± std over 10 trials are plotted, while dots indicate significance when testing M-UVFA against UVFA-ME (Mann-Whitney test, $\alpha = 0.01$).

Figure 3.3 shows the evolution of the average success rate computed over sets of goals uniformly sampled from the agents' goal spaces for M-UVFA and the two baselines described above. The learning curve of HER stays flat. This is expected as almost none of the goals expressed in the flat goal space $\mathcal{G}$ corresponds to a real situation (e.g. the agent cannot reach a 3D target with its gripper while placing a block at another). The agent cannot fulfill the constraints of all modules simultaneously, thus receives no reward. This motivates the use of modular goal representations. The modular goal-conditioned approach M-UVFA learns the set of achievable skills much faster than the modular-expert variant UVFA-ME ($\approx 250 \cdot 10^3$ vs. $\approx 450 \cdot 10^3$ episodes).

3.1.4 Discussion

This study introduced modular-UVFA (M-UVFA), an extension of the UVFA framework enabling agents to learn about goals involving different affordances in a unique architecture. M-UVFA allows an efficient transfer of knowledge between different goals and different modules via weight-sharing and cross-module, cross-goal hindsight learning. This new architecture increases the diversity of skills that autotelic RL agents can learn.

However, we can wonder how far we can push this. Would it still work when considering hundreds of modules? Would it still work when the modules involve completely independent skills? As the number of targeted modules increases, the need to share representations and leverage transfer learning grows as well. Thus, we expect the benefits of using a shared policy over a set of module-expert to increase with the number of modules. This should be paired with increasing an architecture capacity. An alternative strategy is to train several modular policies, each handling a subset of the modules.

When the modules involve independent skills, module interference can counteract the benefits of transfer learning and hinder the global learning performance. How to predict when to expect synergistic or antagonistic transfer learning across modules is an interesting question, whose answer might help solve both problems. Synergistic modules could be grouped into a shared policy, while antagonistic modules would be separated. The field of *continual learning* studies such problems and might bring solutions. The ANML approach, for instance, proposes to meta-learn a modulation network conditioned on the current task to gate the forward and backward passes of the main network (Beaulieu et al., 2020). Although it is currently restricted to the supervised setting because of its high data requirements, similar mechanisms could learn to share representations across modules while specializing parts of the network to some of the modules, thus avoiding catastrophic interferences.

If we want our agents to build open-ended repertoires of skills, we need to discuss the addition of new modules as the agent learns. We could extend the target g and module m encodings and add new connections to the networks. Because the inputs corresponding to the non-targeted modules are all set to zeros (one-hot module encoding and target masking), new connections cannot hinder performance on existing modules. An interesting study could look into the transfer learning properties of M-UVFA by comparing the learning speed for each module addition. We hypothesize that, as the set of learned modules gets larger, new modules should be learned faster.

One aspect of knowledge transfer that M-UVFAs do not consider is the transfer of skills between equivalent objects. In M-UVFA, *pushing* modules corresponding to different blocks are considered independently, preventing the direct generalization from one block to another. This happens because object representations are simply concatenated and the features of each objects are encoded by object-specific weights. In the next chapter, we introduce object-centered architectures that handle object sets and directly encode object-wise invariance.

A central assumption of our work is that agents know *a priori* modular goal representations, associated goal spaces and reward functions. We consider them *priors* of the agent instead of coming from an external source—i.e. from the engineer. Of course, we should lift these assumptions to design *truly autotelic agents*. Indeed, RL-IMGEP agents must construct their own set of module and goal representations and generate their own rewards. Building directly on M-UVFAs, we could integrate a recent approach proposing to learn modular goal representations from experience using β-VAE trained on images (Laversanne-Finot et al., 2018). However, this approach is currently limited to simple settings with few objects. In the next part, we will relax these assumptions and learn goal representations and reward functions from social linguistic interactions with a simulated caretaker.

In the next study, we look at a related problem: now that we can learn about multiple goals of different types, how should we organize our curriculum?

outcome

3.2 Intrinsically Motivated Modular Architecture: CURIOUS

The previous study introduces M-UVFA, an extension of the UVFA architecture enabling agents to target multiple types of goals, to handle modular goal spaces. As the number of goals increases, it becomes more critical for autotelic agents to design their own learning trajectories and prioritize important goals.

UVFA, HER and UNICORN select the next goal to target uniformly (Schaul et al., 2015; Andrychowicz et al., 2017; Mankowitz et al., 2018). This is coherent with the common view that agents must comply with the desires of an engineer and pursue the goals they are instructed. In reward-less environments, on the other hand, the set of goals is not curated by an expert engineer anymore but generated by the agent itself. Thus, no assumption can be made. The set of possible goals can be huge, some goals might be trivial, some might be impossible, and some might require to master stepping-stones first. IMGEP agents need to *prioritize*. They must select which goal to target—i.e. which skill to practice—*here and now*.

The automatic organization of goal selection is a particular case of *automatic curriculum learning* (ACL): the family of mechanisms that automatically adapts the training data distribution by adjusting the selection of learning situations to the agents' capabilities—see our review in Portelas et al. (2020b). At the time of this study, only population-based IMGEPs used automatic goal selection mechanisms. They mainly rely on learning progress—agents should target goals where they progress the most (Baranes & Oudeyer, 2013; Forestier & Oudeyer, 2016b). Concurrently to this study, two alternative approaches appeared. The GOAL-GAN method trains a *generative adversarial network* (GAN) to generate goals of intermediate difficulty (Florensa et al., 2018). The *asymmetric self-play* method trains an adversarial policy with RL to generate goals of intermediate difficulty for the main agent (Sukhbaatar et al., 2018).

Intermediate difficulty can be a good proxy for expected learning progress in tasks that do not contain uncontrollable components. Indeed, agents might or might not reach goals involving uncontrollable components, no matter their performance. With uncontrollable goals, agents might detect intermediate difficulty where no progress can be expected. In contrast, actively measuring learning progress enables agents to discriminate controllable goals (positive LP) from uncontrollable ones (null LP).

Inspired by previous works on LP, we endow the M-UVFA architecture presented in the previous study with a module selection mechanism maximizing absolute learning progress. We call the resulting algorithm CURIOUS: the first RL-IMGEP agent learning about various goal types and crafting its own learning trajectory. In this study, we look at the impact of LP-based goal selection: 1) in environments with distracting modules (impossible goals) and 2) in environments where the agent undergoes forgetting or perturbations of its sensory apparatus.

3.2.1 Absolute Learning Progress for Goal Selection

This study builds on the previous one and reuses the modular-UVFA (M-UVFA) architecture—see Section 3.1. We propose to endow our CURIOUS agent with an intrinsic motivation towards high absolute learning progress (LP for short).

We can model module selection as a non-stationary multi-armed bandit problem (MAB) where the value of each arm (module) is the current absolute LP (Forestier & Oudeyer, 2016b). Absolute learning progress (LP) is the absolute derivative of the agent's competence on a particular module: $\text{LP}_{M_i} = |\frac{dC_{M_i}}{dt}|$. The competence $C_{M_i} : t \rightarrow p_{\text{success}}(t)$ is the probability of success at time t when sampling goals uniformly from the goal space $\mathcal{G}_{M_i}$ of module M_i. Here, the agent focuses its attention on modules for which it is making the largest absolute progress and pays little attention to modules that are already solved or unsolvable—i.e. for which LP stays small. Using the absolute value of LP also leads to prioritizing modules for which the agent is showing *decreasing performance*. This helps dealing with forgetting: the agent reallocates learning resources to the modules being forgotten.

Learning Progress Estimation

Since an autonomous agent is not provided with its true measure of competence or LP, it needs to approximate them for each module. To estimate its competence reliably, it uses some episodes (with $p_{\text{eval}} = 0.1$) to evaluate itself on random modules and targets without exploration noise. Agents store the binary outcomes of these rollouts in *success queues* S_i for each module M_i (success 1 or failure 0). In a similar way as Forestier & Oudeyer (2016b), agents measure their subjective competence as the success rate over the last L self-evaluations, and their subjective LP as the absolute difference between the current competence and the competence measured L self-evaluations earlier. More precisely, the subjective competence is measured by:

$$C_{M_i}(n^i_{\text{eval}}) = \frac{1}{L}\sum_{j=0}^{L-1} S_i(n^i_{\text{eval}} - j),$$

where n^i_{eval} is the number of self-evaluation rollouts performed by the agent in module M_i. The subjective LP_{M_i} after n^i_{eval} self-evaluations is then computed as:

$$\text{LP}_{M_i}(n^i_{\text{eval}}) = |C_{M_i}(n^i_{\text{eval}}) - C_{M_i}(n^i_{\text{eval}} - L)|.$$

Given the subjective LP measures, we use a simple bandit approach called *proportional probability matching* with an additional ϵ-greedy strategy for exploration. More precisely, we compute the *LP probabilities* $p_{\text{LP}}(M_i)$ as:

$$p_{\text{LP}}(M_i) = \epsilon \times \frac{1}{N} + (1-\epsilon) \times \frac{\text{LP}_{M_i}}{\sum_{j=1}^{N} \text{LP}_{M_j}},$$

where N is the number of modules. The ratio ϵ implements a mixture between a random module exploration (left term) and module exploitation via an LP-biased selection (right

term). The random exploration term samples modules that do not show any LP (i.e. already solved, too hard, or at a plateau). This way, the agent can check that it stays competent on modules that are already learned or insist on modules that are currently too hard.

Note that we use LP for two distinct purposes: 1) before data collection, to select the module from which to draw the next goal to target in the environment; 2) before training, to select the substitute module descriptor (module replay). Recall that, once transitions are sampled from the replay buffer, they can be modified (replayed) by substituting the original module descriptor and target by new ones. The substitute module is the one the agent is going to learn about. When replaying a particular module more than others, the agent allocates more learning resources to that module. While the use of LP for module selection is not new (Forestier & Oudeyer, 2016b), CURIOUS is the first algorithm to consider its use for cross-module goal replay.

Module and Goal Selection

Before interacting with the environment, the agent selects the next goal to target by sampling a module from $\mathcal{M}$ using p_{LP} and sampling the target uniformly from the corresponding goal space $g_i \sim \mathcal{G}_{M_i}$.

Cross-Module and Cross-Goal Learning

In an example with three modules, an agent computes $p_{\text{LP}} = [0.6, 0.2, 0.2]$. The agent uses these probabilities to guide learning towards modules with high LP. Suppose the minibatch size is N_{mb}. The agent samples $\lfloor N_{\text{mb}} \times 0.6 \rfloor$ transitions relevant to module 1, $\lfloor N_{\text{mb}} \times 0.2 \rfloor$ transitions relevant to module 2, etc. A transition is said *relevant to module* M_i (e.g. Push module) if it comes from an episode in which the corresponding outcome has changed (e.g. block position). This sampling bias towards "eventful" transitions is similar to the *energy-based prioritization* approach of Zhao & Tresp (2018). In this minibatch, every transition has been sampled to train on a specific module m^*, even though it might have targeted another module m. To perform this cross-module learning, we simply substitute the initial module m with the selected module m^*. Goal substitution is then performed using hindsight: the target g of a transition is sometimes ($p = 0.8$) replaced by an outcome reached later in the same episode g^* (Andrychowicz et al., 2017).

Internal Reward

After module and target encodings have been substituted, the agent computes an internal reward for each transition using a reward function parameterized by the new m^* and goal g^*. Thus it answers: *What would have been my reward for experiencing this transition if I were aiming at that imagined goal from that imagined module?* The reward is non-negative (0) when the outcome satisfies the constraints described by the imagined module m^* relative to the imagined target g^*; negative otherwise (-1). In a reaching module, for instance, a positive reward is generated when the Euclidean distance between the 3D target (goal) and the gripper (outcome) falls below a precision parameter ϵ_{reach}.

3.2.2 The CURIOUS Architecture

We give a schematic view of CURIOUS in Figure 3.4 and its pseudo-code in Algorithm 2. The main steps are:

1. **Module and goal selection.** The agent selects module M_i and target g_i for the next rollout (blue). It respectively samples them from the set of modules $\mathcal{M}$ using p_{LP} (purple) and uniformly from the corresponding goal space $\mathcal{G}_{M_i}$.

2. **Data collection.** The agent interacts with the environment (grey) using its current M-UVFA policy (yellow). It collects transitions and stores them in memory (purple).

3. **LP update.** If this was a self-evaluation rollout, the agent updates its subjective measures of competence, LP and p_{LP} given the new outcome (success or failure, purple).

4. **Module and target substitution.** The agent selects the modules and targets to train on. To update the policy and critic, the algorithm first samples a minibatch from the replay buffers (purple) using p_{LP} and implements module and target substitutions to perform cross-module, cross-goal learning (red).

5. **Internal reward.** The agent computes its reward r for each transition using R_{m^*, g^*} parameterized by the substitute module m^* and target g^* (red).

6. **RL updates.** The agent updates its policy and value function with DDPG using the modified minibatch (red).

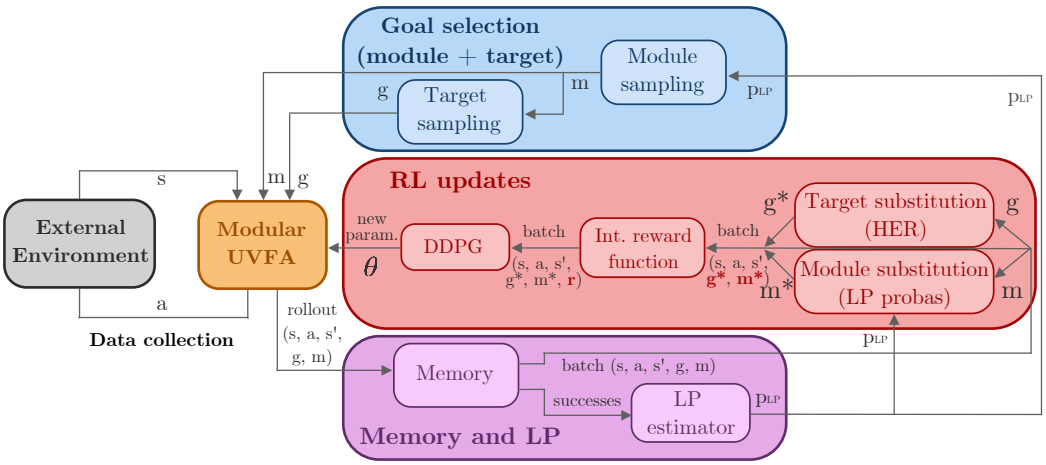


Figure 3.4: Schematic view of CURIOUS. Colored modules compose the CURIOUS architecture. The agent collects data by interacting with its environment (grey-orange), stores it in memory and updates learning progress estimates (purple), then updates its action (red) and goal selection (blue) policies.

Algorithm 2 The CURIOUS algorithm.

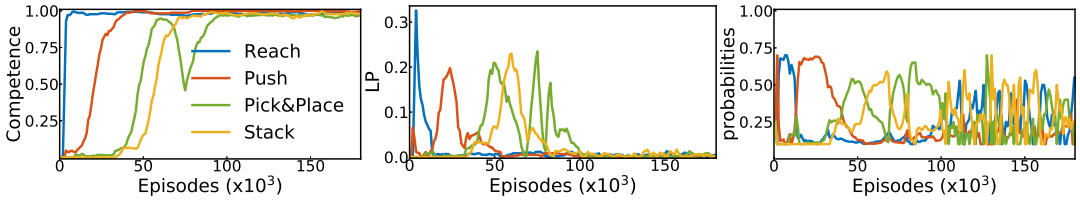
Input: env, set of N modules $\mathcal{M}$, goal spaces $\mathcal{G}_{1:N}$, internal reward $\mathcal{R}$, self evaluation probability p_{eval}.
Initialize: policy π, empty buffer, uniform module probabilities p_{LP}.
repeat
  eval $\leftarrow$ random() $< p_{\text{eval}}$ ▷ self-evaluation Boolean
  ▷ Goal sampling
  $g, m \leftarrow$ **GoalSampling**(eval) ▷ $M_i \sim p_{\text{LP}}$ (or uniform if eval), $g \sim \mathcal{U}(\mathcal{G}_{M_i})$
  ▷ Interaction loop
  $\{s, a, s', g, m\}_{\text{episode}} \leftarrow$env.rollout($\pi$, g, m, eval)
  buffer.add($\{s, a, s', g, m\}_{\text{episode}}$)
  ▷ LP update
  if eval **then**
    $p_{\text{LP}} \leftarrow$ **LP_Update**(buffer)
  ▷ Replay
  $\{s, a, s', g, m\}_{\text{batch}} \leftarrow$ buffer.sample_batch()
  $\{s, a, s', g^*, m^*\}_{\text{batch}} \leftarrow$ **ModuleTargetReplay**(p_{LP}, $\{s, a, s', g, m\}_{\text{batch}}$)
  $\{s, a, s', g^*, m^*, r\}_{\text{batch}} \leftarrow \mathcal{R}(\{s, a, s', g^*, m^*\}_{\text{batch}})$
  ▷ RL update
  $\pi \leftarrow$ **RL_Updates**($\{s, a, s', g^*, m^* r\}_{\text{batch}}$)
until learning over

3.2.3 Visualizing Developmental Trajectories

This section aims at showing the inner working of CURIOUS's intrinsic motivation towards LP. We focus on a set of four achievable modules (Reach, Push, Pick and Place, and Stack).

Figure 3.5a shows the evolution of the module-dependent competence measures as subjectively perceived by a learning agent, while Figure 3.5b shows the evolution of the corresponding LP measures. Finally, Figure 3.5c shows the corresponding module selection probabilities p_{LP}, a mixture of random selection with probability ϵ and active selection proportional to LP measures with probability $1 - \epsilon$. These figures pictures successive learning phases that we can interpret as developmental phases (Oudeyer & Smith, 2016). The robot first learns how to control its gripper (M_1), then to push objects on a desired target on the table (M_2) before it learns how to place the block on a 3D target (M_3) and how to stack the two blocks (M_4). Figure 3.5b shows that LP stays small for modules that are already solved (e.g. M_1 after 10^4 episodes) or too hard to solve (e.g. M_3 and M_4 before $35 \cdot 10^3$ episodes) and increases when a module is being learned.

Traditionally, the study of learning phases is based on behavior. This is true in psychology, which can access behavior but cannot directly access internal representations (Thelen & Smith, 1996). Goal-conditioned RL papers as well, when they do provide behavioral studies, usually present the test performance of the agent on various goals as a function of time (e.g. Mankowitz et al., 2018). In our study, we have direct access to our agents' internal representations; to their subjective measures of competence and learning progress for each of the modules. It is from these internal representations that we study developmental progressions. This enables us to distinguish active curriculum



(a) (b) (c)

Figure 3.5: Visualization of a single run. a) Module-dependent subjective measures of competence for CURIOUS (1 run). b) Corresponding module-dependent subjective measures of absolute LP. c) Corresponding probabilities p_{LP} to select modules to practice or to learn about.

learning from passive experience of the environmental structure. Indeed, the structure of the environment can be such that some modules are simpler than others, leading random agents to learn them faster. On the other hand, agents performing active module selection such as CURIOUS actively guide their learning trajectories as a function of their internal representations of competence and progress.

In Figure 3.6, we present five sets of subjective competence curves and their associated learning progress. We can see that M_1 (*Reach* module) is always learned first and M_2 (*Push* module) is always learned second. However, M_3 (*Pick and Place*) and M_4 (*Stack*) can be learned in various order or even simultaneously depending on the individual learning trajectories. Once the agent has experienced a few successes on a module, LP increases and the agent focuses on it, which generates even more successes. What happened by chance in the early stages of learning leads this agent to focus first on either M_3 or M_4. Although some modules might be easier to learn first or necessary to learn others, individual experiences influence learning trajectories just as for humans (Oudeyer & Smith, 2016).

3.2.4 Resilience to Forgetting and Sensor Perturbations

During learning, agents can see their performance decrease on a previously mastered module. This can happen because they do not target it often (catastrophic forgetting), because of environmental changes (e.g. it gets windy) or because of body changes (e.g. sensor failure). Ideally, CURIOUS agents should be able to detect and react when such situations arise. This section investigates the resilience of CURIOUS to such perturbations and compares it to the M-UVFA baseline.

We first look at a run where forgetting occurs and explain how CURIOUS detects the situation and reacts. Since forgetting cannot be triggered, we emphasize a second experiment simulating a time-locked sensory failure. We present the following setup to the agent: first, it learns about a set of four modules (Reach, Push, Pick and Place for block 1, and Push for block 2). Then, we trigger a time-locked sensory perturbation ($epoch = 250$, $episode = 237.5 \times 10^3$) such that the perception of block 2 gets shifted by 0.05 (size of a block in simulation units) until the end of the run. The performance on this module suddenly drops and we compare the recoveries of CURIOUS and M-UVFA.

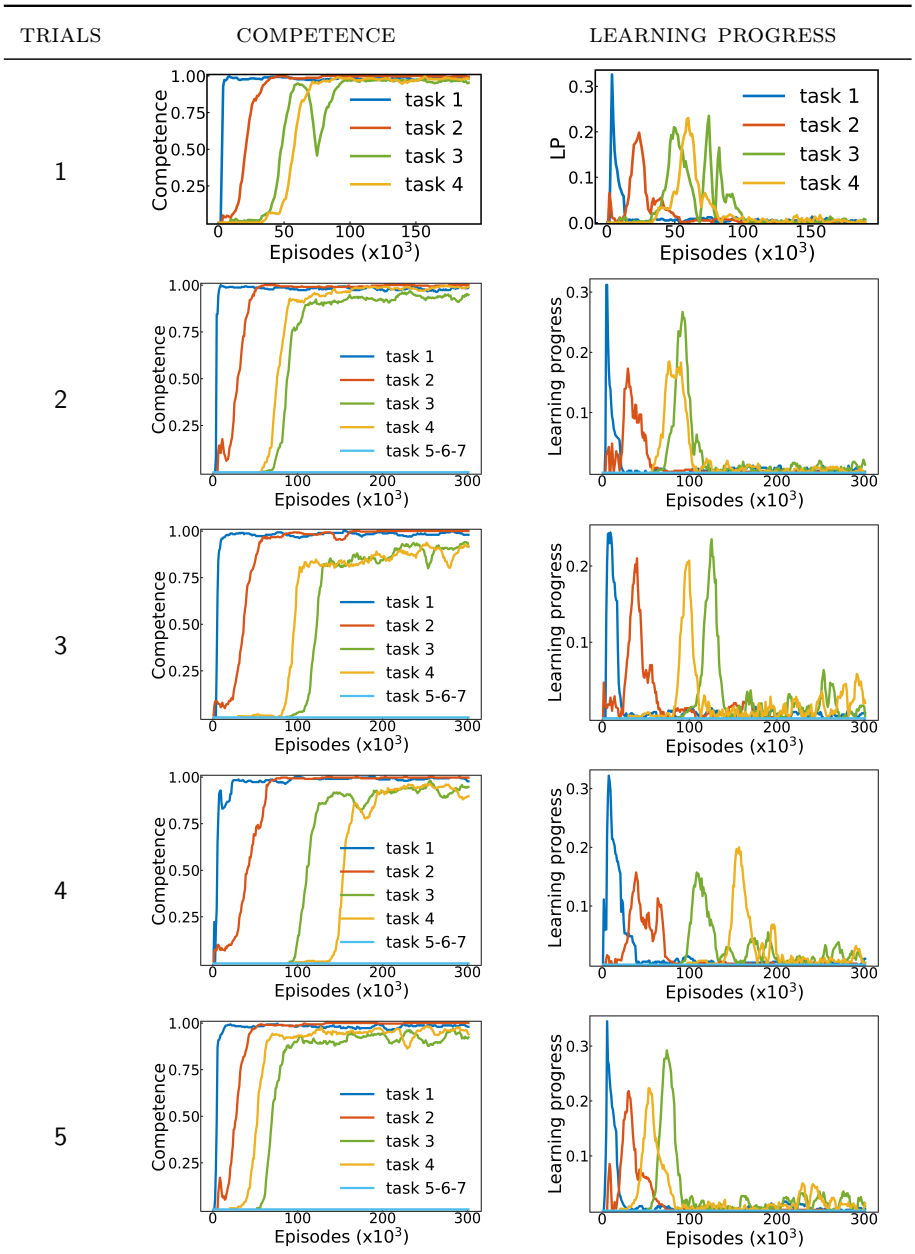


Figure 3.6: Learning Phases. Competence (left) and absolute learning progress (right) for five trials of the CURIOUS algorithm (Reach, Push, Pick and Place, Stack + 3 distracting modules).

Forgetting

Looking at Figure 3.5a, we can observe a drop in the competence on M_3 around episode $80 \cdot 10^3$. This phenomenon is called *catastrophic forgetting*: because it is trained on other modules, the network forgets about the previously mastered module M_3 without any apparent reason. The corresponding period of Figure 3.5b shows an increase in LP for M_3 which, in turn, triggers an additional focus of the agent towards that module (see the corresponding probability increase in Figure 3.5c). Using LP to bias its attention, the agent monitors its competence on the modules and can react when it forgets about a previously mastered module. This mechanism helps the agent deal with forgetting and facilitates the learning of multiple modules in parallel. To prove its efficiency, we need to compare CURIOUS to its baseline M-UVFA using a time-locked perturbation.

Sensor Perturbation

In Figure 3.7, we can observe the drop in the average success rate after the perturbation (around $240 \cdot 10^3$ episodes). This perturbation only affects one of the four modules (Push block 2) and results in a drop of about $1/4^{th}$. As described above, CURIOUS agents detect that perturbation and react by using more transitions to improve on the corresponding module. This translates into a significantly faster recovery when compared to M-UVFA. Agents recover 95% of their pre-perturbation performance in $78 \cdot 10^3$ and $43 \cdot 10^3$ episodes (random and active respectively). This corresponds to a 45% faster recovery for CURIOUS ($p < 10^{-4}$), see Figure 3.7.

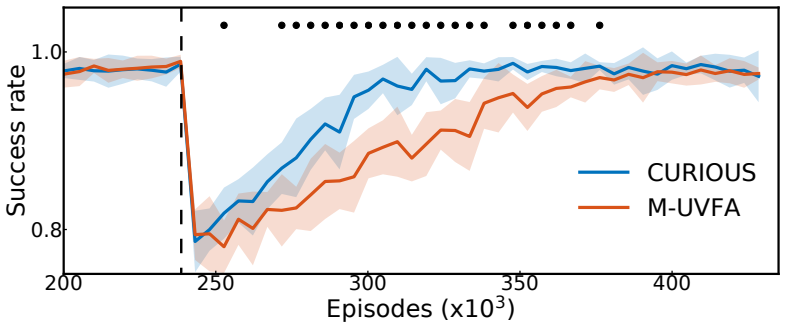


Figure 3.7: Impact of the intrinsic motivation towards LP in the context of sensory failure recovery. We plot the mean success rates over the four modules ± std over ten trials. The dashed line indicates the onset of the perturbation, while the dots indicate significance when testing CURIOUS against M-UVFA.

3.2.5 Resilience to Distracting Modules

In this section, we investigate the resilience of our learning algorithm when the number of distracting modules increases (0, 4, 7). The agent faces four achievable modules in addition to the distracting modules (Push modules relative to randomly-moving and out-of-reach blocks). For each new distracting block, the agent receives extra noisy inputs

corresponding to their movements.

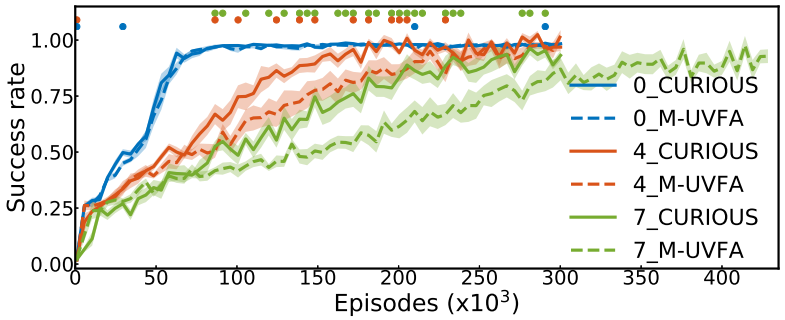


Figure 3.8: Impact of the intrinsic motivation towards LP when the number of distracting modules grows. Average success rates computed over achievable goals when the number of distracting modules increases (0, 4, 7). Mean and standard error of the mean (for visualization purpose) over ten trials are plotted. The dots indicate significance when testing CURIOUS against M-UVFA.

The number of distracting modules highly impacts the learning speed on achievable goals (Figure 3.8). In particular, M-UVFA agents do not know that these goals are impossible to achieve and waste time and resources trying to improve on them. Since these agents sample distracting modules just like others, we can expect the learning speed to be scaled by $\frac{\#achievablemodules}{\#modules}$. On the other hand, CURIOUS agents try to learn which modules are too difficult at the moment and target them less often. Note that CURIOUS agents still need to choose them sporadically to keep updated measures of their LP: they sample a random module with probability ϵ. In Figure 3.8, we see that the advantage of CURIOUS over its random counterpart increases as the number of distracting modules grows (see colored dots indicating significant differences). Although the addition of distracting modules might sound a bit ad-hoc here, it is important to note that autonomous agents in the real world would face numerous distracting modules. Humans themselves cannot reach all goals (predicting the movement of leaves on a tree, trying to walk on walls, etc.). Like humans, artificial agents need to discard them based on experience and LP.

3.2.6 Discussion

In this study, we adapt the learning progress module selection from Forestier & Oudeyer (2016b) for our first autotelic RL agent, CURIOUS. We use LP to help sample relevant goals for data collection and data exploitation. Our intrinsic motivation based on LP increased agents' robustness to distracting goals, catastrophic forgetting and sensory failures.

In a setting without any distracting module, LP does not seem to bring any significant advantage, even though some of the modules seem to be stepping stones towards others—*reach* easier than *push* easier than *pick and place* easier than *stacking*. In addition, LP-based agents also see their performance decrease as the number of distracting modules grows—although less so than their random goal selection counterparts. These two

behaviors may result from the intrinsic difficulty in adequately tuning the exploration-exploitation trade-off in goal selection. While a lack of exploration affects the quality of LP estimations and prevents agents from returning to tasks they estimated impossible, too much exploration leads agents to spend too much time on distracting tasks (too easy or too hard). In this scenario, humans leverage priors to judge whether goals will always be impossible (e.g. walking on the ceiling) or whether they need to find the suitable stepping stones first and should return later (e.g. beat your neighbor at chess).

In this work, we define the target selection policy as the uniform sampling of a pre-defined space of reachable targets. In the future, agents could learn it autonomously using automatic curriculum learning methods for continuous goal spaces. SAGG-RIAC, for example, recursively splits wide continuous goal spaces and focuses on sub-regions of high LP (Baranes & Oudeyer, 2013). On the other hand, GOAL-GAN trains a generative adversarial network on the set of visited outcomes to generate goals of intermediate difficulty. Since this study, many more approaches have emerged, see a review in Section 1.2. However, a preliminary study on the use of SAGG-RIAC for goal selection when the goal space is much larger than the reachable space (5x) seemed to show no benefit. Indeed, the initial goal only affects data collection. Data exploitation is conducted after hindsight learning, i.e. after the initial goals have been substituted by the later outcomes. In the end, the algorithm learns about substitute goals selected from experienced trajectories.

We wanted to compare CURIOUS to its POP-IMGEP counterpart MACOB but failed to train a policy to solve a simple pushing task (Forestier & Oudeyer, 2016b). The considerable variations of the initial states forced us to condition the research of a policy to solve a given goal by the initial state. Indeed, a policy achieving a particular outcome from a particular initial state will probably achieve a much different outcome when executed from a different initial state. The open-loop policies and the nearest neighbor inverse model used in MACOB could not tackle the Modular Goal Fetch Arm setup. An alternative could be to train closed-loop neural policies and to evaluate them over many rollouts. However, these rollouts might lead to a large distribution of outcomes. Representing the policy's outcome by the average outcome might not be very satisfying.

Finally, a natural extension of our work could replace our one-step goal selection policy based on MAB with a multi-step goal selection policy trained with RL. This approach is coined hierarchical RL (HRL) (Sutton et al., 1999; Precup, 2000) and has already been used with traditional UVFA architectures implementing the low-level policy (Levy et al., 2019). HRL implementations that rely on M-UVFA-based low-level policies would benefit from a more diverse action space resulting from the more diverse set of goals M-UVFAs can target.

This chapter introduced CURIOUS, the first IMGEP agent powered by deep reinforcement learning. CURIOUS demonstrates several properties of truly autonomous learning agents: 1) it can target a large diversity of goals (Section 3.1) and 2) it can organize its own learning trajectories (Section 3.2). We proposed a modular-UVFA architecture (M-UVFA) to support the first property—agents learn about multiple goals of different types in a single policy and benefit from transfer learning via weight-sharing and hindsight learning. The automatic curriculum learning mechanism targets goals associated with large absolute learning progress and helps agents focus on relevant goals and increase their robustness to distractors, forgetting, and sensory failures.

However, CURIOUS is neither *fully autotelic* nor *open-ended* as it relies on the prior knowledge of bounded modular goal representations and associated reward functions. In the next part of this manuscript (Part II), we look at how linguistic social interactions can help autonomous agents represent goals and perform efficient autotelic exploration. After presenting relevant concepts and related work (Chapter 4), we introduce and study two linguistic RL-IMGEP agents overcoming the limitations of CURIOUS (Chapters 5 and 6).

Part Summary

The first part of this manuscript introduced the framework of *rl-based intrinsically motivated goal exploration processes* (RL-IMGEP) to tackle the problem of the acquisition of open-ended repertoires of skills. The RL-IMGEP framework builds on the standard RL framework and its extensions to goal-conditioned RL and intrinsically motivated RL. It also builds on the concept of *intrinsic motivations* developed in psychology, cognitive science and developmental robotics. RL-IMGEPs are a sub-family of IMGEPs, a set of methods leveraging competence-based intrinsic motivations to design agents that learn to represent, generate, pursue and master their own goals. RL-IMGEPs leverage modern deep RL optimization techniques to overcome some of the limitations of existing population-based implementations of IMGEP.

After presenting these different concepts and computational frameworks, we turned to two empirical studies. We showed that autotelic exploration can be used as an auxiliary task to solve external hard-exploration problems (Chapter 2). Then, we introduced our first RL-IMGEP agent. CURIOUS uses DRL to learn a variety of skills involving multiple affordances and organizes its own learning trajectories by pursuing learning progress (Chapter 3). In the second part of this research, we look at how language can be used to help autotelic agents learn goal representations and structure their exploration.

Part II

Vygotskian Autotelic Reinforcement Learning Agents

The first part of this manuscript introduced the RL-IMGEP framework and studied a first algorithmic instance called CURIOUS. The main limitation of CURIOUS is its need for prior knowledge about module goal representations and associated reward functions. In other words, CURIOUS is not yet genuinely autonomous and does not demonstrate open-endedness.

Autotelic learning agents must leverage intrinsic motivations to drive the learning or goal representations and reward functions. This can take the form of *self-supervised learning*. Existing approaches mainly rely on reconstruction tasks and learn goal representations as the intermediate representations of an auto-encoder trained to encode and reconstruct visual states (Péré et al., 2018; Laversanne-Finot et al., 2018; Nair et al., 2018b; Warde-Farley et al., 2019; Cully, 2019; Pong et al., 2020). Each goal—i.e. each latent code—is nothing else than a compressed representation of a single visual state, which practically limits the abstraction of learned goal representations. One could think of other self-supervised tasks such as predicting the future from the past, predicting masked patches in visual states, predicting visual inputs from sound inputs, etc.

In this second part, we are interested in using another kind of intrinsic motivation to drive the learning of goal representations and reward functions: social motivations. Refining the concept of *situatedness* developed by nouvelle AI in the 1980s (Brooks, 1990, 1991a,b), *social situatedness* emphasizes the importance of socio-cultural environments in human development (Clark, 1998b; Brooks et al., 1999; Dautenhahn et al., 2002; Zlatev, 2001; Lindblom & Ziemke, 2003). Humans are social beings intrinsically motivated to interact and cooperate with their peers. As some argued, this tendency to interact with rich socio-cultural environments is even key to the development of our higher cognitive abilities (Brooks et al., 1999; Tomasello, 2000b; Zlatev, 2001).

These ideas build on the pioneering work of a Soviet psychologist named Vygotsky (Vygotsky, 1930, 1933, 1934). In his theory of child development, he argues that intelligence emerges through the interactions between the properties of physical and social environments and the development of biological factors. As children interact with their socio-cultural environment, they progressively internalize the help provided by caretakers. What was social cognitive help at first turns into internal psychological tools helping children to structure their cognition. Following several proposals from developmental robotics (Dautenhahn & Billard, 1999; Lindblom & Ziemke, 2003; Mirolli & Parisi, 2011), we propose to take inspiration from this line of work to improve the design of our autotelic agents. As an homage to Vygotsky's work, we coin this approach *Vygotskian autotelic reinforcement learning*. We organize this part as follows:

- **Chapter 4.** This chapter presents the supra-communicative view of language (Vygotsky, 1934; Clark, 1998a). As suggested by Vygotsky and others, language, more than a communication tool, is also a cognitive tool contributing to some of the most impressive human capacities: imagination, creativity, analogical thinking, abstract reasoning or planning (Vygotsky, 1934). Taking this point of view, we organize a review of existing works using language to enhance skill learning.
- **Chapter 5.** This chapter presents IMAGINE, the first RL-IMGEP to learn its own goal representations, to generate its own learning signals and to leverage language as a cognitive tool. IMAGINE learns goal representations and reward functions

from linguistic interactions with a social partner. After learning this mapping between structures in language and skill spaces, it uses linguistic productivity to imagine new goals by composing known ones. As they imagine goals and pursue them, agents demonstrate increased exploration abilities and learn to correct for over-generalizations.

- **Chapter 6.** This chapter introduces the *Language-Goal-Behavior* (LGB) architecture and its decoupled skill learning and language grounding phases. LGB assumes innate semantic representations as a pivot between autotelic skill learning (from semantic goals to behavior) and social language grounding (from linguistic descriptions to semantic goals). Language grounding occurs via a language-conditioned goal generator mapping each linguistic description to a set of corresponding semantic representations. This language-conditioned world model allows agents to build abstract representations on top of innate representations, to demonstrate behavioral diversity and strategy-switching behaviors.

Chapter 4

Foundations: Language as a Cognitive Tool for Humans and Artificial Agents

Ask anyone what language is about. Chances are, they will answer something along these lines: "*language is for people to communicate their thoughts to each other*." They are right of course, but research in developmental psychology and linguistics shows that language is a lot more. It significantly contributes to our most important cognitive abilities (e.g. Vygotsky, 1930, 1934; Whorf, 1956; Berk, 1994; Gentner, 1983; Rumelhart et al., 1986; Clark et al., 1998; Dautenhahn & Billard, 1999; Carruthers, 2002; Lindblom & Ziemke, 2003; Spelke, 2003; Parisi, 2010; Mirolli & Parisi, 2011; Bergen, 2012).

This chapter draws inspirations from the communicative and cognitive uses of language in humans and surveys existing works attempting to transfer them to artificial agents. We start by providing a short account of some of Vygotsky's ideas in Section 4.1. Autotelic agents, human or artificial, benefit from the communicative aspects of language. Humans use language to describe events, to instruct and guide each other. In the same way, linguistic artificial agents can leverage human-generated knowledge databases, direct advice, descriptions and instructions (Luketina et al., 2019) (Section 4.2). More than a communication vehicle, language demonstrates *supra-communicative* functions (Clark et al., 1998), also called *cognitive functions* (Carruthers, 2002). Language is thought to facilitate the representation of abstractions in humans and, thus, can be used to represent abstract goals in autotelic agents (Section 4.3). Because of its compositional and recursive properties, language also enables humans to *generalize systematically*, i.e. to readily understand the meaning of entirely novel expressions and behave accordingly. Artificial agents have not yet achieved perfect systematicity, but researchers make progress every day (Section 4.4). Abstract reasoning and generalization pave the way for abstract planning in long-horizon tasks in both humans and artificial agents (Section 4.5). We finally go over two crucial cognitive uses of language in humans that have yet to be transferred to artificial agents: creative imagination and language-guided mental simulations (Section 4.6). Experiments in Chapters 5 and 6 will introduce the first embodied autotelic agents to leverage these capacities for autonomous skill learning.

4.1 Vygotsky's Theory of Child Development

In the 1920s and 1930s, Piaget and Vygotsky developed two prominent theories of cognitive development: the child as a solitary thinker (Piaget, 1926) and the child as a social thinker (Vygotsky, 1934). In Piaget's theory, children are the leaders of their own cognitive development. They spontaneously develop through qualitatively different and universal cognitive phases that are only faintly influenced by social interactions and instruction. On the other hand, Vygotsky's theory focuses on *social situatedness*[1]: human intelligence develops through social interactions with others.

Vygotsky makes a distinction between *elementary* and *higher* mental functions (Vygotsky, 1934). While elementary mental functions directly map stimuli to responses, higher mental functions mediate this connection by *psychological tools*. According to Vygotsky, all animals develop elementary mental functions, but only humans develop higher ones. An essential aspect of the theory is the idea that psychological tools first appear under the form of social interactions with more capable peers and are subsequently internalized by the learner:

> "*An interpersonal process is transformed into an intrapersonal one. Every function in the child's cultural development appears twice: first, on the social level, and later, on the individual level; first, between people (interpsychological), and then inside the child (intrapsychological).*" (chap. 4 Vygotsky, 1934)

Humans use technical tools to manipulate their environment and psychological tools to direct, control and organize their thoughts and behaviors. Children, for example, are first taught to count on their fingers, then internalize this socially learned skill and count *in their head*. As they learn to do so, the initial social process of counting turns into an internal psychological tool they can use to shape their thoughts. Language is perhaps the most powerful of these psychological tools. In Vygotsky's theory, language starts as a social communicative process. Parents interact with the child, describe, explain and play with him. This social speech then transforms into what Piaget called *egocentric speech*, an outer speech of the child for himself. As it develops, egocentric speech becomes more goal-oriented and provides cognitive aids of the type caretakers would provide (Vygotsky, 1934; Berk, 1994). Progressively, it becomes more efficient and abbreviated, less vocalized, until it is entirely internalized by the child and becomes *inner speech* (Vygotsky, 1934). For the rest of their lives, humans use inner speech to organize their thoughts, maintain memory, evaluate alternatives and plan. When a task gets hard, adults sometimes reignite this egocentric speech, and talk to themselves aloud (at least I do).

Another important concept of Vygotsky's theory is the *zone of proximal development* ZPD. ZPD describes these learning situations where children solve challenges just beyond what they can do alone thanks to social interactions with more knowledgeable peers. It is in this zone that children internalize social processes. What a child can do with social help today, she can do tomorrow alone. In Vygotsky's words:

[1] The term was coined later (Dautenhahn et al., 2002).

> "*[The zone of proximal development] is the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers.*" (Vygotsky, 1934)

These social interactions can take many forms, as noted by Vygotsky and in later studies by Wood and colleagues: maintaining motivation, suggesting goals or comparisons to the goal, descriptions, explanations, demonstrations (Wood et al., 1976).

Vygotsky's body of work (more than 180 publications) underwent a 20-year ban under the Stalinist regime and only resurfaced in the 1960s, with his first translated publication in 1962 (Vygotsky, 1934). Since then, empirical and theoretical works in developmental psychology, linguistics, philosophy, cognitive robotics and artificial intelligence took inspiration from this vision, developed and reinforced it (e.g. Gentner, 1983; Rumelhart et al., 1986; Berk, 1994; Clark et al., 1998; Dautenhahn & Billard, 1999; Carruthers, 2002; Lindblom & Ziemke, 2003; Spelke, 2003; Parisi, 2010; Mirolli & Parisi, 2011; Bergen, 2012). However, to this day, most works in robotics and reinforcement learning seem to be influenced by Piaget's theory. Artificial agents learn their own concepts and behaviors from external signals. Even intrinsically motivated agents mostly learn autonomously without relying on any form of social interactions. Following a series of calls for a Vygotskian approach to developmental robotics (Dautenhahn & Billard, 1999; Zlatev, 2001; Lindblom & Ziemke, 2003; Mirolli & Parisi, 2011), we argue for a similar approach for RL. In the following sections, we take this perspective and analyze existing works under this new light.

4.2 Language to Communicate, Instruct and Advise

The developmental approach to AI aims to design a *child machine* that develops, interacts and learns in contact with humans via a process similar to children's education (Turing, 1950; Weng et al., 2001; Asada et al., 2009; Mikolov et al., 2016). In this process, humans often use language as a communicative tool to convey their intentions, instruct, guide, provide feedback or bond with children. We will see that Bruner's notion of *pragmatic frames* offers a valuable framework to think about social interactions and their implementations in artificial agents (Bruner, 1985).

Communicative Uses of Language in Humans

Following Vygotsky's work, others have argued that social interactions facilitate learning processes by offering stable structures (Bruner, 1985; Rohlfing et al., 2016; Vollmer et al., 2016). Bruner first formalized this as *pragmatic frames*: verbal or non-verbal patterns of goal-oriented behaviors that evolve over repeated interactions between learners and teachers (Bruner, 1985). As they frequently interact, learner and teacher converge towards stable pragmatic frames characterized by a *syntax* (the coordinated roles of the actors) and a *meaning* (the learning content). In Bruner's *book-reading frame*, the caretaker reads a book to the child and often asks, "*what's that?*" while pointing to

an image. Then, the learner answers and the caretaker provides feedback. This stable structure only varies in its learning content, such that the learner can quickly isolate the mapping between object and label.

Humans use a variety of pragmatic frames, most of which include linguistic interactions. In problem-solving tasks, caretakers often use language to keep children engaged, point to errors or suggest alternative approaches (Wood et al., 1976). This linguistic aid often takes the form of events descriptions and attention guidance, more than they involve direct instructions (Bornstein et al., 1992; Yoshida & Smith, 2003; Tomasello et al., 2005). Pragmatic frames also evolve as a function of children's abilities following Vygotsky's *zone of proximal development* (Vygotsky, 1934) or Bruner's *scaffolding theory* (Wood et al., 1976; Bruner, 1985). As children get older, social partners constrain learning opportunities so that children always remain optimally engaged and challenged. Caretakers especially adapt the way they talk as a function of the children's capacities. Infant-directed speech—also called *baby talk* or *motherese*—seems to promote the child's attention to language, foster caretaker–infant social interactions and provide scaffolding for language learning (e.g. more articulated vowels, better word segmentation, Golinkoff et al., 2015).

In their research, Smith and Gasser argue that the tendency of parents to describe events helps infants ground language by synchronizing sounds and actions, one of the many forms of multi-modal integration infants leverage to learn:

> "*In one study, Yoshida and Smith observed that both English-speaking and Japanese-speaking parents routinely couple action and sound when talking to young children (Yoshida & Smith, 2003). For example, one parent demonstrated a toy tape measure to their child and when pulling the tape out said, "See, you pullllllll it," elongating the word pull to match the stopping and starting of the action of pulling. This same parent, when winding the tape back in, said, "Turn it round and round and round and round and round and round," with each "round" coinciding with the start of a new cycle of turning. By tying action and sound, parents ground language in the same multimodal learning processes that undergird all of cognition, and in so doing, they capture children's attention, rhythmically pulling it to the relevant linguistic and perceptual events, and tightly binding those events together.*" (Smith & Gasser, 2005)

The Language Grounding Problem

Artificial agents can only interact with non-expert human caretakers if they understand language in the form of narratives, descriptions, advice and instructions. To this end, they must learn mappings between language and their internal representations of the world; that is, they need to ground language in experience. *Indexical reference* is the most basic approach. Here, each linguistic expression is mapped to a corresponding internal representation in a one-to-one manner, independently from other mappings (left in Figure 4.1). However, indexical reference prevents any form of generalization; a new

mapping needs to be learned for each new expression. Instead, agents should solve the *language grounding problem*, a particular case of the *symbol grounding problem* (right in Figure 4.1, Harnad, 1990; Nieder, 2009). Here, words are symbols embedded in a referential system called *syntax*. They can be recursively composed to form larger structures called *expressions* (Nieder, 2009). Grounding language is about linking both word–symbols to their objects—i.e. internal representations—and syntaxes to compositional structures of internal representations. Once language is grounded, agents can systematically generalize to any new expression by mapping the words to their representations and the syntax to the corresponding structure in representation space.

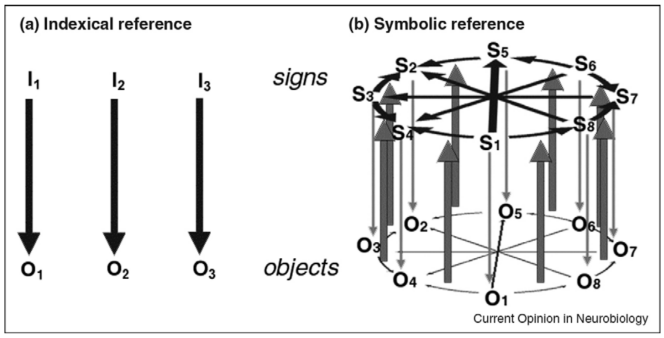


Figure 4.1: Graphic illustration of the difference between indexical and symbolic reference. Grounding language is about mapping linguistic symbols (S_i) and their structures (bold horizontal arrows in symbol space) to corresponding objects (O_i) and their structures (light horizontal arrows in representation space). Mappings are represented by vertical arrows. From (Nieder, 2009).

Pragmatic Frames in Reinforcement Learning

In 2016, a group of researchers reemployed the notion of pragmatic frames to formalize human-robot interactions (Vollmer et al., 2016; Rohlfing et al., 2016). Analyzing the literature from this perspective, they note that most existing works focus on a single hard-coded pragmatic frame, thereby constrain social interactions. In contrast, humans use multiple flexible pragmatic frames that they co-negotiate and adapt as a function of the learner's capabilities (Bruner, 1985).

The notion of *pragmatic frame* is never explicitly mentioned in deep reinforcement learning research but is still helpful to understand teacher–learner dynamics. A recent review of language-informed DRL approaches notes that the community primarily focuses on a pragmatic frames called *instruction-following* Luketina et al. (2019). In this frame, teachers provide linguistic instructions and reinforcement feedback to the learners.

In the AI literature, *instruction-following* can refer to 1) the execution of actions sequences to solve a linguistic goal called *instruction* (Oh et al., 2017; Chaplot et al., 2018; Hermann et al., 2017; Bahdanau et al., 2019b,a) or 2) the execution of a sequence of high-level actions called instructions (MacMahon et al., 2006; Kollar et al., 2010; Mei et al., 2016; Bisk et al., 2016; Misra et al., 2017; Fu et al., 2019). Early approaches focused on the second setting and relied on pre-trained linguistic modules and extensive domain

knowledge. Some approaches learned parsers to map instruction sequences to a formal internal language used to control the agent (MacMahon et al., 2006; Chen & Mooney, 2011). Others mapped part of the instructions to their corresponding objects or actions in hard-coded models of the environment (Kollar et al., 2010; Tellex et al., 2011). Recent advances in deep learning and the access to more extensive computational resources recently made it possible to train artificial agents to learn to execute instructions from experience without any domain knowledge (Misra et al., 2017; Oh et al., 2017; Hermann et al., 2017; Chaplot et al., 2018; Bahdanau et al., 2019b,a). Although most approaches use templated language to cope with the increased demand in interaction data, some works leverage pre-trained language models (Hill et al., 2020b; Lynch & Sermanet, 2020) or demonstrations (Abramson et al., 2020; Zhou & Small, 2020; Chen et al., 2021) to handle human-generated instructions.

Besides, a few works look at other types of pragmatic frames. In a first type, agents receive linguistic help or *advice* Kuhlmann et al. (2004). The advice can be corrections of the agent's behavior (Co-Reyes et al., 2019), descriptions of the particular dynamics of the environment (Zhong et al., 2020) or indications of intermediate sources of rewards used for reward shaping in hard-exploration tasks (e.g. in Montezuma's Revenge: Goyal et al., 2019).

Another type of pragmatic frame considers interactions involving labeled demonstrations. Here, the teacher provides a sensorimotor demonstration and a linguistic description of the trajectory. Instead of grounding instructions in behavior via the policy as in instruction-following setups, agents ground language into reward functions. Agents learn to tell when a trajectory and a description match. Grounding language in behavior is about solving both language understanding and control simultaneously. On the other hand, grounding language in rewards is about solving language understanding first; learning representations of the instructions without learning to solve them. In a navigation task, grounding "*go to the fruit bowl*" in rewards might just be about detecting a fruit bowl in the visual field (example from Fu et al., 2019). It is much easier and is expected to generalize much better than grounding in behavior—i.e. learning to find the fruit bowl in any environment. Given new instructions and new environments, zero-shot generalization of the policy is not required anymore: agents can rely on the zero-shot generalization of their reward function to further train their policy in new contexts (Fu et al., 2019, and our study in Chapter 5). Grounding language in rewards can be achieved via *inverse reinforcement learning* mechanisms, the set of methods used to infer reward functions from experience (Ziebart et al., 2008; Ho & Ermon, 2016). Existing approaches usually leverage a dataset of expert demonstrations and corresponding instructions (MacGlashan et al., 2015; Bahdanau et al., 2019a; Zhou & Small, 2020) and sometimes require known environment dynamics (Fu et al., 2019).

4.3 Language to Represent Abstract Goals

Language is thought to facilitate *abstraction*, the ability to "*decrease the specificity—and thereby increase the scope—of a concept*" (Gentner & Hoyos, 2017). How does this work in humans? How can we use it in artificial agents?

Language for Abstraction in Humans

Words are invitations to form categories (Waxman & Markow, 1995). When they hear the same word in various contexts, humans are invited to compare situations and find similarities and differences. This orients their attention towards specific features and speeds up the process of category learning in both children and adults (Waxman & Markow, 1995; Yoshida & Smith, 2003). How could that work in practice? In their call for a Vygotskian cognitive robotics, Mirolli and Parisi offer two explanations (Mirolli & Parisi, 2011). First, language guides agents to focus on the specific aspects of perception relevant for categorization:

> "*Hearing the same linguistic stimulus, let's say the word 'red', when perceiving red cars, red apples and red flowers facilitates—or may even induce—the acknowledgement that all those different stimuli have something in common, namely the red colour. In neural network terms, this means that the occurrence of the same activation pattern in the acoustic input units—namely the pattern that corresponds to the sound 'red'—increases the similarity of the internal representations of all red stimuli, and this in turn can help—or induce—the network itself to learn that all those stimuli belong, to some extent, to the same category, namely that of red things.*" (Mirolli & Parisi, 2011)

Their second explanation insists on the link between categorization and behavior: two sets of stimuli are categorized differently if they lead to different behaviors. Based on this insight, they advance an exciting hypothesis: if humans are so good at categorizing things and forming abstractions, it is because they feel the urge to talk. In this view, talking is the downstream behavior and humans refine their categorizations to produce the correct labels and descriptions to succeed in social communication (Mirolli & Parisi, 2011). As their categorization skills develop, humans refine their perceptions and notice new similarities and differences, leading to new concepts then propagated to others through social interactions. Recursively, language itself becomes a sensory input for categorizing higher-level concepts. This leads to the vast diversity of abstractions developed by humans. In computational terms, making the same linguistic action downstream—e.g. producing the word 'red'—pushes the representations of the related inputs (red car, red apple) closer to each other, decreasing their specificity.

Just as words invite the formation of categories, relational language facilitates *relational abstractions* by offering invitations to compare structures. Analogies, for example, prompt the listener to perform a structural alignment between two relational patterns, to observe their similarities and differences, extract relational abstractions and transfer them to other problems (Gentner, 1983, 2016; Gentner & Hoyos, 2017). With words, the continuous world can be structured into mental entities, symbols which, when composed and manipulated, enable reasoning and give rise to the incredible expressiveness and flexibility of human thoughts (Whitehead, 1927).

According to the developmentalist Spelke, language is also essential to coordinate domain-specific modules. In her *core knowledge theory*, she argues that the specific features

of human cognition are built on a limited set of early-developing *core knowledge systems* representing 1) objects and their interactions; 2) agents and their goal-directed actions; 3) numbers and basic arithmetic; 4) space and geometry; 5) social partners and social group members (see reviews in Spelke, 2003; Spelke & Kinzler, 2007). These core systems alone, however, fall short of explaining uniquely human capacities. They are domain-specific, independent from each other and mostly shared with non-human animals. Spelke argues that language, because of its combinatorial properties, is the human-specific capacity linking these different core representations into general cross-domain representations (Spelke, 2003). For instance, language can combine object representations (e.g. blue cupboard) with spatial representations (e.g. left of) and enable humans to understand spatial indications like *left of the blue cupboard*, whereas non-human animals and pre-verbal infants cannot (Hermer-Vazquez, 2001).

Language for Abstract Goals in Artificial Agents

Goal-directed agents can use the same properties to formulate abstract goal specifications. Traditionally, goal-conditioned RL approaches formulate concrete goals as target states (e.g. an image) or target coordinates in a sub-state space (e.g. block coordinates in manipulation tasks, agent coordinates in navigation tasks). Language, on the other hand, can express *goal regions* in *abstract representation spaces*. In previous works, language as been use to express abstract goals such as "*go to a circle*" (Janner et al., 2018), relational goals "*sort objects by size*" (Jiang et al., 2019), "*put the cylinder in the drawer*" (Lynch & Sermanet, 2020), sequential goals "*open the yellow door after opening a purple door*" (Chevalier-Boisvert et al., 2019) or even learning goals "*is the ghargh edible?*" (Yuan et al., 2019).

In all these examples, linguistic goals act as sensory cues shaping multi-modal representations of both the goal and the state to perform the appropriate behavior downstream—reaching the goal. Targeting red objects, agents learn to focus on the red aspect of objects and abstract away irrelevant features (e.g. shape) to facilitate mappings between the representations of red objects and the shared behavior (reaching the object). How to interpret linguistic goals and shape state representations for efficient control becomes a crucial design choice. Recent methods propose to convert the linguistic goal into attentional vectors. In the *gated-attention* mechanism, state features are linearly scaled by language-conditioned attention coefficients $\varphi(z_g)$: input $= s \odot \varphi(z_g)$, where $\odot$ is the term-by-term product (Chaplot et al., 2018). The *feature-wise linear modulation* (FILM) approach generalizes this principle to affine transformations: input $= s \odot \varphi(z_g) + \psi(z_g)$ (Perez et al., 2018). Neuro-symbolic approaches parse the instructions to compose neural modules implementing the corresponding function (e.g. reward function, policy, Andreas et al., 2016; Mao et al., 2019; Bahdanau et al., 2019a).

As an alternative to language, one could pre-define a set of abstract goals and encode each of them with a particular one-hot embedding (Mankowitz et al., 2018; Chan et al., 2019; Ecoffet et al., 2021). For instance, agents could be rewarded for grasping any red when they target the goal [0, 0, 1]. The experimenter represents the abstract goal in her own language (e.g. *grasp any red block*) and hard-codes a corresponding reward function ($r = 1$ when agents grasp any red block) but only instructs agents with the corresponding one-hot embedding [0, 0, 1]. If we can encode abstract goals without language, why should

we use language at all? The answer is: *because it facilitates generalization.* Here, the one-hot embedding forces agents to perform *indexical references*—i.e. mapping instructions to behaviors in a one-to-one manner. On the other hand, language allows *language grounding*—agents map words to internal representations and syntax to structures in representation space such that they can readily generalize to new expressions (see a discussion in Section 4.2).

4.4 Language for Systematic Generalization

Generalization refers to an agent's ability to perform well in novel tasks that it did not encounter in the past. Humans are particularly good at it. They can generalize from a few examples in a wide variety of contexts. Language helps humans and artificial agents achieve generalization when the structure of language—its syntax—is aligned to the structure of the physical world.

Compositionality and Systematicity

Languages are made of "*meaningful expressions built up from other meaningful expressions*" (Szabó, 2020). Underlying this definition is the notion of *compositionality*:

> "*The meaning of a complex expression is determined by its structure and the meanings of its constituents.*" (Szabó, 2020)

Although the strict compositionality of natural languages is still debated (Szabó, 2020), one can at least acknowledge its *partial compositionality.* Only partial-to-complete compositionality can explain humans' ability to learn languages, readily understand new utterances and productively generate creative expressions (Chomsky, 1957; Fodor & Pylyshyn, 1988; Szabó, 2020). Chomsky illustrated the productive property of language with an original example "*colorless green ideas sleep furiously*" (Chomsky, 1957). Here, Chomsky leverages linguistic productivity to generate this original expression. Although you never read this sentence before, you might still be able to represent aspects of what it could mean.

Systematicity is the property of agents for which "*the ability to entertain a given thought implies the ability to entertain thoughts with semantically related contents*" (Fodor & Pylyshyn, 1988). Cummins might offer a more testable definition:

> "*A system is said to exhibit systematicity if, whenever it can process a sentence s, it can process systematic variants of s, where 'systematic variation' is understood in terms of permuting constituents or (more strongly) substituting constituents of the same grammatical category.*" (Cummins, 1996)

Humans are at least partially systematic: when someone understands *"Eric gives a cookie to the child,"* he automatically understands *"the child gives a cookie to Eric."* This

systematic generalization—also called *combinatorial generalization* (Battaglia et al., 2018)—is powerful because understanding a few concepts combined in a few ways automatically opens the door for understanding a large number of novel combinations of these concepts.

Systematic Generalization in Artificial Agents

Can artificial agents *learn* systematicity from the experience of a recursive and compositional language? Initial opponents to connectionism argued that only classical architectures could achieve systematicity (Fodor & Pylyshyn, 1988; Marcus, 1998). Recent empirical works mostly support this stance in seq2seq models (Lake et al., 2017), visual question answering models (Bahdanau et al., 2019b) and RL models (Hill et al., 2019).

Although perfect systematicity still seems out of reach, current artificial agents already demonstrate *partial systematicity*. One can evaluate systematicity by measuring the generalization performance of agents on linguistic goals obtained by systematic recombinations of training goals. If *"visit X"* and *"push Y"* are in the set of training goals, then systematic generalization can be measured as the performance on *"visit Y"* and *"push X"* goals. More generally, benchmarks that test systematic generalization should maximize the compound divergence (distance between sentences) and minimize the atom divergence (same vocabulary) between training and testing sets (Keysers et al., 2020). Above-chance performance on recombined goals is a marker of partial systematicity. This property can be obtained with non-linguistic categorical encodings by adding an extra learning loss favoring it (e.g. $\|\phi(\text{push } X), \phi(\text{push } Y)\| = \|\phi(\text{visit } X), \phi(\text{visit } Y)\|$, where ϕ is the language representation function and $\|\cdot, \cdot\|$ a distance metric Oh et al. (2017)). Language-conditioned RL agents also demonstrate partial systematic generalization (e.g. Hermann et al., 2017; Chaplot et al., 2018; Chan et al., 2019; Jiang et al., 2019; Zhong et al., 2020; Chevalier-Boisvert et al., 2019; Cideron et al., 2020b). A thorough empirical study helped shed some light on these questions (Hill et al., 2019). If neural networks do not yet offer perfect systematicity, some design choices positively impact performance: 1) training on a large distribution of objects and words; 2) taking an egocentric point of view so that agents experience objects in isolation from others and 3) increasing the diversity of perceptual experience.

Recent contributions re-introduced symbolic computations to neural approaches to implement inductive biases favoring systematic generalization (Andreas et al., 2016; Battaglia et al., 2018; Mao et al., 2019). In *visual question answering* setup, Andreas and colleagues use a pre-trained parser to translate questions into a composition of neural modules trained end-to-end to answer questions from images (Andreas et al., 2016). Differentiable parsers can replace pre-trained ones to retain complete question-to-answer end-to-end training (Mao et al., 2019). However, thorough empirical studies (Bahdanau et al., 2019b; Ding et al., 2020) did not find these neuro-symbolic approaches to generalize better than the best non-symbolic approaches.

4.5 Language for Long-Horizon Tasks

The two previous sections describe the use of language to represent abstractions and generalize systematically. These two properties underlie a third one: abstract planning in

long-horizon tasks.

Language for Planning and Task Solving in Humans

Piaget first described that two-to-seven-year old children often use egocentric speech to describe their ongoing activities and organize themselves, but thought this was a sign of cognitive immaturity (Piaget, 1926). Vygotsky, Berk and others investigated further and concluded that egocentric speech helps children control their behavior and master new skills (see a review and discussion in Berk, 1994). Egocentric speech develops similarly in all children, often involves self-guiding comments to orient their behavior and is more frequent in challenging situations, especially when no caretaker can help (Vygotsky, 1934; Berk, 1994). More recent studies show that children who cannot yet formulate sentences like "*at the left of the blue wall*" show lower spatial orientation capacities than children who can (Hermer-Vazquez, 2001). Interfering with adults' inner speech by asking them to repeat sentences also hinders their ability to orient spatially (Hermer-Vazquez, 2001). These studies show that the properties of abstraction, compositionality and recursivity of language enable humans to construct plans and cognitive self-aids in challenging tasks. Language helps us segment our continuous experience in semantic chunks that can then be composed recursively to form plans.

Language for Long-Horizon Control in Artificial Agents

How would you go about planning a trip to visit your friend? Would you think about every motor action from your couch to his door? Probably not. Humans plan with temporal abstractions, usually based on language: I would first book a train, pack a suitcase, wake up early that day, etc. Hierarchical RL (HRL) is the framework that formulates such temporal abstractions for RL (Dayan & Hinton, 1993; Parr & Russell, 1998; Sutton et al., 1998, 1999; Precup, 2000; Barto & Mahadevan, 2003). A high-level policy generates sub-goals to a low-level policy generating actions. These temporal abstractions—sometimes called *options* (Sutton et al., 1999; Precup, 2000)—enable agents to solve long-horizon tasks. This is due to a more effective exploration strategy. Instead of exploring the state space in a temporally unstructured manner, HRL algorithms explore with sequences of temporally structured sub-goal-directed explorations of outcome spaces.

Language can be used to encode abstract sub-goals that are reminiscent of our plans generation. One approach is to decouple the training of the low-level and high-level policies. The low-level policy is trained to perform sequences of actions to execute linguistic instructions. The high-level policy is trained to execute high-level goals by generating sequences of linguistic sub-goals for the pre-trained low-level policy. The high-level policy either selects from a fixed set of linguistic sub-goals (Jiang et al., 2019; Hu et al., 2019) or can be trained to generate them through imitation learning from human-generated datasets (Chen et al., 2021). Such agents benefit from abstract sub-goal representations and leverage systematic generalization for the low-level policy (Jiang et al., 2019; Hu et al., 2019; Chen et al., 2021).

Text-based environments take a slightly different approach (Côté et al., 2018). In these setups, algorithms train a high-level policy to perform sequences of instructions

from linguistic observations and assume a low-level oracle policy (Narasimhan et al., 2015; Côté et al., 2018; Das et al., 2018; Yuan et al., 2019; Ammanabrolu & Hausknecht, 2020; Madotto et al., 2020). This allows circumventing the problems of sensorimotor interactions (assumed perfect) to focus on other learning problems such as long-horizon exploration, sparse rewards, combinatorially large action spaces or knowledge representation in partially observable worlds. One can interpret text worlds as linguistic world models that agents explore to plan for sequences of sub-goals to be performed in an aligned physical world. This idea is implemented in *AlfWorld*, a learning environment aligning a photo-realistic sensorimotor world with a text world (Shridhar et al., 2021). The high-level policy is first trained in the text world (linguistic world model), then transferred to the sensorimotor world to provide sub-goals to a low-level sensorimotor policy.

In fact, language does not even need to condition low-level policies explicitly. Andreas and colleagues align environments, high-level goal and policy sketches: a sequence of symbolic steps towards the high-level goal (Andreas et al., 2017). When targeting a high-level goal, the agent executes a sub-policy for each step of the policy sketch, one after the other. Sub-policies are not pre-trained to perform their symbolic steps. Instead, all sub-policies are tied together according to the policy sketch and learned end-to-end. Here, the temporal structure of the policy sketch is implicitly transferred to the behavioral space by a simple alignment of sketches and behaviors.

4.6 Language for Creativity and Mental Simulations

The previous sections covered the use of language as a communicative tool (Section 4.2) and as a cognitive tool for abstraction, generalization and long-horizon control (Sections 4.3, 4.4 and 4.5). This section presents two other fundamental cognitive functions of language: creative imagination and semantic mental simulations. We argue that the capacity to imagine, generate and simulate future possibilities could significantly augment the performance of autotelic agents in their acquisition of skills repertoires. However, these abilities have yet to be transferred to embodied artificial agents.

Language for Imagination and Creativity

The remarkable ability of humans to generate new ideas is the source of our incredible success in the animal kingdom. Although our species, the *homo sapiens*, evolved about 200,000 years ago, our oldest imaginative artifacts (e.g. figurative arts, elaborate burials, first dwellings) seem to be only 70,000 years old (Harari, 2014; Vyshedskiy, 2019). This sudden *cognitive revolution* is thought to result from the onset of *recursive language* (Goldberg, 1999; Hoffmann, 2020).

Simonton defined creativity as *originality times appropriateness* (Simonton, 2012). If joining a blues jam session by randomly hitting a cow bell might be original, it is hardly appropriate and will not be called creative. On the other hand, playing one whole note per bar might be appropriate but completely boring. Only the musician who can follow the twelve-bar blues form (appropriate), improvise within it (creative) and interact with other musicians (appropriate) will be called *creative*. How can language be creative in this sense? In his theory of generative grammar, Chomsky argued that linguistic creativity

directly emerges from *linguistic productivity* (Chomsky, 1957). If words and ideas are like lego blocks, we can combine them recursively in infinite ways to form an infinite space of sentences and thoughts. Grammatical rules ensure appropriateness, while new combinations offer originality.

Recently, Sampson argued that the computational approach of Chomsky prevents any meaningful originality because new combinations only result from the application of known rules—assumed innate—thus lack any form of novelty or surprise (Sampson, 2016). He called this type of creativity the *(F)ixed-creativity*: the recombination of components in novel ways, following known rules. He called true creativity *(E)xtending-creativity*; when creators break the existing rules and add new dimensions to their creative exploration. While finding an exciting new motif in the major blues scale is F-creative, the first rapid chord and key changes that emerged from Charlie Parker's solo in December 1939 were truly E-creative.[2]

However, language can still be E-creative. Usage-based theories of language reject the Chomskian vision of the innate and universal generative grammar. Construction grammar theories (CG), for example, argue that all levels of grammatical descriptions—morphemes, words, idioms—can be understood as pairing between forms and meanings called *constructions* (Fillmore, 1985; Goldberg, 1995, 2003, 2006; Tomasello, 2000a; Steels & Van Trijp, 2011; Hoffmann, 2020). CG theories see language as an evolving social construct where constructions are learned and created through social interactions. Because constructions are a social construct, anyone can create new ones (originality) which might or might not be retained by the community (appropriateness). New expressions built on invented constructions are a true form of linguistic E-creativity (Steels, 2012; Van Eecke & Beuls, 2018; Turner, 2018; Bergs, 2019; Hoffmann, 2020).

Linguistic World Models

Recursive language and its creative potential also seem to impact other imagination abilities in humans. In neurobiology, the conscious, purposeful process of synthesizing novel mental images from two or more objects stored in memory is called *prefrontal synthesis* (PFS). Recent studies argue that the early exposure to recursive language is a precondition to the emergence of PFS (Vyshedskiy, 2019). Indeed, children born deaf with no access to recursive sign language and Romanian children left on their own in Ceausescu's orphanages—among others—were shown to lack PFS abilities and failed to acquire abstract compositional thinking even after intensive language therapy (Vyshedskiy, 2019).

Mental simulation is an essential characteristic of human cognition. It enables us to envision future possibilities and develop plans towards them (Kahneman & Tversky, 1981). If I have a causal model of how the world works, then I can use it to simulate

[2] "*I was jamming in a chili house on Seventh Avenue between 139th and 140th. It was December 1939. Now I'd been getting bored with the stereotyped changes that were being used all the time at the time, and I kept thinking there's bound to be something else. I could hear it sometimes but I couldn't play it... Well, that night I was working over 'Cherokee' and, as I did, I found that by using the higher intervals of a chord as a melody line and backing them with appropriately related changes, I could play the thing I'd been hearing. I came alive.*" Charlie Parker (Giddins, 2000).

counterfactuals, a variety of possible scenarios. As I do so, I list possible future scenarios, grow my set of options and thereby augment my control of the future. In fact, this capacity is similar to the one of imagination. Planning for future possibilities and imagining crazy scenarios rely on the same processes but only vary in the probability to see the scenarios really occurring, i.e. vary on the *breadth of the search* (Gopnik, 2009).

Mental simulations seem to be easily triggered by language. The embodied simulation hypothesis indeed argues that humans have rich, multi-sensory representations prompted by language. If you read the sentence "*he saw a pink elephant in the garden*," chances are you are visualizing a pink elephant. More generally, understanding language seems to involve parts of the brain that would be active if you were in the situation described by the sentence. Reading about pink elephants? Your visual cortex activates. Reading about someone cutting a tree? Your motor cortex activates. This was even shown to work for metaphorical uses of words (Bergen, 2012).

4.7 Chapter Summary

In this chapter, we have seen that language, more than a communication tool, is also a cognitive tool. Humans use it all the time to represent abstract knowledge, generalize, plan, invent new ideas or simulate future possibilities. If it is so helpful, it is because we align it to the real world and, doing so, project its structure and compositional properties onto our continuous physical world.

The approaches we surveyed can be seen as the first steps towards a more ambitious goal—*artificial agents demonstrating a rich linguistic mental life*. Like humans, autotelic agents must be able to describe their world, leverage linguistic productivity, generate new sentences and goals from known ones or project linguistic representations into visual, auditory and behavioral simulations. Linguistic productivity can drive pretend play, the imagination of creative made-up goals for the agent to practice its problem resolution skills. Only agents that conduct such an intensive alignment between language and the physical world can project linguistic structures onto their sensorimotor experience and learn to recognize the building blocks that will help them plan, compose, generalize and create.

In line with propositions from the field of developmental robotics (Dautenhahn, 1995; Lindblom & Ziemke, 2003; Mirolli & Parisi, 2011), we advocate for *Vygotskian autotelic agents*. Whereas standard language-conditioned RL approaches only use language to communicate instructions, Vygotskian autotelic agents align language and sensorimotor interactions to build structured world models. The following two chapters (5 and 6) make the first steps in that direction. In Chapter 5, the IMAGINE architecture first learns language by interacting with a linguistic social partner, then uses it as a cognitive tool to implement creative goal imagination. In Chapter 6, social interactions are used to train a language-conditioned goal generator to simulate a diversity of possible futures from linguistic descriptions.

Chapter 5

Language as a Cognitive Tool to Imagine Goals: IMAGINE

The CURIOUS algorithm made the first steps towards the autonomous acquisition of repertoires of skills. CURIOUS agents target a diversity of goals involving different affordances and craft their own learning trajectories via LP-based intrinsic motivations. However, CURIOUS requires the experimenter to pre-define modular goal representations, the associated reward functions and the set of goal spaces. These constraints limit the autonomy and generality of the learning agents and restrict them to a bounded exploration—*no goal can exist outside of the pre-defined goal spaces.*

This chapter introduces IMAGINE, the first language-augmented RL-IMGEP. We use language with two objectives in mind. The first objective is to overcome the limitations of CURIOUS. IMAGINE agents interact with a linguistic social partner to learn a goal space, goal embeddings and goal-achievement functions. The advantages of expressing goals with language have been discussed extensively in Chapter 4: language facilitates interpretability and communication with humans, can represent abstract goals and enhance generalization. With IMAGINE, we also want to use language as a *cognitive tool*. Here, we give agents the ability to invent new goals by composing known ones. This simple mechanism underlies the generation of creative goals, which powers a creative exploration of the environment and enhances systematic generalization.

5.1 Motivations and Related Work

Most goal-conditioned RL and POP-IMGEP approaches require pre-defined goal spaces, goal embeddings and goal-achievement functions. This leads to three main drawbacks. First, agents cannot be considered autonomous. Second, environment-specific goal representations limit the generality of the approach. Third, the pre-definition of the goal space sets a bound on the types of behaviors the agent can acquire.

Recent approaches proposed to learn goal representations in a self-supervised way by training generative models of visual states (e.g. variational auto-encoders Péré et al., 2018; Laversanne-Finot et al., 2018; Cully, 2019; Nair et al., 2018b, 2020). Here, the high-dimensional states are embedded into compact latent codes and the learned latent space forms the goal space. Sampling a latent code from the generative model is sampling

a goal from the goal space. The goal-achievement function is based on the distance between the latent codes of the goal and the current state. These approaches are powerful because they are task-agnostic: agents learn goal representations and goal spaces autonomously and generate learning signals internally. However, there are two drawbacks. The first one concerns the level of abstraction. Because goals are states, the resulting skills can only target concrete state representations. Although humans represent some goals as sensorimotor targets, their most interesting goals are defined at higher levels of abstraction—e.g. "*being a good friend*". The second drawback concerns the limitation of such goal generation to representations *within* the distribution of already-discovered effects. Indeed, generative models simply learn the distribution of the training set (here the visited states), thus cannot generate genuinely novel targets. Moving beyond *within-distribution* goal generation, *out-of-distribution* goal generation could power creative exploration by pushing agents to experience entirely novel things. However, this challenge remains to be tackled.

In this difficult task, children leverage the properties of language to assimilate thousands of years of experience embedded in their culture in only a few years (Bruner, 1991; Tomasello, 1999). As developed in Section 4.6, the productive capacity of language can push the limits of the known or even the real (our *pink elephant example*) (Chomsky, 1957). A take-away of these ideas is that we can use the compositionality of language to productively compose new out-of-distribution goals from known ones. Imagining new creative goals for oneself is reminiscent of *pretend play*, the type of play where children generate their own problems and constraints—e.g. *"let's pretend I'm a cook"* (Vygotsky, 1933; Singer & Singer, 2009; Chu & Schulz, 2020a). Delving into the evolutionary accounts of animal and human play, Chu and Schulz identify this kind of play as "distinctively human" (Chu & Schulz, 2020a). As they argue, it is remarkably adaptive:

> "*We believe novel problems and goals may be critical to human cognition because problems constrain search, and narrowing the search space sufficiently to generate new hypotheses is arguably, far more than learning per se, the hard problem of cognition.*" (Chu & Schulz, 2020b)

The present study proposes to use linguistic productivity to enable agents to generate creative problems—i.e. goals—for themselves. As we will see, this capability powers adaptive improvements down the line: it drives creative exploration and enhances systematic generalization. **I**ntrinsic **M**otivations **A**nd **G**oal **IN**vention for **E**xploration (IMAGINE) is a learning architecture leveraging linguistic interactions with a descriptive social partner (SP) to explore procedurally-generated scenes and interact with objects. IMAGINE agents discover meaningful environment interactions through their own exploration (Figure 5.1a) and episode-level descriptions provided by SP (5.1b). They turn these descriptions into targetable goals (5.1c). As they receive more descriptions of their trajectories, agents learn to represent goals by jointly training a language encoder mapping language to goal embeddings and a goal-achievement reward function (5.1d). With the goal-achievement function, agents can tell whether a given scene satisfies a given goal and generate their own training signals for policy learning (ticks in 5.1d-e). More importantly, IMAGINE can invent new goals by composing known ones (5.1f). Using their internal goal-achievement function, agents can train on imagined goals autonomously.

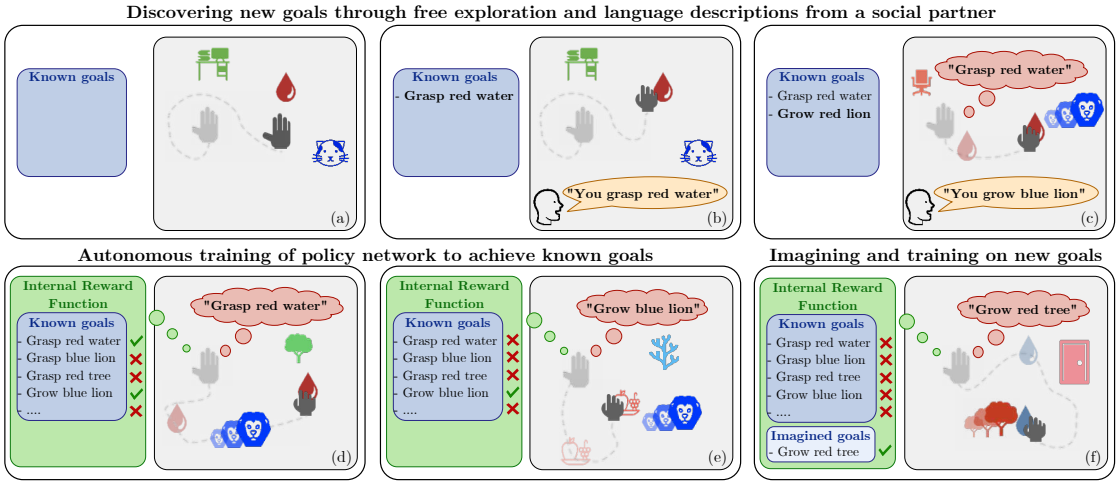


Figure 5.1: IMAGINE overview. In the *Playground* environment, the agent (hand) can move, grasp objects and grow some of them by putting them in contact with food or water. Scenes are generated procedurally with objects of different types, colors and sizes. A social partner provides descriptive feedback (orange) that the agent stores as targetable goals (red bubbles).

Related work

The idea that language understanding is grounded in one's experience of the world and should not be secluded from the perceptual and motor systems has a long history in cognitive science (Glenberg & Kaschak, 2002; Zwaan & Madden, 2005). This vision was transposed to intelligent systems (Steels, 2006; McClelland et al., 2019), applied to human-machine interaction (Dominey, 2005; Madden et al., 2010) and recently to deep RL via frameworks such as *BabyAI* (Chevalier-Boisvert et al., 2019).

In their review of *RL algorithms informed by natural language*, Luketina and colleagues distinguish between *language-conditional* problems where language is required to solve the task and *language-assisted* problems where language is an additional help (Luketina et al., 2019). In the first category, most works propose instruction-following agents (Branavan et al., 2010; Chen & Mooney, 2011; Bahdanau et al., 2019a; Co-Reyes et al., 2019; Jiang et al., 2019; Goyal et al., 2019; Cideron et al., 2020b). Although our system is *language-conditioned*, it is not *language-instructed*: it is never given any instruction or reward but sets its own goals and learns its own internal reward function. Two recent approaches also learn a reward function but require extensive expert knowledge (respectively expert dataset and known environment dynamics), whereas our agent uses experience generated by its own exploration (Bahdanau et al., 2019a; Fu et al., 2019).

Language is also particularly well suited for hindsight experience replay (Andrychowicz et al., 2017): descriptions of the current state can be used to relabel trajectories, enabling agents to transfer skills across goals. Previous works used a hard-coded descriptive function (Chan et al., 2019; Jiang et al., 2019) or trained a generative model to generate goal substitutes (Cideron et al., 2020b; Zhou & Small, 2020). Here, we leverage the learned reward function to scan goal candidates and find good descriptions.

To our knowledge, no previous work has considered the use of compositional goal imagination to enable creative exploration. The linguistic basis of our goal imagination

mechanism is grounded in construction grammar (CG). CG is a usage-based approach that characterizes language learning as a trajectory starting with pattern imitation and discoveries of equivalence classes for argument substitution before evolving towards the recognition and composition of more abstract patterns (Tomasello, 2000a; Goldberg, 2003). This results in a structured inventory of constructions as form-to-meaning mappings that can be combined to create novel and creative utterances (Goldberg, 2003; Bergs, 2019; Hoffmann, 2018; Turner, 2020). The discovery and substitution of equivalent words in learned schemas are observed directly in child language studies (Tomasello & Olguin, 1993; Tomasello, 2000a). Computational implementations of this approach have demonstrated its ability to foster generalization (Hinaut & Dominey, 2013) and were used for data augmentation to improve the performance of neural seq2seq models in natural language processing (Andreas, 2020).

Imagining goals by composing known ones only works in association with *systematic generalization* (Bahdanau et al., 2019a; Hill et al., 2019): generalizations of the type *grow any animal + grasp any plant → grow any plant*. These were found to emerge in instruction-following agents, including generalizations to new combinations of motor predicates, object colors and shapes (Hermann et al., 2017; Hill et al., 2019; Bahdanau et al., 2019a). Systematic generalization can occur when objects share common attributes (e.g. type, color). We directly encode that assumption into our models by representing objects as *single-slot object files* (Green & Quilty-Dunn, 2017): separate entities characterized by shared attributes. Because all objects have similar features, we introduce a new object-centered inductive bias: object-based modular architectures based on *deep sets* (Zaheer et al., 2017).

Contributions

This study introduces:

1. The concept of imagining new goals with language compositionality to drive exploration.
2. IMAGINE: an autotelic agent that uses goal imagination to explore its environment, discover and master object interactions by leveraging linguistic descriptions from a social partner.
3. Modular policy and reward function architectures that demonstrate systematic generalization properties enabling IMAGINE to train on imagined goals. Modularity is based on deep sets, gated attention mechanisms and object-centered representations.
4. *Playground*: a procedurally-generated environment designed to study several types of generalizations (across predicates, attributes, object types and categories).
5. Experiments investigating: i) the effects of our goal imagination mechanism on generalization and exploration; ii) the general properties of imagination mechanisms required for any algorithm to have a similar impact; iii) the impact of modularity and iv) the impact of social interactions.

5.2 Autotelic Skill Acquisition via Social Interactions

Our IMAGINE agent faces the problem of the autonomous acquisition of skills repertoires: it has no prior on the set of achievable goals and receives neither instructions nor external rewards. However, it interacts with a synthetic *social partner* (SP) that offers linguistic guidance towards relevant interactions. Let us describe the environment, the social partner, the grammar and the evaluation metrics.

5.2.1 The Playground Environment

We argue that the study of new mechanisms requires the use of controlled environments. We thus introduce *Playground*, a simple environment designed to study the impact of goal imagination on exploration and generalization by disentangling these questions from the problems of perception and fully-blown natural language understanding. The *Playground* environment is a continuous 2D world, with procedurally-generated scenes containing $N = 3$ objects selected from a set of 32 object types organized into five categories, see Figure 5.2.

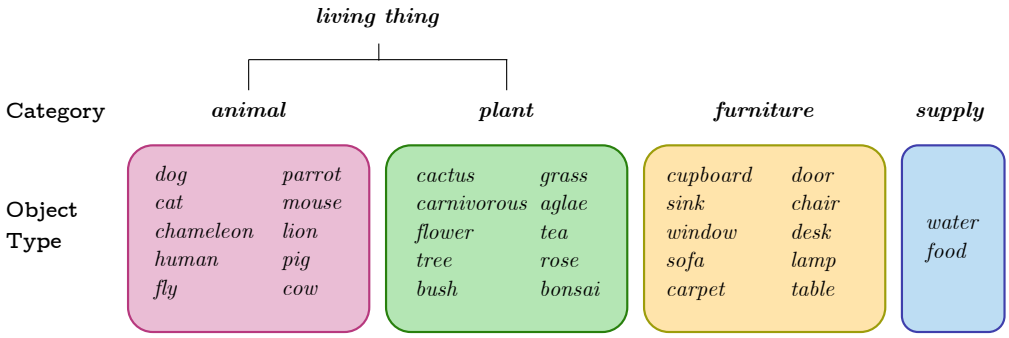


Figure 5.2: Representation of possible object types and categories.

Agents have access to state vectors describing the scene, i.e. the agent's body and the objects. Each object is represented by features describing its type, position, color, size and whether it is grasped. Object types are encoded by one-hots and positioned in the 2D space. Their colors (red, green or blue) are encoded via continuous RGB codes sampled uniformly from corresponding RGB sub-spaces. Categories are not explicitly encoded but can be derived from the object type. Every object is made unique by the procedural generation of its color, size and initial position. The agent can perform bounded translations in the 2D plane, grasp and release objects with its gripper. It can make animals and plants grow by bringing them the suitable supply (food or water for animals, water for plants).

5.2.2 The Social Partner

The agent learns about interesting interactions by interacting with a simulated human caregiver that we call the *social partner* (SP). Following a developmental approach (Asada

et al., 2009), we propose a hard-coded surrogate SP that models essential aspects of the developmental processes seen in humans:

- At the beginning of each episode, the agent chooses a goal by formulating a sentence. SP then provides agents with optimal learning opportunities by organizing the scene with 1) the required objects to reach the goal (not too difficult); 2) procedurally-generated distracting objects (not too easy and providing further discovery opportunities). This constitutes a developmental scaffolding modeling the process of *zone of proximal development* introduced by Vygotsky to describe infant-parent learning dynamics (Vygotsky, 1934).
- At the end of each episode, SP provides the agent with sentences describing achieved and meaningful outcomes (except sentences from a test set). Linguistic guidance given through descriptions is a crucial component of how parents "teach" language to infants, which contrasts with instruction following (providing a linguistic command and then a binary feedback) that is rarely seen in real parent-child interactions (Tomasello, 2009; Bornstein et al., 1992). By default, SP respects the three following properties: 1) *precision*: descriptions are accurate; 2) *exhaustiveness*: it provides all valid descriptions and 3) *full-presence*: it does so at each episode. Section 5.5.5 investigates relaxations of the last two assumptions. Note that SP only describes the final state $\mathbf{s}_T$ of the trajectory: $\mathcal{D}_{\text{SP}}(\mathbf{s}_T) \subset \mathcal{D}^{\text{SP}}$, where $\mathcal{D}^{\text{SP}}$ is the set of descriptions SP can generate.

IMAGINE thus receives linguistic descriptions of its self-directed exploration. As it hears a new description, IMAGINE adds it to the set of targetable goals and learns to represent it by leveraging the alignment between SP's descriptions and its own behavior.

5.2.3 Linguistic Goals and Descriptions

The following grammar generates the descriptions of the 256 achievable goals ($\mathcal{G}^{\text{A}}$):

1. Go: <*go* + ***zone***> *(e.g. go bottom left)*
2. Grasp: < *grasp* + *any* + ***color*** + *thing*> *(e.g. grasp any blue thing)* OR <*grasp* + ***color*** ∪ {*any*} + ***object type*** ∪ ***object category***> *(e.g. grasp red cat)*
3. Grow: <*grow* + *any* + ***color*** + *thing*> *(e.g. grow any red thing)* OR <*grow* + ***color*** ∪ {*any*} + ***living thing*** ∪ {*living_ thing, animal, plant*}> *(e.g. grow green animal)*

Italic bold and { } are sets of words, while *italic* are specific words. ***zone*** refers to areas of the scene (e.g. *left*), ***object type*** refers to a type (e.g. *parrot*) and ***object category*** to one of the five object categories (e.g. *living_ thing*). ***living thing*** refers to any plant or animal word. ***color*** is one of *blue, green, red* and *any* refers to any color or any object. The grammar is structured around the three predicates *go*, *grasp* and *grow*. Objects can be referred to by a combination of their color and either their type or category, or simply by one of these (e.g. *any blue thing*).

The set of achievable goals is partitioned into *training* ($\mathcal{G}^{\text{train}}$) and *testing* ($\mathcal{G}^{\text{test}}$) sets. $\mathcal{G}^{\text{train}}$ is the set of goals corresponding to the subset of interactions SP knows about and can describe $\mathcal{G}^{\text{train}} = \mathcal{D}^{\text{SP}}$. $\mathcal{G}^{\text{test}}$ maximizes the compound divergence and minimizes the atom divergence with respect to $\mathcal{G}^{\text{train}}$. In simpler words, testing and training words (atoms) are the same but testing sentences (compounds) are out of the distribution of training sentences (Keysers et al., 2020). SP only provides descriptions from $\mathcal{G}^{\text{train}}$. We limit the set of goals to control the complexity of our environment and carefully study the generalization properties of IMAGINE.

Pre-verbal infants are known to acquire object-based representations very early (Spelke, 2003; Johnson et al., 2003) and, later, to benefit from a simplified parent-child language during language acquisition (Mintz, 2003). Pursuing a developmental approach, we assume corresponding object-based representations and a simple grammar. As we aim to design agents that bootstrap creative exploration without prior knowledge of possible interactions or language, we do not consider using pre-trained language models. To our knowledge, *Playground* is the first environment to introduce object categories and category-dependent combinatorial dynamics, which allows the study of new types of generalizations. Section 5.4 presents the different types of generalization in depth. Appendix Section B.1 provides more details about the environment, the grammar and SP. We release *Playground* in a separate repository.[1]

5.2.4 Evaluation Metrics

This paper investigates how goal imagination can lead agents to efficiently and creatively explore their environment to discover interesting interactions with objects around. In this quest, SP guides agents towards a set of interesting outcomes by uttering linguistic descriptions. Through compositional recombinations of these sentences, goal imagination aims to drive creative exploration, to push agents to discover outcomes beyond the set of outcomes known by SP. We evaluate this desired behavior with three metrics:

1. **State generalization:** We test agents on a variety of known linguistic goals from the training set (goals that SP knows and describes). This evaluates agents' ability to reach known linguistic goals and generalize to new scenes.

2. **Goal generalization:** We test agents on goals from the testing set (goals that SP never describes). This evaluates agents' ability to generalize systematically to new linguistic goals.

3. **Exploration:** We evaluate the quality of the exploration with an object-oriented exploration metric that we call the *interesting interaction count* (I2C). I2C is computed on different sets of interesting interactions: behaviors a human could infer as goal-directed. These sets include the training, testing sets and an extra set containing interactions such as bringing water or food to inanimate objects. $\text{I2C}_{\mathcal{I}}$ measures the number of times interactions from $\mathcal{I}$ were observed over the last epoch

[1] https://github.com/flowersteam/playground_env

(600 episodes), whether they were targeted or not (see Appendix Section B.3). Thus, I2C measures the penchant of agents to explore interactions with objects around them.

These measures assess the quality of the agents' autotelic exploration. In 1) and 2), the goal-specific generalization score is measured as the success rate over 30 procedurally-generated episodes. Generalization scores are then computed as the average of goal-specific scores over the training (1) and testing (2) sets of goals. Unless specified otherwise, we present means μ and standard deviations over 10 seeds and report statistical significance using a two-tail Welch's t-test with null hypothesis $\mu_1 = \mu_2$ at level $\alpha = 0.05$ (noted by star and circle markers in figures, see justifications in Appendix Section A).

5.3 The IMAGINE Architecture

IMAGINE agents build a repertoire of goals and train two internal models: 1) a goal-achievement reward function $\mathcal{R}$ to predict whether a given description matches a state and 2) a policy π to achieve states matching goal descriptions. The architecture is depicted in Figure 5.3 and follows this logic:

1. The *goal generator* samples a target goal g_{target} from the set of known and imagined goals ($\mathcal{G}_{\text{known}} \cup \mathcal{G}_{\text{im}}$).
2. The *RL agent* interacts with the environment using its policy π conditioned on the goal g_{target}.
3. State-action trajectories are stored in a replay buffer $mem(\pi)$.
4. SP's descriptions of the last state $\mathcal{D}_{\text{SP}}(\mathbf{s}_T)$ are considered as potential goals $\mathcal{G}_{\text{SP}}(\mathbf{s}_T) = \mathcal{D}_{\text{SP}}(\mathbf{s}_T)$.
5. The agent uses positive pairs $(\mathbf{s}_T, \mathcal{G}_{\text{SP}}(\mathbf{s}_T))$ to infer negative pairs $(\mathbf{s}_T, \mathcal{G}_{\text{known}} \setminus \mathcal{G}_{\text{SP}}(\mathbf{s}_T))$ and stores both in memory $mem(\mathcal{R})$.
6. The agent then updates:
   - *Goal generator*: we update the set of known goals $\mathcal{G}_{\text{known}} \leftarrow \mathcal{G}_{\text{known}} \cup \mathcal{G}_{\text{SP}}(\mathbf{s}_T)$ and the set of imagined goals $\mathcal{G}_{im} \leftarrow \text{Imagine}(\mathcal{G}_{\text{known}})$.
   - *Language encoder* (L_e) and *reward function* ($\mathcal{R}$) are updated using data from $mem(\mathcal{R})$.
   - *RL agent*: We sample a batch of state-action transitions $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ from $mem(\pi)$. Then, we use *hindsight replay* and $\mathcal{R}$ to bias the selection of substitute goals to train on ($\mathbf{g}_{\text{s}}$) and compute the associated rewards $(\mathbf{s}, \mathbf{a}, \mathbf{s}', \mathbf{g}_{\text{s}}, r)$. Substituted goals $\mathbf{g}_{\text{s}}$ can be known or imagined goals. Finally, the policy and critic are trained via RL.

Algorithm 3 outlines the pseudo-code of our learning architecture.

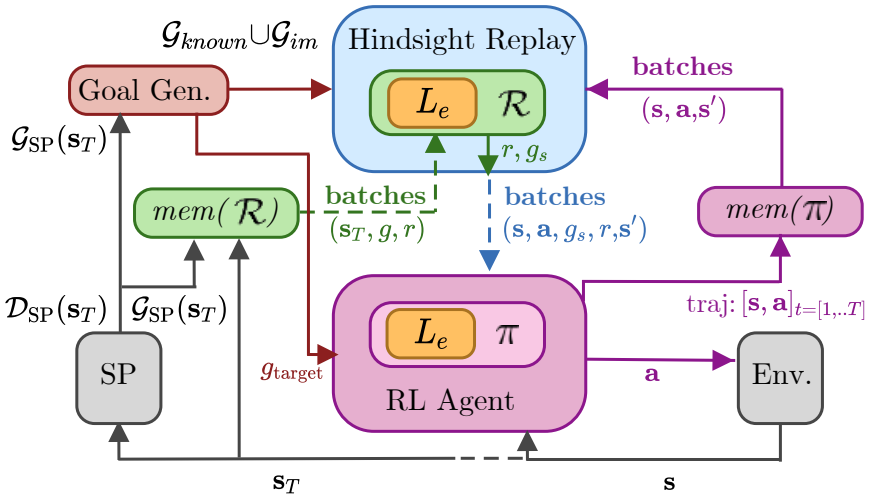


Figure 5.3: The IMAGINE architecture. Colored boxes show the different modules of IMAGINE. Lines represent update signals (dashed) and function outputs (plain). The language encoder L_e is shared by the reward function and the policy.

Algorithm 3 IMAGINE

Input: env, SP
Initialize: language encoder L_e, reward function $\mathcal{R}$, policy π, empty reward function memory $mem(\mathcal{R})$, empty replay buffer $mem(\pi)$, empty set of known goals $\mathcal{G}_{\text{known}}$, empty set of imagined goals $\mathcal{G}_{\text{im}}$
for $e = 1 : N_{\text{episodes}}$ **do**

▷ Goal selection
  if $\mathcal{G}_{\text{known}} \neq \emptyset$ **then**
    sample g_{NL} from $\mathcal{G}_{\text{known}} \cup \mathcal{G}_{\text{im}}$
    $g \leftarrow L_e(g_{\text{NL}})$
  else
    sample g from $\mathcal{N}(0, \mathbf{I})$
▷ Interactions
  $(s_{i-1}, a_i, s_i)_{i \in [\![1,T]\!]} \leftarrow$ env.rollout(π, g)
  $mem(\pi)$.add($(s_{i-1}, a_i, s_i)_{i \in [\![1,T]\!]}$)
  $\mathcal{G}_{\text{SP}} \leftarrow$ SP.get_descriptions(s_T) ▷ ask for descriptions from SP
  $\mathcal{G}_{\text{known}} \leftarrow \mathcal{G}_{\text{known}} \cup \mathcal{G}_{\text{SP}}$ ▷ update set of known goals
  $mem(\mathcal{R})$.add(s_T, $\mathcal{G}_{\text{SP}}$)
▷ Goal Imagination
  if goal imagination allowed **then**
    $\mathcal{G}_{\text{im}} \leftarrow$ **Imagination**($\mathcal{G}_{\text{known}}$) ▷ see Algorithm 4
▷ RL Updates
  Batch$_\pi \leftarrow$ **BatchGenerator**($mem(\pi)$) ▷ Batch$_\pi$=$\{(s, a, s')\}$
  Batch$_\pi \leftarrow$ **Hindsight**(Batch$_\pi$, $\mathcal{R}$, $\mathcal{G}_{\text{known}}$, $\mathcal{G}_{\text{im}}$) ▷ Batch$_\pi$=$\{(s, a, r, g_s, s')\}$ where $r = \mathcal{R}(s, g)$
  $\pi \leftarrow$**RL_Update**(Batch$_\pi$)
▷ Reward function updates
  if e % reward_update_freq == 0 **then**
    Batch$_\mathcal{R} \leftarrow$ **BatchGenerator**($mem(\mathcal{R})$)
    L_e, $\mathcal{R} \leftarrow$ **LE&RewardFunctionUpdate**(Batch$_\mathcal{R}$)

5.3.1 Goal Generator

The goal generator is a generative model of linguistic goals. It generates target goals g_{target} to condition environment interactions and substitutes goals g_{s} for hindsight replay. When goal imagination is disabled, the goal generator samples uniformly from the set of known goals $\mathcal{G}_{\text{known}}$, sampling random vectors if empty. When enabled, it samples with equal probability from known $\mathcal{G}_{\text{known}}$ or imagined $\mathcal{G}_{\text{im}}$ goals. $\mathcal{G}_{\text{im}}$ is generated using a mechanism grounded in construction grammar that leverages the compositionality of language to imagine new goals from $\mathcal{G}_{\text{known}}$. The heuristic consists in computing sets of *equivalent words*: words that appear in two sentences that only differ by one word. For example, from *grasp red lion* and *grow red lion*, *grasp* and *grow* can be considered *equivalent* and, starting from *grasp green tree*, one can imagine a new goal *grow green tree* by substituting "*grasp*" by its equivalent "*grow*" (see Figure 5.1f). Imagined goals do not include known goals. Among them, some are meaningless (e.g. *go left right*), some are syntactically correct but infeasible (e.g. *grow red lamp*) and some belong to $\mathcal{G}^{\text{test}}$, or even to $\mathcal{G}^{\text{train}}$ before they are encountered by the agent and described by SP. Appendix Section B.4 provides the pseudo-code of the goal imagination mechanism and the list of imaginable goals.

5.3.2 Language Encoder

The language encoder (L_e) embeds linguistic goals ($L_e : \mathcal{G}^{\text{NL}} \to \mathbb{R}^{100}$) using an LSTM (Hochreiter & Schmidhuber, 1997) trained jointly with the reward function. L_e acts as a goal translator. It turns the goal-achievement reward function, policy and critic into language-conditioned functions.

5.3.3 Object-Centered Modular Architectures for the Reward Function and Policy

The goal-achievement reward function, policy and critic leverage novel *modular-attention* (MA) architectures based on *deep sets* (Zaheer et al., 2017), gated attention mechanisms (Chaplot et al., 2018) and object-centered representations. The idea is to ensure efficient skill transfer between objects, no matter their position in the state vector. This is done through the combined use of a shared neural network that encodes object-specific features and a permutation-invariant function to aggregate the resulting latent encodings. The shared network independently encodes, for each object, an affordance between this object (object observations), the agent (body observations) and its current goal. The goal embedding generated by L_e is first cast into an attention vector with features in [0, 1], then fused with the concatenation of object and body features via a term-by-term product (gated-attention Chaplot et al., 2018). The resulting object-specific encodings are aggregated by a permutation-invariant function and mapped to the desired output via a final network (e.g. into actions or action-values). Figure 5.4 depicts a graphic representation of these architectures.

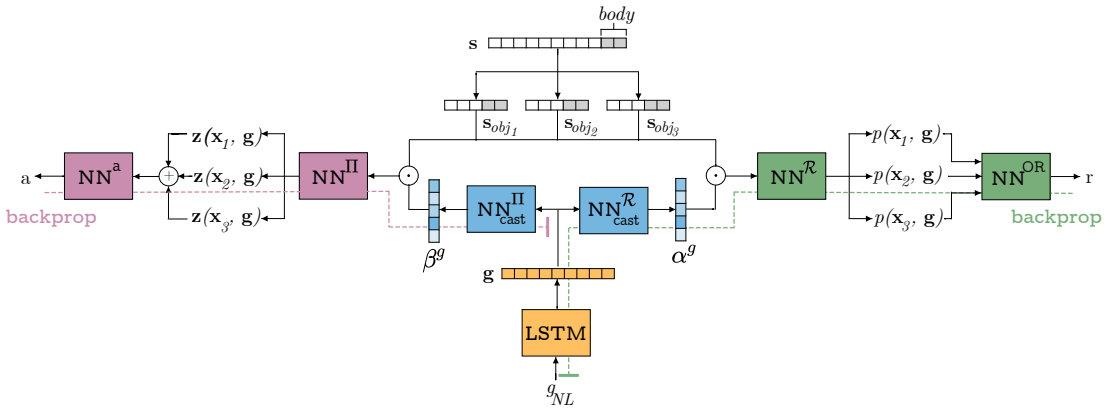


Figure 5.4: Policy and reward function architectures: modular-attention architectures for the policy (left, pink) and reward function (right, green). The language encoder is represented in the bottom in yellow and the attention mechanism in at the center in blue.

5.3.4 Internal Reward Function

The reward function ($\mathcal{R}$) is trained to evaluate the probability that a state and a description match. This can be framed as a binary classification problem $\mathcal{R}(\mathbf{s}, \mathbf{g}) : \mathcal{S} \times \mathbb{R}^{100} \to \{0, 1\}$. We use the MA architecture with attention vectors $\boldsymbol{\alpha}^g$, a shared network $\mathrm{NN}^{\mathcal{R}}$ with output size 1 and a logical OR aggregation. $\mathrm{NN}^{\mathcal{R}}$ computes object-dependent rewards r_i in [0, 1] from the object-specific inputs and the goal embedding. The final binary reward is computed by $\mathrm{NN}^{\mathrm{OR}}$ which outputs 1 whenever $\exists j : r_j > 0.5$. We pre-trained a neural OR function to enable end-to-end training with back-propagation. The overall function is:

$$\mathcal{R}(\mathbf{s}, g) = \mathrm{NN}^{\mathrm{OR}}([\mathrm{NN}^{\mathcal{R}}(\mathbf{s}_{obj(i)} \odot \boldsymbol{\alpha}^g)]_{i \in [\![1, N]\!]}).$$

Interacting with the environment and SP, the agent builds a set of entries $[\mathbf{s}_T, g, r]$ where $r \in \{0, 1\}$ rewards the achievement of goal g in state $\mathbf{s}_T$ for all g in $\mathcal{G}_{\text{known}}$: $r = 1$ if $g \in \mathcal{G}_{\text{SP}}(\mathbf{s}_T)$ and 0 otherwise. We periodically update L_e and $\mathcal{R}$ by minimizing the cross-entropy loss on batches sampled uniformly from the collected data.

5.3.5 Goal-Conditioned RL Agent

Our agent is controlled by a goal-conditioned policy π (Schaul et al., 2015) based on the MA architecture (see Figure 5.4). It uses an attention vector $\boldsymbol{\beta}^g$, a shared network NN^{π}, a sum aggregation and a mapper NN^{a} outputting the actions. Similarly, the critic produces action-values via $\boldsymbol{\gamma}^g$, NN^{Q} and $\mathrm{NN}^{\text{a-v}}$:

$$\pi(\mathbf{s}, g) = \mathrm{NN}^{\mathrm{a}}\Big(\sum_{i \in [1..N]} \mathrm{NN}^{\pi}(\mathbf{s}_{obj(i)} \odot \boldsymbol{\beta}^g)\Big)$$
$$Q(\mathbf{s}, \mathbf{a}, g) = \mathrm{NN}^{\text{a-v}}\Big(\sum_{i \in [1..N]} \mathrm{NN}^{Q}([\mathbf{s}_{obj(i)}, \mathbf{a}] \odot \boldsymbol{\gamma}^g)\Big).$$

We use DDPG to train both the actor and the critic (Lillicrap et al., 2016), although we could use any other off-policy algorithm. As detailed in Appendix Section B.6, our agent uses a form of hindsight experience replay (Andrychowicz et al., 2017).

5.4 Systematic Generalization in Playground

Because we generate scenes procedurally, the average success rate ($\overline{\texttt{SR}}$) computed on $\mathcal{G}^{\text{train}}$ measures the generalization to new states. However, when computed on $\mathcal{G}^{\text{test}}$, $\overline{\texttt{SR}}$ measures both this state generalization and the systematic generalization to new goal descriptions. This section focuses solely on generalization in the language space: $\overline{\texttt{SR}}_{\text{test}}$.

5.4.1 Different Types of Generalization

Generalization can occur in two different modules of the IMAGINE architecture: the reward function and the policy. Agents can only benefit from goal imagination when their reward function can generalize the meanings of imagined goals from the meanings of known ones. In that case, agents can further train on imagined goals, which might, in turn, reinforce the generalization of their policy. This section characterizes different types of generalizations that the reward and policy can both demonstrate.

- Type 1 – *Attribute–object generalization*: The ability to accurately associate an attribute and an object that were never seen together before. For example, interpreting the goal *grasp red tree* requires 1) the isolation of the *red* and *tree* concepts from other sentences and 2) their combination to recognize a *red tree*. To measure this ability, we removed from the training set all goals containing the attribute-object combinations *{blue door, red tree, green dog}* and added them to the testing set (4 goals).

- Type 2 – *Object identification*: The ability to identify a new object from its attribute. We left out of the training set all goals containing the word *flower* (4 goals). For example, interpreting the goal *grasp red flower* requires 1) the isolation of *red* concept and 2) its transposition to the unknown object *flower*. Note that, in the case of *grasp any flower*, the agent cannot rely on the attribute and must perform some kind of complement reasoning: "if these are known objects, and that is unknown, then it must be a *flower*."

- Type 3 – *Predicate–category generalization*: The ability to interpret an action predicate applied on an object category when they were never seen together before. A category regroups a set of objects and is not encoded in the object state vector. It is only a linguistic concept. We left out all goals with the *grasp* predicate and the *animal* category (4 goals). Correctly interpreting *grasp any animal* requires 1) the identification of objects belonging to the *animal* category (from experiences with the *grow* predicate), 2) the isolation of the *grasping* concept (acquired from grasping non-*animal* objects) and 3) their combination.

- Type 4 – *Predicate–object generalization*: The ability to interpret an action predicate applied on an object when they were never seen together before. We leave out all

goals with the *grasp* predicate and the *fly* object (4 goals). To correctly interpret *grasp any fly*, the agent should leverage its knowledge about the *grasp* predicate (acquired from the "grasping non-fly objects" goals) and the *fly* object (acquired from the "growing flies" goals).

- Type 5 – *Predicate–dynamics generalization*: The ability to generalize the behavior associated with an action predicate to another category of objects for which the dynamics differs. In the *Playground* environment, the dynamics of *grow* with ***animals*** and ***plants*** is a bit different. ***animals*** can be grown with *food* and *water*, whereas ***plants*** only grow with *water*. We want to see whether IMAGINE can transfer growing behaviors from ***animals*** to ***plants***. We left out all goals with the *grow* predicate and any ***plant*** object, *plant* and *living thing* words (48 goals). For example, interpreting *grow any plant* requires 1) the identification of ***plant*** objects (acquired from the "grasping plants" goals) and 2) knowing that objects need supplies (food or water) to *grow* (acquired from the "growing animals" goals). Type 5 is more complex than Type 4 because the dynamics change and because it mixes objects (***plants***) and categories (*plant*, *living thing*). By definition, we test zero-shot generalization without additional reward signals (before imagination). As a result, even the perfect zero-shot generalization cannot perfectly adapt the *grow* behavior from animals to plant and would bring food and water with equal probability $p = 0.5$ for each.

Appendix Table B.1 provides the exhaustive list of goals used to test each type of generalization.

5.4.2 Different Ways to Generalize

Agents can generalize to out-of-distribution goals (from any of the five categories above) in three different ways:

1. *Policy zero-shot generalization*: The policy can achieve a new goal without additional training.
2. *Reward zero-shot generalization*: The reward can tell whether a goal is achieved or not without additional training.
3. *Policy n-shot generalization or behavioral adaptation*: When allowed to imagine goals, IMAGINE agents can use the zero-shot generalization of their reward function to train their policy to improve on imagined goals autonomously. After such training, the policy might show improved generalization performance compared to its zero-shot abilities. We call this performance *n-shot generalization*. The policy received additional training thanks to the internal reward function but did not leverage additional external feedback. This is crucial for achieving Type 5 generalization. As we said, zero-shot generalization cannot figure out that plants only grow with water. Fine-tuning the policy based on experience and internal rewards enables agents to perform *behavioral adaptation*: adapting their behavior towards imagined goals autonomously.

5.5 Experiments

This section first showcases the impact of goal imagination on exploration and generalization (Section 5.5.1). To complete the picture, we analyze other goal imagination mechanisms and investigate the properties enabling these effects (Section 5.5.3). Finally, we show that our modular architectures are crucial to a successful goal imagination (Section 5.5.4) and discuss more realistic interactions with SP (Section 5.5.5).

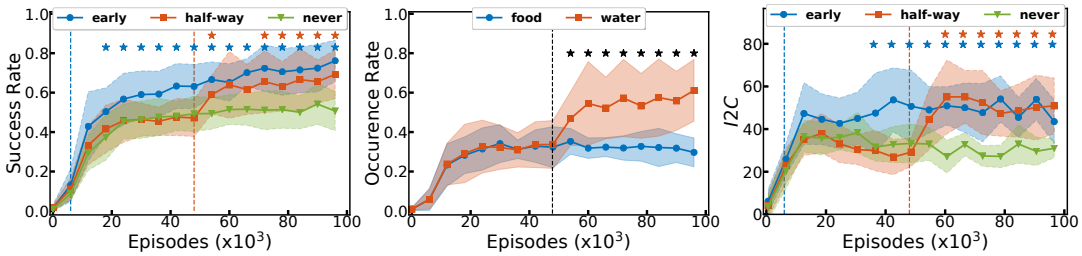
IMAGINE agents achieve near-perfect generalizations to new states, $\overline{\texttt{SR}}_{\text{train}} = 0.95 \pm 0.05$. Thus, we focus on generalization in language space and exploration ($\overline{\texttt{SR}}_{\text{test}}$). Appendix Sections B.2 to B.7 provide additional results and insights organized by theme (generalization, exploration, goal imagination, architectures, reward function and visualizations).

5.5.1 The Impact of Goal Imagination on Generalization and Exploration

This section investigates the impact of goal imagination on the systematic generalization and the exploration abilities of the IMAGINE agent.

Impact on Generalization

Figure 5.5a shows $\overline{\texttt{SR}}_{\text{test}}$ when the agent starts imagining goals early (after $6 \cdot 10^3$ episodes), halfway (after $48 \cdot 10^3$ episodes) or when not allowed to do so. Goal imagination leads to significant improvements in generalization.



(a) (b) (c)

Figure 5.5: Goal imagination drives exploration and generalization. Vertical dashed lines mark the onset of goal imagination. a) $\overline{\texttt{SR}}_{\text{test}}$. b) Behavioral adaptation, empirical probabilities that the agent brings supplies to a plant when trying to grow it. c) I2C computed on the testing set. Stars indicate significance (a and c are tested against *never*).

Behavioral Adaptation

Agents learn to grow animals from SP's descriptions but are never told they could grow plants. When evaluated offline on the *growing-plants* goals before goal imagination, agents' policies perform a sensible zero-shot generalization and bring them water or food

with equal probability, just like they would do for animals (Figure 5.5b, left). This, however, is a case of over-generalization. Although the grammar is the same, the required behaviors differ—plants only grow with water. The structure of the world does not match the compositional structure of language. As agents start to imagine and target these goals, their behavior adapts (Figure 5.5b, right). If the reward function generalizes well, it only provides positive rewards when the agent brings water to the plants. The policy slowly adapts to this internal reward signal and pushes agents to bring more water. We call this phenomenon *behavioral adaptation*—agents recover from cases of over-generalization.

Impact on Exploration

Figure 5.5c presents the I2C metric computed on the set of interactions related to $\mathcal{G}^{\text{test}}$ and demonstrates the exploration boost triggered by goal imagination. Appendix Section B.3 presents other I2C metrics computed on additional interaction sets.

5.5.2 Systematic Generalization

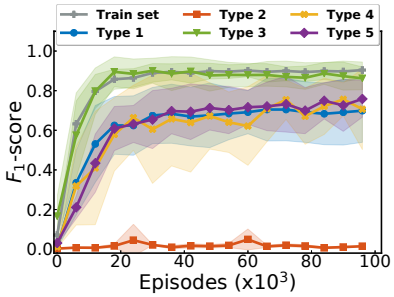
Figure 5.6 presents training and generalization performance of the reward function and policy. We discuss the zero-shot and n-shot generalization abilities of the reward function and policy in the five types of generalizations.

Reward Function Zero-Shot Generalization

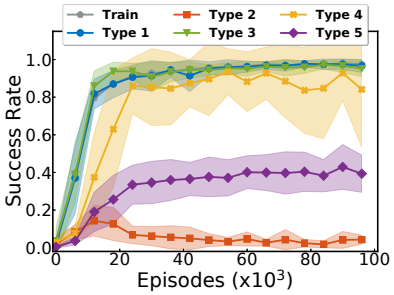
When the reward function is trained in parallel with the policy, we monitor its zero-shot generalization capabilities by computing the F_1-score over a dataset collected separately with a trained policy run on goals from $\mathcal{G}^{\text{test}}$ (kept fixed across runs for fair comparisons). As shown in Figure 5.6a, the reward function exhibits good zero-shot generalization properties over four types of generalization after 25×10^3 episodes. Note that, because we test on data collected with a different RL policy, the F_1-scores presented in Figure 5.6a may not faithfully describe the true generalization of the reward function during co-training.

Policy Zero-Shot Generalization

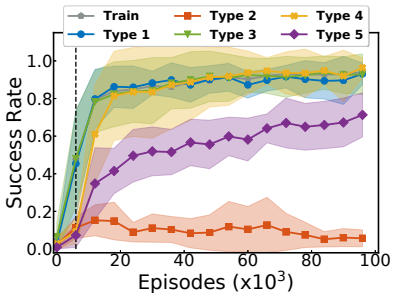
The zero-shot performance of the policy is evaluated in Figure 5.6b (*no imagination* condition) and in the period preceding goal imagination in Figure 5.6c and 5.6d (before the vertical dashed lines). The policy shows excellent zero-shot generalization properties for Type 1, 3 and 4, average zero-shot generalization on Type 5 and fails to generalize on Type 2. Type 1, 3 and 4 can be said to have similar levels of difficulty, as they all require to learn two concepts individually before combining them at test time. Type 2 is much more difficult as the word "flower" appears for the first time. The language encoder receives a new word token, which seems to disturb behavior. As said earlier, zero-shot generalization on Type 5 cannot do better than 0.5, as it cannot infer that plants only require water.



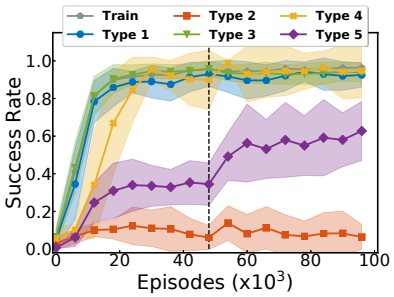
(a) **Reward Function, no imagination**



(b) **Policy, no imagination**



(c) **Policy, imagination halfway**



(d) **Policy, imagination early**

Figure 5.6: Zero-shot and n-shot generalizations of the reward function and policy. Each figure represents the training and testing performance (split by generalization type) for the reward (a) and the policy (b, c, d). (a) and (b) represent zero-shot performance in the *no imagination* conditions. In (c) and (d), agents start to imagine goals as denoted by the vertical dashed line. Before that line, $\overline{\texttt{SR}}$ evaluates the zero-shot generalization. After, it evaluates the n-shot generalization, as agents can train autonomously on imagined goals.

Policy N-Shot Generalization

When goal imagination begins (Figures 5.6c and 5.6d after the vertical lines), agents can imagine goals and train on them. This means that $\overline{\texttt{SR}}$ evaluates n-shot policy generalization. Agents can now perform *behavior adaptation*. They can learn that plants need water. As they learn this, their generalization performance on Type 5 goals increases and goes beyond 0.5. Note that this effects fights the zero-shot generalization. By default, the policy and the reward function apply zero-shot generalization: e.g. they bring water or food equally to plants. Behavioral adaptation attempts to modify that default behavior. Because of the poor zero-shot generalization of the reward on goals of Type 2, agents cannot hope to learn Type 2 behaviors. Moreover, they cannot imagine Type 2 goals because they never encountered the word *flower*.

5.5.3 Properties of Imagination Mechanisms

We propose to characterize goal imagination mechanisms by two properties: 1) *Coverage*: the fraction of $\mathcal{G}^{\text{test}}$ found in $\mathcal{G}_{\text{im}}$ and 2) *Precision*: the fraction of the imagined goals that are achievable. We compare our goal imagination mechanism based on the construction grammar heuristic (CGH) to variants characterized by 1) lower coverage; 2) lower precision; 3) perfect coverage and precision (oracle); 4) random goal imagination baseline (random sequences of words from $\mathcal{G}^{\text{train}}$ leading to near null coverage and precision). These measures are computed at the end of experiments, when all goals from $\mathcal{G}^{\text{train}}$ have been discovered (Table 5.1).

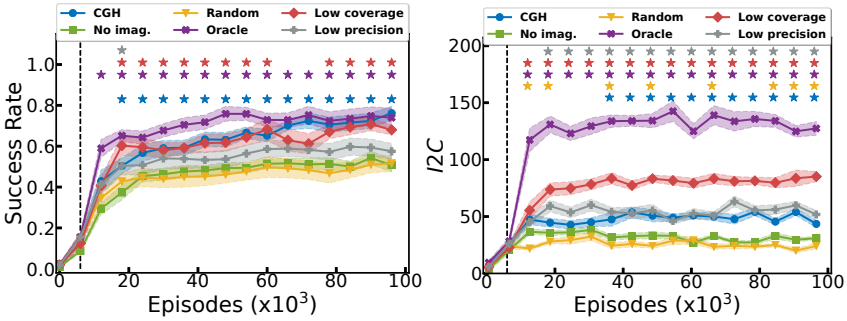
	Coverage	**Precision**
CGH	0.87	0.45
Oracle	1	1
Low Coverage	0.44	0.45
Low Precision	0.87	0.30
Random G.	≈ 0	≈ 0

Table 5.1: Coverage and precision of different goal imagination heuristics.

Figure 5.7a shows that CGH achieves a generalization performance on par with the oracle. Reducing the coverage of the goal imagination mechanism still brings significant improvements in generalization. In the *low coverage* condition, Appendix Section B.4 shows that the generalization performance on the imagined testing goals is not statistically different from the performance on similar non-imagined ones. This implies that the systematic generalization boost triggered by imagined goals also benefits non-imagined ones. Finally, reducing the precision of imagined goals (gray curve) seems to impede generalization (no significant difference with the *no imagination* baseline).

Figure 5.7b shows that all non-random goal imagination heuristics enable a significant exploration boost. The random goal baseline acts as a control condition. It demonstrates that the generalization and exploration boosts are not due to a mere effect of network regularization resulting from the addition of random goals (no significant effect with the

no imagination baseline). Using random goal embeddings instead did not produce any significant effect neither.



(a) (b)

Figure 5.7: **Goal imagination properties.** (a): $\overline{\text{SR}}$ on testing set. (b): I2C on $\mathcal{G}^{\text{test}}$. We report the standard error of the mean. Stars indicate significant differences with the *no imagination* condition.

5.5.4 Modularity Interacts with Goal Imagination

We compared MA to flat architectures (FA) that consider the whole scene at once. As the use of FA for the reward function showed poor performance on $\mathcal{G}^{\text{train}}$, Table 5.2 only compares the use of MA and FA for the policy and assumes an MA architecture for the reward function. MA shows stronger generalization and is the only architecture allowing an additional boost with goal imagination. Only MA policy architectures can leverage the novel reward signals coming from imagined goals and turn them into *behavioral adaptation*. Appendix Section B.5 provides additional details.

Table 5.2: Policy architectures performance ($\overline{\text{SR}}_{\text{test}}$ after convergence). Bold indicates significant differences between the imagination condition and the baseline.

	MA	FA
Im.	0.76 ± 0.1	0.15 ± 0.05
No Im.	0.51 ± 0.1	0.17 ± 0.04
p-val	$\mathbf{4.8 \times 10^{-5}}$	0.66

5.5.5 Properties of Linguistic Feedback

We study the relaxation of the *full-presence* and *exhaustiveness* assumptions of the social partner, see Figure 5.8. We first relax the *full-presence* while keeping *exhaustiveness* (blue, yellow and purple curves). When SP has a 10% chance of being present (yellow), imaginative agents show generalization performance on par with the unimaginative agents trained in a full-presence setting (green). However, when the same amount of feedback

is concentrated in the first 10% episodes (purple), goal imagination enables significant improvements in generalization compared to the no-imagination condition in green. This is reminiscent of children who become more and more autonomous as they grow into adulthood and is consistent with experiments from Chan et al. (2019). When we relax *exhaustiveness*, SP only provides one positive and one negative description every episode (red) or in 50% of the episodes (gray). Then, the generalization performance matches the one of unimaginative agents in the exhaustive setting (green). In all conditions, the training performance $\overline{\text{SR}}_{\text{train}}$ remains near-perfect.

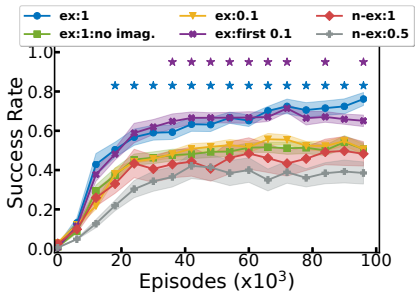


Figure 5.8: Influence of social feedback. $\overline{\text{SR}}$ on $\mathcal{G}^{\text{test}}$ for different social strategies. Stars indicate significant differences with the exhaustive condition without imagination (*ex:1 no imag.*). sem plotted, 5 seeds.

5.6 Discussion

IMAGINE is a learning architecture that tackles the autonomous acquisition of skills repertoires by leveraging linguistic interactions with a social partner. Goal embeddings, goal spaces and goal-achievement functions are all learned from interactions with a synthetic social partners describing the agent's trajectories. By internalizing goal representations, the IMAGINE agent becomes more and more self-sufficient: it targets its own goal, generates its own learning reward and learns goal-oriented skills autonomously.

The standard definition of creativity as *novelty times appropriateness* (Simonton, 2012; Runco & Jaeger, 2012) shares similarities with the *intermediate novelty principle* (Berlyne, 1950). Not novel enough is boring (e.g. known sentences), but too novel is overwhelming (e.g. sentences with random words). The sweet spot is in the middle—using novel instances of known constructions. Thus, our goal imagination mechanism offers a simple way to achieve such intermediate novelty and creativity. With it, IMAGINE can explore beyond the set of goals it was nudged towards by the social partner. It uses language as a cognitive tool to imagine out-of-distribution goals that power a creative object-oriented exploration and favor systematic generalization. Goal imagination also leads to funny behaviors like attempting to grow pieces of furniture with food or water. This echoes the way a child may try to feed his doll during pretend play (Vygotsky, 1933; Singer & Singer, 2009; Chu & Schulz, 2020b).

In 1950, Alan Turing hypothesized that human-level intelligence would be reached by teaching machines like we teach children (Turing, 1950). Building on this intuition, a *roadmap towards machine intelligence* recently proposed to first pre-train agents to learn

efficiently and communicate with humans in a controlled simulated environment before letting them interact and learn with humans (Mikolov et al., 2016). IMAGINE is in line with these propositions. Although the current implementation requires much feedback, our experiments show that feedback is mainly required in the early phases of learning. Moreover, improvements in supervised learning methods (to learn the reward function) and RL methods (to learn the policy) can directly benefit IMAGINE's sample efficiency, making it more and more accessible to direct human teaching.

Attention mechanisms contribute to interpretability and are thus interesting for future human-agent interactions. Looking at the attentional scaling factors that result from linguistic descriptions can indeed offer a sneak peek at the agent's inner workings (see Appendix Figure B.11).

IMAGINE differs from standard language-conditioned RL setups in three aspects: 1) it is not instructed but generates its own goals; 2) the social feedback is descriptive, not valued and 3) it internalizes a reward function and learns from internal signals. Because IMAGINE is not instructed, it is left free to generate its own goals and organize its own exploration. This aligns with studies from developmental psychology showing that children explore more in non-pedagogical settings, when they are not explicitly instructed but left free to interact:

> "*Initially, direct instruction offers a fast strategy for concept learning. However, over time, children who receive only direct instruction will be less likely to explore and discover relevant strategies, and thus less able to acquire and consolidate the relevant concepts.*" (Bonawitz et al., 2009)

In IMAGINE as well, the social partner provides crucial feedback for the agents to ground their first concepts. However, free exploration organized as the pursuit of imagined goals allows agents to enhance their generalization abilities, i.e. consolidate and generalize their concepts.

IMAGINE learns about interesting behaviors from linguistic interactions with SP. In contrast with hand-crafted reward functions, linguistic descriptions provide an easy way to guide machines towards relevant interactions. *A posteriori* counterfactual feedback is easier to communicate for humans, especially when possible effects—and thus possible instructions—are unknown. Agents can also greatly benefit from such counterfactual feedback via *hindsight learning*—agents can use a single trajectory to learn about several pretend goals (Andrychowicz et al., 2017; Eysenbach et al., 2020).

We can draw a parallel with the pedagogy implemented in Montessori-like schools. In such schools, children mainly receive factual feedback (achieved/not achieved or descriptive) and often rely on feedback from their peers. In contrast, children in traditional schools receive valued feedback (good/bad) primarily from their teacher. As studies showed, Montessori children seem to demonstrate a more positive attitude towards failure, adapt quicker and autonomously to new situations, be more creative and explore more in general (Dénervaud & Gentaz, 2015; Denervaud et al., 2019, 2020). A hypothesis—that remains to be tested—could be that hearing factual feedback from several people, including peers, compels the child to model feedback internally; *if my friends can judge,*

why couldn't I? In contrast, the child might not see feedback that contains an emotional value and comes from a unique teacher as something that can be modeled. Such feedback appears more subjective and external. Children from Montessori-like schools would thus have a higher tendency to internalize reward functions than others. With an internal reward function, as we saw in the IMAGINE study, agents can target and learn about imagined goals. This could explain higher creativity, exploration and adaptation. Internal signals are not social, thus deprived of emotional values. This could explain a more positive attitude towards failure.

Playground is a tool that we hope will enable the community to further study under-explored descriptive setups with rich combinatorial dynamics. It is designed for the study of goal imagination and combinatorial generalization and is disentangled from other RL problems such as state-space exploration, sparse rewards or partial observability. Compared to existing environments (e.g. Hermann et al., 2017; Chevalier-Boisvert et al., 2019; Chan et al., 2019), we allow the use of descriptive feedback, introduce the notion of object categories and category-dependent object interactions (*grow* refer to different modalities for *plants* or *animals*). Playground can easily be extended by adding objects, attributes, category- or object-type-dependent dynamics.

Future Work

Although IMAGINE was presented in a controlled setting, we think the approach can scale to harder problems involving more complex states and language inputs. IMAGINE could be combined with unsupervised multi-object representation learning algorithms (Burgess et al., 2019; Greff et al., 2019, 2020) to work directly from pixels (see a similar approach in Ding et al., 2020). The resulting algorithm would still be different from goal-as-state approaches (Nair et al., 2018b; Pong et al., 2020; Nair et al., 2020). Indeed, linguistic goals represent abstract binary problems whereas goal-as-state approaches deal with target features—see discussion in Appendix Section B.8.

A more complex language could be introduced. Fpr example, one could consider object relationships (e.g. *grasp any X left of Y*), see Karch et al. (2020) for our preliminary experiments in this direction. While the use of pre-trained language models (Radford et al., 2019; Brown et al., 2020) is incompatible with our developmental approach, it would be interesting to study how to leverage them for creative goal imagination. Because CGH performs well in our setup with a medium precision (0.45) and because similar mechanisms were successfully used for data augmentation in complex natural language processing tasks (Andreas, 2020), we believe our goal imagination heuristic could scale to more natural languages. A potential solution to this problem could be to train a language *translator* to project natural language to the simple internal synthetic language using methods inspired by Marzoev et al. (2020).

We could reduce the burden on SP by considering unreliable feedback (lower precision) or conditioning goal generation on the initial scene (e.g. using mechanisms from Cideron et al., 2020b). One could also add new interaction modalities by letting SP perform demonstrations, propose goals or guide the agent's attention. Our modular architectures, because they are set functions, could also directly be used to consider variable numbers of objects. Finally, we could use offline learning (Fujimoto et al., 2019; Levine et al., 2020)

to reinterpret past experience in the light of new imagined goals without any additional environment interactions.

Links

Demonstration videos are available at https://sites.google.com/view/imagine-drl. The source code of playground environment can be found at https://github.com/flowersteam/playground_env and the source code of the IMAGINE architecture https://github.com/flowersteam/Imagine.

Chapter 6

Language-Conditioned Goal Generation: LGB

This chapter introduces a novel linguistic RL-IMGEP architecture called LGB for *Language-Goal-Behavior*. In contrast to existing language-conditioned RL algorithms, LGB decouples language grounding from skill learning through an intermediate semantic representation. Let us see why this might be a good idea.

6.1 Motivations and Related Work

The last chapter presented IMAGINE, a RL-IMGEP agent conditioned on its own linguistic goals. Linguistic goal representations offer two main advantages: abstraction and generalization. Whereas goal-as-state approaches express goals as concrete targets in the state space, language can express abstract goals as arbitrary constraints on the agent's trajectory. Language further offers the possibility to *generalize systematically*, i.e. to readily represent and achieve novel linguistic goals formed by recombinations of known linguistic concepts, see Sections 4.3 and 4.4.

However, direct language conditioning suffers from three drawbacks. First, it couples skill learning and language grounding. Language-conditioned agents cannot learn skills without linguistic inputs, which contrasts with the goal-directed behaviors of preverbal infants (Mandler, 1999). Second, direct conditioning limits the behavioral diversity associated with a particular language input: a single goal leads to a low diversity of behaviors that only results from undirected stochasticity in the policy or in the environmental dynamics. The agent can achieve several goals but cannot achieve a goal in several ways. Third, this lack of behavioral diversity prevents agents from switching strategy after a failure.

We propose to decouple skill learning and language grounding via an intermediate semantic representation to circumvent these three limitations. On the one hand, agents can learn skills without language by targeting configurations from the semantic representation space (skill learning). On the other hand, they can learn to generate valid semantic configurations matching the constraints expressed by language descriptions (language grounding). This language-conditioned goal generation is the backbone of behavioral diversity: a given sentence can correspond to a whole set of matching configurations.

After skill learning and language grounding, the agent can execute linguistic goals by first sampling a set of matching semantic configurations, then pursuing one of them.

The architecture we propose takes inspiration from research in developmental psychology. Spelke's theory of *core knowledge* argues that the specific features of human cognition are built on a restricted set of early-developing *core knowledge systems*: for objects and their interactions, for agents and their goal-directed behaviors, for numbers and arithmetic, for space and geometry and for social partners (Spelke, 2003; Spelke & Kinzler, 2007). In the same spirit, Mandler demonstrated that infants possess innate or early-developing predicates characterizing spatial relations between objects (Mandler, 2012). Our architecture uses such innate semantic predicates to form intermediate representations between skills and language. Spelke argues that language, thanks to its combinatorial properties, is the human-specific feature that binds all core knowledge systems into general cross-domain representations. In our architecture as well, language is used to build more abstract representations on top of innate semantic ones.

The ability to generate possible configurations from a linguistic description is also reminiscent of the *embodied simulation hypothesis*. According to this hypothesis, language triggers a rich diversity of mental simulations in humans (see an exciting account of these ideas in Bergen, 2012). When our agents receive a linguistic description, they generate a diversity of matching cognitive representations—i.e. a diversity of possible futures. From these representations, they can select one and try to make it happen.

Contributions

We propose a novel conceptual RL architecture named LGB for Language-Goal-Behavior and pictured in Figure 6.1 (right). The LGB architecture enables agents to decouple the autotelic acquisition of repertoires of skills (Goals → Behavior) from language grounding (Language → Goals) via the use of semantic goal representation. To our knowledge, the LGB architecture is the only one to combine the following four features:

- It is autotelic: it selects its own semantic goals and generates its own rewards,
- It decouples skill learning from language grounding, accounting for preverbal infants learning,
- It exhibits a diversity of behaviors for any given linguistic goal,
- It can switch strategy in case of failure.

Besides, we introduce an instance of LGB named DECSTR for **DE**ep sets and **C**urriculum with **S**eman**T**ic goal **R**epresentations. Using DECSTR, we showcase the advantages of the conceptual decoupling idea. In the *skill learning* phase, DECSTR evolves in a robotic manipulation environment and leverages semantic representations based on predicates describing spatial relations between physical objects. These predicates are known to be used by infants from a very young age (Spelke, 2003; Spelke & Kinzler, 2007; Mandler, 2012). DECSTR autonomously learns to discover and master all reachable configurations in its semantic representation space. In the *language grounding* phase, we train a conditional VAE (C-VAE) to generate semantic goals from linguistic descriptions. Finally, we can

evaluate the agent in an *instruction-following* phase by composing the two first phases: first, sampling a set of semantic goals from a description, then pursuing one. The experimental section investigates three questions: how does DECSTR perform in the three phases? How does it compare to end-to-end language-conditioned RL approaches? Do we need intermediate representations to be semantic? Code and videos can be found at https://sites.google.com/view/decstr/.

Related Work

Most approaches from the language-conditioned RL literature (LC-RL) define *instruction-following* agents receiving external instructions and rewards (e.g. Hermann et al., 2017; Chan et al., 2019; Bahdanau et al., 2019a; Cideron et al., 2020b; Jiang et al., 2019; Fu et al., 2019, see review in Chapter 4). One exception is the autotelic IMAGINE approach introduced in the previous chapter. In all cases, the linguistic condition prevents the decoupling of language grounding and skill learning, true behavioral diversity and efficient *strategy-switching* behaviors. Our approach is different. It decouple language grounding from skill learning. The language-conditioned goal generation allows behavioral diversity and strategy-switching behaviors.

DECSTR, our proposed implementation of LGB, evolves in a block manipulation domain. Stacking blocks is one of the earliest benchmarks in artificial intelligence (e.g. Sussman, 1973; Tate & others, 1975) and has led to many simulation and robotics studies (Deisenroth et al., 2011; Xu et al., 2018). Recently, two papers demonstrated impressive results by stacking up to 4 and 6 blocks, respectively (Lanier et al., 2019; Li et al., 2020). However, these approaches are not intrinsically motivated, involve hand-defined curriculum strategies and express goals as specific target block positions. In contrast, DECSTR is intrinsically motivated, builds its own curriculum and uses semantic goal representations (symbolic or language-based) based on spatial relations between blocks.

Several works use semantic representations to associate meanings and skills (Tellex et al., 2011; Kulick et al., 2013; Al-Omari et al., 2017). The first two use semantic representations as an intermediate layer between language and skills while the third does not use language. Whereas DECSTR acquires skills autonomously, previous approaches all use skills that are either manually generated (Al-Omari et al., 2017), hand-engineered (Tellex et al., 2011) or obtained via optimal control methods (Kulick et al., 2013). Closer to us, an interesting approach decouples skill learning from language grounding in a goal-conditioned imitation learning paradigm by mapping both linguistic goals and visual goals to a shared representation space (Lynch & Sermanet, 2020). This approach is not intrinsically motivated because agents are trained from a dataset of human teleoperated trajectories and are externally instructed. Behavioral diversity and the resulting strategy-switching behaviors are also limited by a one-to-one correspondence between visual and linguistic goals.

6.2 The Language-Goal-Behavior Architecture

LGB architectures are designed to solve the problem of the autonomous acquisition of skills repertoires in non-linguistic setups, followed by a language grounding phase. LGB

agents leverage a semantic representation of their environment and, in a first skill learning phase, must discover and master as many semantic configurations as possible. In a second phase, they can interact with a descriptive social partner and must ground linguistic descriptions into the skills acquired in phase 1.

LGB architectures are composed of three main modules. First, the *semantic representation* defines the behavioral and goal spaces of the agent. Second, an autotelic GC-RL algorithm organizes the skill learning phase. Third, the language-conditioned goal generator implements the language grounding phase. The instruction-following phase is the combination of the skill learning and language grounding phases. The three phases are respectively called G→B for Goal → Behavior, L→G for Language → Goal and L→G→B for Language → Goal → Behavior, see Figure 6.1 and Appendix C.1.

Instances of the LGB architecture should demonstrate the four properties listed in the introduction: 1) be autotelic; 2) decouple skill learning and language grounding; 3) favor behavioral diversity; 4) allow strategy-switching. We argue that any LGB algorithm should fulfill the following constraints. For LGB to be autotelic (1), the algorithm must integrate the generation and selection of semantic goals and generate its own rewards. Decoupling is ensured by design (2). For LGB to demonstrate behavioral diversity and strategy switching (3, 4), the language-conditioned goal generator must effectively model the distribution of semantic goals satisfying the constraints expressed by any linguistic description.

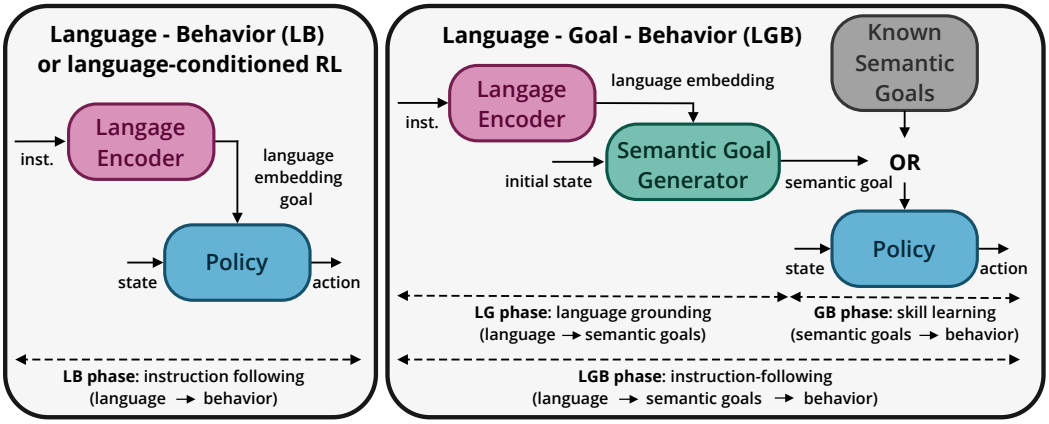


Figure 6.1: A standard language-conditioned RL architecture (left) and our proposed LGB architecture (right).

6.2.1 Semantic Representations in the Fetch Manipulate Environment

DECSTR evolves in the *Fetch Manipulate* environment: a robotic manipulation domain based on MUJOCO (Todorov et al., 2012) and derived from the Fetch tasks (Plappert et al., 2018a), see Figure 6.2. Actions are 4-dimensional: 3D gripper velocities and grasping velocity. Observations include the Cartesian and angular positions and velocities of the gripper and the three blocks.

We assume DECSTR has access to innate semantic representations based on a list of predicates describing spatial relations between pairs of objects in the scene. We consider two of the spatial predicates infants demonstrate early in their development: the *close* and the *above* binary predicates (Mandler, 2012). We apply these predicates to all permutations of object pairs for the 3 objects we consider: 6 permutations for the *above* predicate and 3 combinations for the *close* predicate due to its order-invariance. A *semantic configuration* is the concatenation of the evaluations of these 9 predicates and represents spatial relations between objects in the scene. In the resulting semantic configuration space $\{0,1\}^9$, the agent can reach 35 physically valid configurations, including stacks of 2 or 3 blocks and pyramids, see examples in Figure 6.2. The binary reward function directly derives from the semantic mapping: the agent rewards itself when its current configuration c_p matches the goal configuration $c_p = g$. Appendix C.2 provides formal definitions and properties of predicates and semantic configurations.

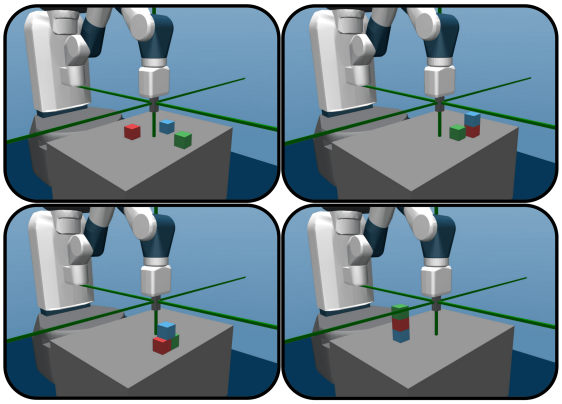


Figure 6.2: Example configurations. Top-right: all pairs of blocks are *close* (111) and the blue block is above the red block (000100). The resulting configuration is 111000100.

Inspired by Vygotsky's concept of *zone of proximal development* (Vygotsky, 1934), we let a social partner facilitate DECSTR's exploration by providing non-trivial initial configurations. After a first period of autonomous exploration, the social partner initializes the scene with stacks of 2 blocks in 21% of the episodes, stacks of 3 blocks in 9%, and puts a block in the agent's gripper 50% of the episodes. This help aims at relaxing the exploration burden—which is orthogonal to our contributions. The social partner provides no help during offline evaluations.

6.2.2 Autotelic Reinforcement Learning

This section describes the implementation of the autotelic RL module in DECSTR. The policy takes as input the current state, the current semantic configuration and the goal configuration while the critic also receives the action. Both are trained with the soft-actor critic algorithm (SAC, Haarnoja et al., 2018b) and hindsight experience replay (HER, Andrychowicz et al., 2017). DECSTR samples goals via its curriculum strategy, collects experience in the environment, then performs policy updates with SAC. This section describes two particularities of our RL implementation: the self-generated goal selection

curriculum and the object-centered network architectures. Implementation details and hyperparameters can be found in Appendix C.3.

Goal Selection and Curriculum Learning

DECSTR only selects goals among the set of semantic configurations it has already discovered. We use an automatic curriculum strategy (Portelas et al., 2020b) inspired by the CURIOUS algorithm (Chapter 3). DECSTR tracks aggregated estimations of its *competence* (C) and *learning progress* (LP). Its selection of goals to target during data collection and goals to learn about during policy updates is biased towards goals associated with high absolute LP and low C.

Automatic bucket generation. LP is often computed on sets of goals with similar difficulties or similar dynamics to robustify the estimations (Forestier & Oudeyer, 2016a, and Chapter 3). While previous works leveraged expert-defined *goal buckets*, we cluster goals based on their time of discovery. The intuition is that the time of discovery is a good proxy for goal difficulty: agents discover easier goals earlier than difficult ones. Buckets are initially empty (no known configurations). When an episode ends in a new configuration, the $N_b = 5$ buckets are updated. Buckets are equally filled and the first buckets contain the configurations discovered earlier. Thus, goals change buckets as new goals are discovered.

Tracking competence, learning progress and sampling probabilities. Regularly, DECSTR evaluates itself on goal configurations sampled uniformly from the set of known ones. For each bucket, it tracks the recent history of past successes and failures when targeting the corresponding goals (last $W = 1800$ self-evaluations). C is estimated as the success rate over the most recent half of that history $\mathrm{C} = \mathrm{C}_{\mathrm{recent}}$. LP is estimated as the difference between $\mathrm{C}_{\mathrm{recent}}$ and the competence evaluated over the first half of the history ($\mathrm{C}_{\mathrm{earlier}}$). This is a crude estimation of the derivative of the C curve with respect to time: $\mathrm{LP} = \mathrm{C}_{\mathrm{recent}} - \mathrm{C}_{\mathrm{earlier}}$. The sampling probability P_i for bucket i is:

$$P_i = \frac{(1 - \mathrm{C}_i) \times |\mathrm{LP}_i|}{\sum_{j=1}^{N} (1 - \mathrm{C}_j) \times |\mathrm{LP}_j|}.$$

In addition to the usual LP bias, this formula favors goals of lower C when they have similar LP. The absolute value ensures that agents refocus on buckets that contain goals where their performance is decreasing (e.g. because of forgetting).

Object-Centered Architecture

Instead of fully connected or recurrent networks, DECSTR uses *object-centered architectures* adapted from *deep-sets* for the actor and critic (Zaheer et al., 2017) similar to the ones used in IMAGINE (Chapter 5). For each pair of objects, a shared network NN_{shared} independently encodes the concatenation of body features, objects features, current and target semantic configurations, see Figure 6.3. This shared network ensures an efficient transfer of skills between pairs of objects. A second inductive bias leverages the symmetry

of the behavior required to achieve *above*(o_i, o_j) and *above*(o_j, o_i). To ensure automatic transfer between the two, we present half of the features (e.g. those based on pairs (o_i, o_j) where $i < j$) with goals containing one side of the symmetry (all *above*(o_i, o_j) for $i < j$) and the other half with the goals containing the other side (all *above*(o_j, o_i) for $i < j$). As a result, the *above*(o_i, o_j) predicates fall into the same slot of the shared network inputs as their symmetric counterparts *above*(o_j, o_i), only with different permutations of object pairs. Goals are now of size 6: 3 *close* and 3 *above* predicates, corresponding to one side of the *above* symmetry. Skill transfer between symmetric predicates is automatically ensured. Appendix C.3.1 further describes these inductive biases and our modular architecture.

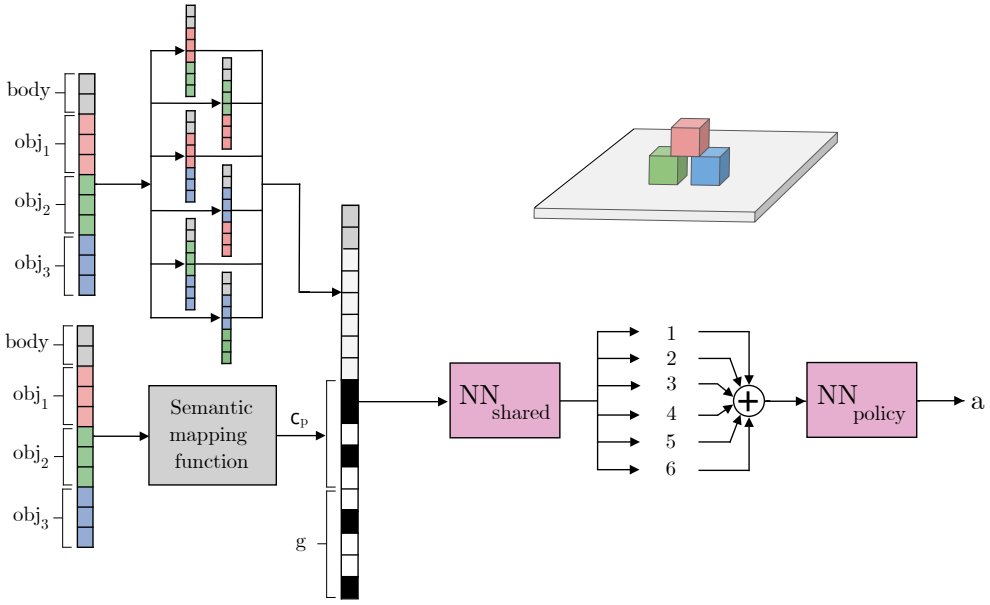


Figure 6.3: Object-centered modular architecture for the policy. The critic uses an equivalent architecture.

6.2.3 Language-Conditioned Goal Generation

The language-conditioned goal generation module (LGG) is a generative model of semantic representations conditioned on linguistic descriptions. It is trained to generate semantic configurations matching the agent's initial configuration and the description of a transformation in one object-pair relation. Thus, it answers: *if I perform the transformation d from configuration c_i, which configurations can I end up in?*

Dataset from Social Interactions

A training dataset is collected via interactions between a DECSTR agent trained in phase G→B and a social partner. DECSTR generates semantic goals and pursues them. For each trajectory, the social partner provides a description d of one change in objects relations from the initial configuration c_i to the final one c_f. The set of possible descriptions contains 102 sentences. Each describes a positive or negative shift for one of

the 9 predicates in a simplified language (e.g. *get red above green*). This leads to a dataset $\mathcal{D}$ of 5000 triplets: (c_i, d, c_f).

C-VAE for Language-Conditioned Goal Generation

We implement LGG with a conditional VAE (C-VAE) trained on this dataset (Sohn et al., 2015). Inspired by the context-conditioned goal generator from Nair et al. (2020), we add an extra condition on linguistic descriptions to improve agent's control on its goal generation. We encode the description with a recurrent network and train it jointly with the VAE to minimize a mixture of Kullback-Leibler and cross-entropy losses. Appendix C.3.2 provides the list of sentences and implementation details.

Strategy-Switching and Logical Combinations

From a single linguistic description, the agent can sample LGG repeatedly and obtain a set of matching goal configurations. This enables skill diversity and *strategy-switching*: if the agent fails, it can sample another valid goal to fulfill the description, effectively switching strategy. Because goal sets have bounded sizes, one can also compute goal configuration matching *logical combinations* of linguistic descriptions: *and* is an intersection, *or* is a union and *not* is the complement within the set of known goals. Here, logical combinations are expressed explicitly (e.g. AND('put red above green', 'put red above blue')), not implicitly (e.g. 'put red above blue and green').

6.2.4 Evaluation of the Three Phases

This section describes the three LGB phases and their evaluations.

Skill Learning Phase G→B

DECSTR explores its semantic representation space, discovers achievable configurations and learns to reach them. Each goal-specific performance is evaluated offline across learning as the success rate (SR) over 20 repetitions. The global performance $\overline{\texttt{SR}}$ is averaged over either the set of 35 valid goals or discovery-organized buckets of goals, see Section 6.2.2.

Language Grounding Phase L→G

DECSTR trains LGG to generate goals that match constraints expressed by linguistic descriptions. From a given initial configuration and a given description, LGG should generate all compatible final configurations (goals) and just these. This is the source of behavioral diversity and strategy-switching behaviors. To evaluate LGG, we construct a synthetic, oracle dataset $\mathcal{O}$ of triplets $(c_i, d, \mathcal{C}_f(c_i, d))$, where $\mathcal{C}_f(c_i, d)$ is the set of all final configurations compatible with (c_i, d). On average, $\mathcal{C}_f$ in $\mathcal{O}$ contains 16.7 configurations, while the training dataset $\mathcal{D}$ only contains 3.4 (20%). We are interested in two metrics:

1. *Precision*: the probability that a goal sampled from LGG belongs to $\mathcal{C}_f$ (ratio of true positive / all positive),

2. *Recall*: the percentage of elements from $\mathcal{C}_f$ that were found by sampling LGG 100 times (ratio of true positive / all true).

These metrics are computed on 5 different subsets of the oracle dataset, each calling for a different type of generalization (see full lists of instructions in Appendix C.3.2):

1. Pairs found in $\mathcal{D}$, except pairs removed to form the following test sets. This calls for the extrapolation of known initialization-effect pairs (c_i, d) to new final configurations c_f ($\mathcal{D}$ contains only 20% of $\mathcal{C}_f$ on average).

2. Pairs that were removed from $\mathcal{D}$, calling for a recombination of known effects d on known initial configurations c_i.

3. Pairs for which the c_i was entirely removed from $\mathcal{D}$. This calls for the transfer of known effects d on unknown initial configurations c_i.

4. Pairs for which the d was entirely removed from $\mathcal{D}$. This calls for generalization in the language space, to generalize unknown effects d from related descriptions and transpose this to known initial configurations c_i.

5. Pairs for which both the c_i and the d were entirely removed from $\mathcal{D}$. This calls for the generalizations 3 and 4 combined.

Instruction-Following Phase L→G→B

DECSTR is instructed to modify an object relation by one of the 102 sentences. Conditioned on its current configuration and instruction, it samples a compatible goal from LGG, then pursues it with its goal-conditioned policy. We consider four evaluation metrics:

1. *Transition*: evaluates the capacity of the agent to execute a single instruction. We report the average success rate when the agent is asked to execute each of the 102 instructions 5 times each, resetting the environment each time.

2. *Sequence*: evaluates the capacity of the agent to execute a sequence of instructions. We report the average length of a sequence of random instructions without reset before a failure occurs (20 sequences). An instruction is said *failed* when the agent cannot solve it with five attempts. These five attempts allow the agent to implement strategy switching.

3. *Expression*: evaluates the capacity of the agent to execute explicit logical combinations of instructions. We report the average success rate of the agent on 500 randomly generated logical expressions of sentences, see details of the generation mechanism in Appendix C.3.2.

4. *Behavioral Diversity*: evaluates the behavioral diversity of the agent. We report the average number of different achieved configurations that result from asking the agent to execute the 102 instructions 50 times each.

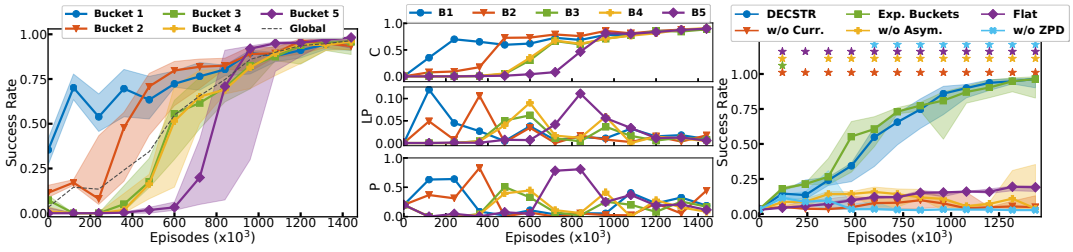
In the *transition* and *expression* setups, we evaluate performance in 1-shot ($\overline{\texttt{SR}}_1$) and 5-shot ($\overline{\texttt{SR}}_5$). In the 5-shot setting, the agent can perform *strategy switching* by sampling new goals when previous attempts failed (without reset).

6.3 Experiments

Our experimental section investigates three questions: 6.3.1): How does DECSTR perform in the three phases? 6.3.2): How does it compare to end-to-end language-conditioned approaches? 6.3.3): Do we need intermediate representations to be semantic?

6.3.1 Performance in the Three Phases

This section presents the performance of DECSTR in the skill learning, language grounding, and instruction following phases.



(a) (b) (c)

Figure 6.4: **Skill Learning**: (a) $\overline{\texttt{SR}}$ per bucket. (b): C, LP and P estimated by a DECSTR agent. (c): ablation study. Medians and interquartile ranges over 10 seeds for DECSTR and 5 seeds for others in (a) and (c). Stars indicate significant differences to DECSTR as reported by Welch's t-tests with $\alpha = 0.05$ (Appendix A).

Skill Learning Phase G→B

Figure 6.4 shows that DECSTR successfully masters all reachable configurations in its semantic representation space. Figure 6.4a shows the evolution of $\overline{\texttt{SR}}$ measures computed per bucket. Buckets are learned in increasing order, which confirms that the time of discovery is a good proxy for difficulty. Figure 6.4b reports C, LP and sampling probabilities P computed online using self-evaluations for an example agent. The agent leverages these estimations to select its goals: first focusing on the easy goals from bucket 1, it moves on towards harder and harder buckets as easier ones are mastered (low LP, high C). Figure 6.4c presents the results of ablation studies. Each condition removes one component of DECSTR:

1. *Flat* replaces our object-centered modular architectures with flat ones,
2. *w/o Curr.* replaces our automatic curriculum strategy with a uniform goal selection,
3. *w/o Sym.* does not use the inductive bias for symmetry,
4. *w/o SP* does not provide non-trivial initial configurations.

In the *Expert Buckets* condition, the curriculum strategy is applied on expert-defined buckets, see Appendix C.4.1. The full version of LGB performs on par with the *Expert Buckets* oracle and significantly outperforms all its ablations. Appendix C.5.1 presents more examples of learning trajectories.

As was proposed in Chapter 3, we bias the selection of goals to target (data collection) and goals to learn about (data exploitation, during hindsight replay). Here, we conducted an ablation study of these two different uses. Figure 6.5 shows we must use LP for both data collection and data exploitation. However, using any of these still leads to significant improvements over random goal selection.

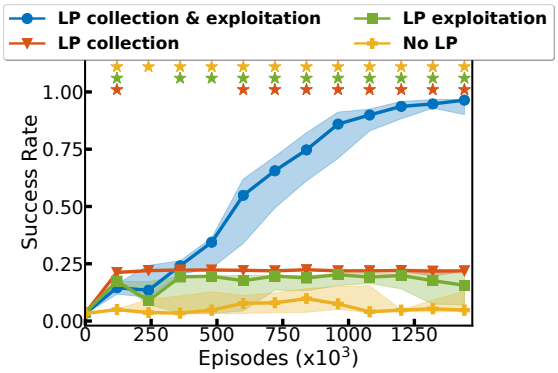


Figure 6.5: Ablation study of the use of learning progress. We use learning progress for data collection only, data exploitation only, both, or none of them. Only the combination of the two uses enables DECSTR to learn skills efficiently. Stars indicate significant differences with the application of LP for data collection and exploitation (DECSTR).

Figure 6.6 depicts the evolution of the buckets' contents across training (epochs 1, 50 and 100). Each pie chart corresponds to a reachable configuration and represents the distribution of configurations into buckets across 10 different seeds. Blue, orange, green, yellow, purple represent buckets 1 to 5 and grey are undiscovered configurations. At each moment, the discovered configurations are equally spread over the 5 buckets. A given configuration can change bucket as new configurations are discovered so that the ones discovered earlier are assigned buckets with lower indexes. Goals are organized by their bucket assignments in the *Expert Buckets* condition (from top to bottom).

After the first epoch (left), DECSTR has discovered all configurations from the expert buckets 1 and 2, and some runs have discovered a few other configurations. After 50 epochs, DECSTR has discovered new configurations but they are not always the same across runs. Finally, by epoch 100, all configurations have been discovered. Buckets are then stabilized and can be compared to expert-defined buckets. It seems that easier

goals (top-most group of the expert classification) are discovered first and assigned in the first-easy buckets (blue and orange). Hardest configurations (stacks of 3, bottom-most group) seem to be discovered last and assigned the last-hardest bucket (purple). In between, different runs show different compositions, which are not always aligned with expert-defined buckets. Goals from expert-defined buckets 3 and 4 (third and fourth group from the top) seem to be attributed different automatic buckets in different runs. This means that they are discovered in different orders depending on the runs. In summary, easiest and hardest goals from expert buckets 1–2 and 5 respectively seem to be well detected by our automatic bucket generations. Goals of medium expected difficulty as defined by expert buckets (3–4) seem not to show any significant difference in difficulty for our agents.

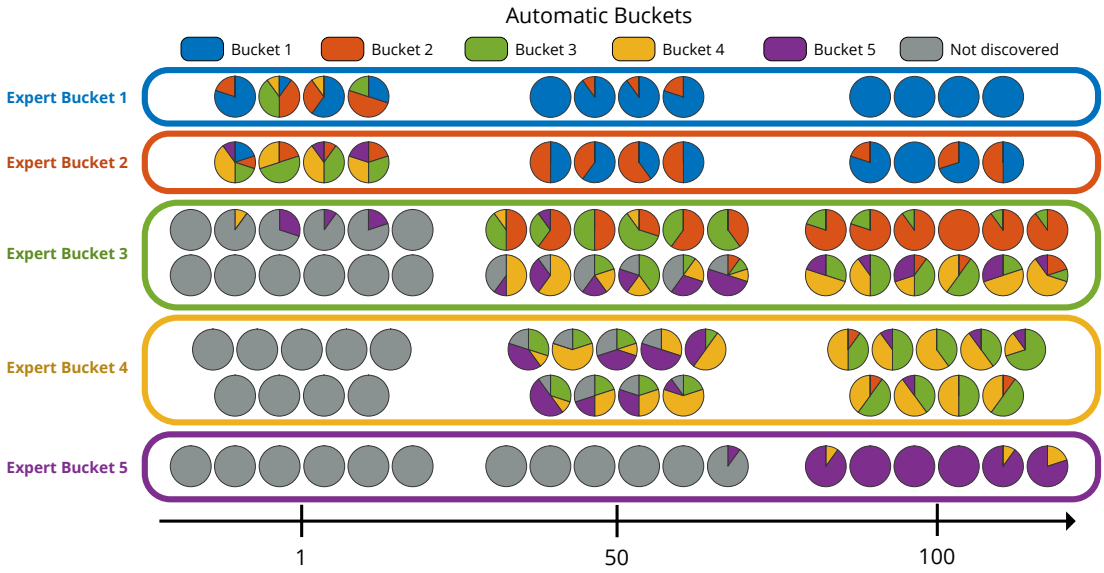


Figure 6.6: Evolution of the content of buckets from automatic bucket generation: epoch 1 (2.400 episodes, left), 50 (120.000 episodes, middle) and 100 (240.000 episodes, right). Each pie chart corresponds to one of the 35 valid configurations. It represents the distribution of the bucket attributions of that configuration across 10 runs. Blue, orange, green, yellow, purple represent automatically generated buckets 1 to 5 respectively (increasing order of difficulty) and grey represents undiscovered configurations. Goals are organized according to their expert bucket attributions in the *Expert Buckets* condition (top-bottom organization).

Language Grounding Phase L→G

LGG demonstrates the 5 types of generalization from Table 6.1. From known configurations, agents can generate more goals than they observed in training data (Tests 1, 2). They can do so from new initial configurations (3). They can generalize to new sentences (4) and even to combinations of new sentences and initial configurations (5). These results assert that LGG generalizes well in a variety of contexts and shows good behavioral diversity.

Table 6.1: L→G phase. Metrics are averaged over 10 seeds, stdev < 0.06 and < 0.07 respectively.

Metric	Test 1	Test 2	Test 3	Test 4	Test 5
Precision	0.97	0.93	0.98	0.99	0.98
Recall	0.93	0.94	0.95	0.90	0.92

Instruction-Following Phase L→G→B

Table 6.2 presents the one-shot and five-shot results in the *transition* and *expression* setups. In the sequence setups, DECSTR succeeds in $L = 14.9 \pm 5.7$ successive instructions (mean±stdev over 10 seeds). These results confirm an efficient language grounding. DECSTR can follow instructions or sequences of instructions and generalize to their logical combinations. Strategy-switching improves performance (SR_5 - $SR_1 > 0$). DECSTR also demonstrates strong behavioral diversity: when asked over 10 seeds to repeat 50 times the same instruction, it achieves at least 7.8 different configurations, 15.6 on average and up to 23 depending on the instruction.

Table 6.2: L→G→B phase. Mean ± stdev over 10 seeds.

Metric	Transition	Expression
SR_1	0.89 ± 0.05	0.74 ± 0.08
SR_5	0.99 ± 0.01	0.94 ± 0.06

Extension to Abstract Language

The 102 sentences only describe transformations in a single block relation. Instead, we want to know whether LGG can be trained to ground more abstract sentences such as '*make a pyramid.*' To this end, we added 25 abstract descriptions, including '*make a construction*', '*stack some blocks*' or '*put red on top*' (see full list in Appendix Table C.3). Here, we focus on the performance of LGG (precision and recall) on linguistic descriptions coming from the training set or from a testing set evaluating the linguistic generalization of LGG on these abstract descriptions (similar to Test 4).

The precision and recall on the training set are 0.83 ± 0.02 and 0.86 ± 0.02 respectively. In the testing set, the precision and recall are 0.16 ± 0.04 and 0.67 ± 0.13 when asking to '*get green on top*' or '*put green on top*' for the first time; 0.76 ± 0.03 and 0.84 ± 0.02 when asking to '*make a construction*', '*build a stack of two*' or '*make a stack of two*' for the first time. These results show that LGG can ground abstract language in lower-level semantic representations and even show forms of generalization.

6.3.2 Benefits of the Intermediate Representation

This section investigates the need for an intermediate semantic representation. To this end, we introduce an end-to-end language-conditioned RL variant mapping language to behavior directly (L→B) and compare its performance with DECSTR in the instruction following phase (L→G→B).

The LB Variant

To limit the introduction of confounding factors and under-tuning concerns, we base this implementation on the code of DECSTR, keeping the same HER mechanism, object-centered architectures and RL algorithm as DECSTR. We simply replace the semantic goal space with the 102 linguistic instructions. This variant can be seen as a version of IMAGINE with oracle reward function and without the imagination mechanism.

Comparison in the Instruction-Following Phase
L→B vs L→G→B

After training the LB variant for 1400K episodes, we compare its performance to DECSTR's in the instruction-following setup. In the *transition* evaluation setup, LB achieves $\text{SR}_1 = 0.76 \pm 0.001$: it always manages to move blocks close or far from each other but consistently fails to stack them. Increasing the number of attempts does not help: $\text{SR}_5 = 0.76 \pm 0.001$. The LB variant cannot be evaluated in the *expression* setup because it does not manipulate goal sets. Because it cannot stack blocks, LB only succeeds in 3.01 ± 0.43 random instructions in a row, against 14.9 for DECSTR (*sequence* setup). We then evaluate LB's diversity on the set of instructions it succeeds in. When asked to repeat 50 times the same instruction, it achieves at least 3.0 different configurations, 4.2 on average and up to 5.2 depending on the instruction against 7.8, 17.1, 23 on the same set of instructions for DECSTR. We did not observe *strategy-switching* behaviors in LB because it either always succeeds (close/far instructions) or fails (stacks).

Conclusion

The introduction of an intermediate semantic representation helps DECSTR decouple skill learning from language grounding, which facilitates instruction-following when compared to the end-to-end language-conditioned learning of LB. This leads to improved scores in the *transition* and *sequence* setups. The direct language-conditioning of LB prevents the generalization to logical combination and reduces behavioral diversity. Decoupling thus brings significant benefits to LGB architectures.

6.3.3 Benefits of a Semantic Intermediate Representation

This section investigates the need for the intermediate representation to be semantic. To this end, we introduce the LGB-C variant, a variant of DECSTR that leverages continuous goal representations in place of semantic ones. We compare them in the two first phases.

The LGB-C Variant

The LGB-C variant uses *continuous* goals expressing target block coordinates in place of semantic goals. The skill learning phase is equivalent to traditional goal-conditioned RL setups in block manipulation tasks (Andrychowicz et al., 2017; Li et al., 2020; Lanier et al., 2019, and Chapter 3). Starting from the DECSTR algorithm, LGB-C adds a translation module that samples a set of target block coordinates matching the targeted semantic configuration. The policy itself only experiences the concrete goal, never the semantic one. In addition, we integrate defining features of the state-of-the-art approach from Lanier et al. (2019): non-binary rewards (+1 for each block placed correctly) and multi-criteria HER (substituting targets for one, two or three blocks).

Because LGB-C is not aware of the original semantic goal, we cannot measure success as the ability to achieve it. Instead, *success* is defined as the achievement of the corresponding specific goal: bringing blocks to their respective targets within an error margin of 5 cm each. In short, DECSTR targets semantic goals and is evaluated on its ability to reach them. LGB-C targets specific goals and is evaluated on its ability to reach them. These two measures do not match exactly. Indeed, LGB-C sometimes achieves its specific goal but fails to achieve the original semantic goal because of margin errors.

Comparison in Skill Learning Phase G→B

The LGB-C variant successfully learns to discover and master the specific goals corresponding to all 35 semantic configurations. It does so faster than DECSTR: $708 \cdot 10^3$ episodes to reach SR$= 95\%$, against $1238 \cdot 10^3$ for DECSTR, see Figure 6.7. This can be explained by the denser learning signals it gets from using HER on continuous targets instead of discrete ones. However, in this phase, the agent only learns one parameterized skill: to place blocks at their target position. It cannot build a repertoire of semantic skills because it cannot discriminate between different block configurations. Looking at the sum of the distances traveled by the blocks or the completion time, we find that DECSTR performs opportunistic goal-reaching: it finds simpler configurations of the blocks which satisfy its semantic goals compared to LGB-C. Blocks move less ($\Delta_{\text{dist}} = 26 \pm 5$ cm), and goals are reached faster ($\Delta_{\text{steps}} = 13 \pm 4$, mean±std across goals with p-values $> 1.3 \cdot 10^{-5}$ and $3.2 \cdot 10^{-19}$ respectively).

Comparison in Language Grounding Phase L→G

We train LGG to generate continuous target coordinates conditioned on linguistic descriptions with a mixture of Kullback-Leibler and mean-squared losses and evaluate it in the same setup as DECSTR's LGG, see Table 6.3. Although it maintains reasonable precision in the first two testing sets, LGG achieves low recall—i.e. low diversity—on all sets. This can be explained by the additional difficulty posed by the generation of continuous data in place of discrete ones. However, the recent DALL-E model showed that the generation of high-dimensional data (here images) conditioned on language inputs is not how of reach, with sufficient data (Ramesh et al., 2021).

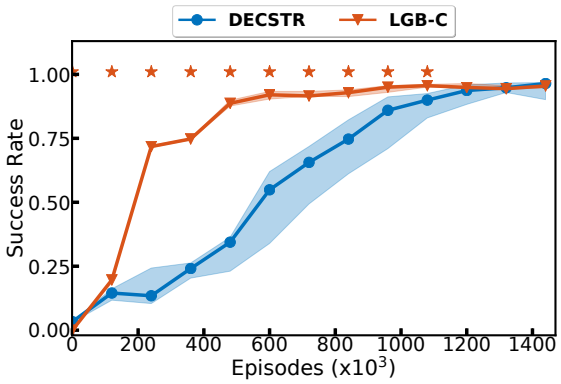


Figure 6.7: Comparison DECSTR and LGB-C in the skill learning phase. Note that both success rates are a bit different: DECSTR is evaluated on its ability to reach semantic goals; LGB-C is evaluated on its ability to reach—up to a tolerance margin—specific goals that correspond to semantic goals. In the latter case, the resulting configurations might not match the original semantic goal because of the tolerance margins.

Table 6.3: LGB-C performance in the L→G phase. Mean over 10 seeds. Stdev < 0.003 and 0.008 respectively.

Metrics	Test 1	Test 2	Test 3	Test 4	Test 5
Precision	0.66	0.78	0.39	0.0	0.0
Recall	0.05	0.02	0.06	0.0	0.0

Conclusion

The skill learning phase of the LGB-C variant is competitive with the one of DECSTR. However, the poor performance in the language grounding phase prevents this variant to execute instructions. For this reason, and because semantic representations enable agents to perform opportunistic goal reaching and to acquire repertoires for semantic skills, we believe the semantic representation is an essential part of the LGB architecture.

6.4 Discussion

This chapter contributes LGB, a new conceptual RL architecture introducing an intermediate semantic representation to decouple sensorimotor learning from language grounding. To demonstrate its benefits, we present an instance called DECSTR. DECSTR discovers and masters all reachable configurations in a manipulation domain from a set of relational spatial primitives, before undertaking an efficient language grounding phase. This was made possible by the use of object-centered inductive biases, a new form of automatic curriculum learning and a novel language-conditioned goal generation module. Note that our main contribution is in the conceptual approach, DECSTR being only an instance to showcase its benefits. We believe that this approach could benefit from any improvement in goal-conditioned RL (for skill learning) or generative models (for language grounding).

Semantic Representations

Results have shown that using predicate-based representations was sufficient for DECSTR to learn abstract goals opportunistically. The proposed semantic configurations showcase promising properties: 1) they reduce the complexity of block manipulation where most effective works rely on a heavy hand-crafted curriculum (Li et al., 2020; Lanier et al., 2019) and specific curiosity mechanisms (Li et al., 2020); 2) they facilitate the grounding of language into skills and 3) they enable decoupling skill learning from language grounding, as observed in infants (Piaget, 1977).

Defining the list of semantic predicates is defining the dimensions of the behavioral space explored by the agent. It replaces the traditional definition of goal spaces and their associated reward functions. We believe it is for the best, as it does not require the engineer to fully predict all possible behaviors within that space, know which behaviors can be achieved and which ones cannot, or define reward functions for each of them. Although the predicates are assumed innate in this study, further work could investigate how to learn them from data.

A New Approach to Language Grounding

The approach proposed here is the first that simultaneously enables agents to decouple skill learning from language grounding and fosters a diversity of possible behaviors for any given instruction. Indeed, while an instruction-following agent trained on goals like *put red close to green* would just push the red block towards the green one, our agent can generate many matching goal configurations. It could build a pyramid, make a blue-green-red pile or target a dozen other compatible configurations. This enables it to *switch strategy* and find alternative approaches to satisfy the same instruction when the first attempts failed. Our goal generation module can also generalize to new sentences or transpose instructed transformations to unknown initial configurations. Finally, with the goal generation module, the agent can deal with any logical expression made of instructions by combining generated goal sets. Future work could investigate parallel skill learning and language grounding phases that would result in *overlapping waves* of sensorimotor and linguistic development (Siegler, 1998).

Semantic Configurations of Variable Size

Considering a constant number of blocks and fixed-size configuration spaces is a current limit of DECSTR. As the number of blocks and the number of predicates grows, the semantic representation space grows combinatorially. In our ongoing work, we replace the deep-set architecture with a *graph neural network* that better handles object sets and their relational representations. This helps us solve tasks with 3 blocks quicker without resorting to the initialization help from the social partner or to any form of curriculum. This approach also seems to scale to more cubes (up to 5 in the current implementation). To handle exploding configuration spaces, we consider *masked goals* made of of a subset of all possible predicates. Indeed, our current representations can be seen as a logical AND applied on all semantic predicates possessed by the agent. A more general representation

of goals could encode them as *logical functions* of predicates instead, augmenting the flexibility of goal representations and avoiding the combinatorial explosion resulting from the consideration of all predicates at once.

Grounding Abstract Language From Semantic Predicates

In line with the vision of Spelke (Spelke, 2003), we found that LGG could ground abstract language (e.g. 'make a pyramid') by building such abstract linguistic representations on top of innate semantic predicates (here spatial ones). Our innate semantic representations encode some basic abstractions in terms of spatial relations between objects and can be seen as a simple model of the object core system (Spelke & Kinzler, 2007). From these simple abstractions, and thanks to LGG, language can be used to represent even greater abstractions (e.g. pyramids, constructions).

Conclusion

This work has shown that abstract representations based on relational predicates can serve as a pivot between skill learning and linguistic social interactions. Here, the role of the social partner was limited to 1) helping the agent to experience non-trivial configurations and 2) describing the agent's behavior in a simplified language. In the future, we intend to study more intertwined skill learning and language grounding phases, making it possible for the social partner to teach the agent during skill learning.

Part Summary

We opened the second part of this manuscript on the idea of using language as a communicative and cognitive tool to structure the learning process of autotelic learning agents. This view was pioneered by Vygotsky in the 1930s, then further developed across many fields. Learners first experience language in the context of social interactions with more knowledgeable peers, then slowly turn it into internal psychological tools to direct and control their thought processes. In our quest for *Vygotskian autotelic agents*, we reviewed existing uses of language in reinforcement learning agents for communication, abstraction, generalization and planning. However, we noticed that no RL agent uses language as a cognitive tool to power imagination, creativity or semantic mental simulations (Chapter 4). We proposed two such implementations in our two experimental chapters (Chapters 5 and 6). IMAGINE integrates descriptive feedback from a social partner to represent its own goals and reward functions. As it interacts with the social partner, IMAGINE builds a crude model of the grammar and uses it to generate counterfactual possibilities, new goals composed from pieces known ones that it uses to self-organize exploratory behaviors (Chapter 5). The LGB architecture integrates descriptive feedback from a social partner to learn a language-conditioned model of the world mapping linguistic descriptions to possible future configurations (Chapter 6). This linguistic world model allows agents to simulate a diversity of configurations from any description, which leads to improved behavioral diversity and strategy-switching behaviors.

Part III

Discussion

It is now time to take a step back and discuss our conceptual and empirical contributions. We organize this discussion into three short chapters. The first chapter summarizes our contributions and discusses them in the light of a shortlist of recent relevant approaches (Chapter 7). The second acknowledges the limitations of our proposed implementations and discusses possible ways to overcome them to design autotelic RL agents at scale (Chapter 8). The last chapter (Chapter 9) comes back to the *Vygotskian autotelic agents* proposition introduced in Chapter 4. Like children, autotelic agents should evolve in a rich socio-cultural environment where social partners provide help, explanations, advice and participate in shaping their zone of proximal development. As agents interact with social partners, they should learn to turn interpersonal social feedback into intrapersonal psychological tools (Vygotsky, 1934). As they learn to generate feedback for themselves, they turn the external socio-cultural life into an inner mental life (Vygotsky, 1934).

Chapter 7

Summary and Recent Related Work

This chapter reviews our contributions and integrates them in the light of a shortlist of recent works.

7.1 Summary of the Contributions

This research aims at making progress on the problem of the autonomous acquisition of open-ended repertoires of skills. We took insights and mechanisms from developmental robotics, transposed, adapted and extended them to work with state-of-the-art optimization methods from the field of deep reinforcement learning. This resulted in a new computational framework at the crossroad of developmental robotics and reinforcement learning: *rl-based intrinsically motivated goal exploration processes* (RL-IMGEP). RL-IMGEPs use deep reinforcement learning to train autotelic agents implemented by goal-conditioned policies to represent, generate, pursue and master their own goals.

We organized our contributions into two parts. The first part focused on autotelic reinforcement learning agents. Chapter 1 had two objectives: 1) to cover our inspirations and review related works from the fields of psychology, developmental psychology, robotics and reinforcement learning and 2) to introduce generalized definitions of *goals*, *skills* and the RL-IMGEP framework. RL-IMGEPs target the *autotelic reinforcement learning problem*; they evolve in a reward-free MDP and need to represent, target and master their own goals to build repertoires of skills. The transposition of this objective from developmental robotics to the RL framework and the extension of the IMGEP family to RL-based IMGEPs make our first contribution.

We support this conceptual contribution with two empirical chapters. The study of GEP-PG and ME-ES, the learning algorithms introduced in Chapter 2, showed that pursuing self-generated goals is an efficient way to organize exploration in external tasks characterized by sparse or deceptive rewards. In the study of CURIOUS, the first RL-IMGEP, we showcased the use of learning progress and modular architectures to organize the acquisition of a diversity of skills involving different affordances (Chapter 3). Whereas existing POP-IMGEP implementations could hardly handle a diversity of initial states and did not scale to continuous manipulation tasks, our RL-based IMGEPs successfully learned to master multiple types of goals and transfer knowledge across skills.

The second part of this manuscript extended RL-IMGEPs with linguistic abilities.

Building on the pioneering work of Vygotsky (Vygotsky, 1930, 1933, 1934) and the following developments in psychology (Gentner, 1983; Gentner & Goldin-Meadow, 2003; Berk, 1994; Spelke, 2003), philosophy (Clark, 1998a; Clark et al., 1998; Clark, 2006), linguistic (Rumelhart et al., 1986; Lupyan, 2012) and developmental robotics (Dautenhahn, 1995; Zlatev, 2001; Lindblom & Ziemke, 2003; Mirolli & Parisi, 2011), Chapter 4 insisted on the centrality of *social situatedness* in the cognitive development of autotelic agents. This led us to advocate for *Vygotskian autotelic agents*. Like children, autotelic agents must evolve in a rich socio-cultural environment and leverage language as both a communicative and cognitive tool to structure their behavior. Agents must learn to integrate the help provided by caretakers in social situations into internal psychological tools for themselves. After reviewing existing goal-conditioned RL approaches using language, we identified a gap: no RL agent uses language to power creative imagination or condition structured world models.

We supported this conceptual contribution with two empirical chapters. IMAGINE, the learning algorithm introduced in Chapter 5, uses both the communicative and cognitive functions of language. It first infers abstract goal representations and reward functions from linguistic interactions with a simulated social partner (communicative function), then uses linguistic productivity to imagine creative goals by composing known ones (cognitive function). This cognitive use of language enhances the exploratory and systematic generalization abilities of the agent. Finally, our last study (LGB) kept the idea of learning language from social interactions but formalized language grounding as a mapping from language to concrete goals—as opposed to the traditional language to behavior or language to rewards. This special type of grounding occurred within a language-conditioned goal generator, a module enabling agents to generate sets of possible futures conditioned on linguistic descriptions (Chapter 6). Aking to mental imagery in humans, this ability favors behavioral diversity and enables strategy-switching behaviors. IMAGINE and LGB are RL-IMGEPs using language as a cognitive tool; they are *Vygostkian autotelic RL agents*.

From the first algorithm to the last, we removed external components one by one, offering more and more autonomy and creativity to our agents. GEP-PG and ME-ES conduct autotelic exploration at the service of an external task. CURIOUS explores for its own sake; targets its own goals and crafts its own learning trajectory. Going further, IMAGINE gets rid of the pre-coded goal representations and reward functions assumed innate in CURIOUS by leveraging linguistic social interactions. DECSTR, although it assumes some innate semantic representations (binary spatial relations), learns higher-level ones (e.g. pyramids) from social interactions. This progression was made possible by the introduction of deep reinforcement learning methods. It is important to note that the use of DRL is essential to the linguistic capabilities of our agents, just like it is essential to their abilities to handle high-dimensional inputs, various initial conditions or procedurally-generated scenes. Indeed, existing POP-IMGEP implementations have trouble scaling to higher-dimensional tasks and training high-dimensional controllers or, at least, would require extensive training data to achieve comparable results (see ME-ES in Chapter 2).

7.2 Recent Related Work

This section discusses a shortlist of recent related work and how they relate to our contributions.

Autotelic Exploration with Go-Explore

Last year, *Go-Explore* achieved the major milestone of solving the infamous Montezuma's Revenge game from the Atari suite (Ecoffet et al., 2021). Although the last version solves all 57 Atari games with super-human performance, the first release focused on solving two games where previous approaches dramatically failed: Montezuma and Pitfall. These games are respectively known for their sparse and deceptive rewards. Previous attempts involved knowledge-based intrinsic motivations: e.g. agents were rewarded for experiencing novel states (Bellemare et al., 2016) or for maximizing errors in their learned prediction models (Pathak et al., 2017; Burda et al., 2019). As the authors of Go-Explore argue, this leads to a phenomenon called *detachment*: agents lose sight of interesting areas to explore as represented in Figure 7.1.

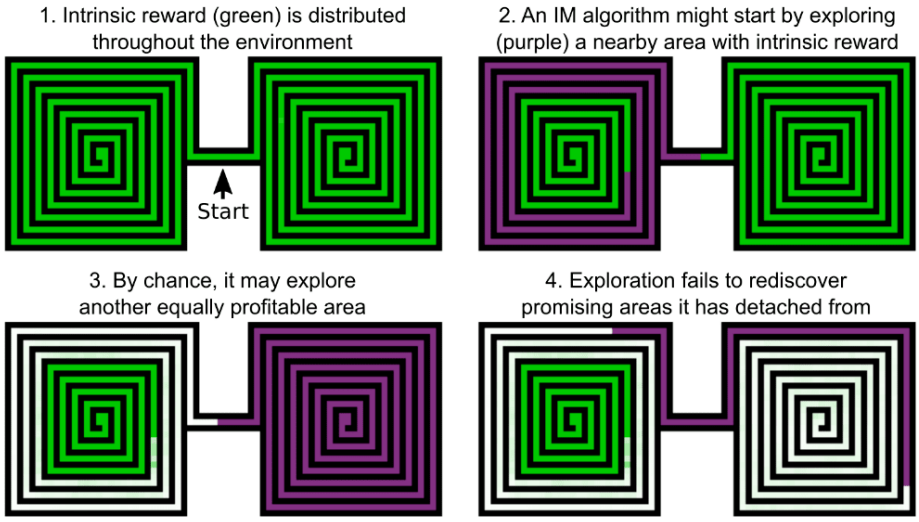


Figure 7.1: Example of detachment caused by knowledge-based intrinsic motivations. Green and white respectively indicate high and low intrinsic motivation. Purple indicates regions the algorithm currently explores, from Ecoffet et al. (2021).

Their solution is to implement the following algorithm (Ecoffet et al., 2021):

1. Discretize the state space into behavioral cells,
2. Select a cell from the archive,
3. Return to that cell,
4. Explore from that cell,
5. Add newly discovered cells to the archive,
6. Loop back to 2.

We can make two parallels between Go-Explore and the algorithms presented in this manuscript. First, it decouples exploration and exploitation, just like GEP-PG and ME-ES (Chapter 2). Here, agents *exploit* when they maximize their cell-oriented rewards—i.e. try to reach their targeted cell without exploration—and *explore* from there with random actions. "Go" and "explore" can be translated to "exploit" then "explore." Whereas GEP-PG and ME-ES decouple exploration and exploitation across episodes, Go-Explore decouples them within the same trajectory. This idea is simple but extremely powerful because agents better find their way back to known cells. Indeed, without this returning phase, agents could lose their way back by trying to explore and exploit at the same time, a problem called *derailment* (Ecoffet et al., 2021). Reaching known cells and exploring from there is a good way to discover *stepping stones* towards skills involving sequential combinations of sub-skills. This is especially true in navigation tasks, where the skill of reaching a cell is always composed of sub-skills reaching cells along the way.

The second parallel is even more important: Go-Explore conducts an autotelic exploration and, thus, is a RL-IMGEP. Indeed, Go-Explore replaces the use of knowledge-based IMs with the use of competence-based IMs. When it targets cells, it targets goals made of both a goal embedding (the cell description) and a goal-achievement function (whether descriptions of the target and current cells match). Downsampling acts as the high-level representation function turning states into what we called *outcomes* or *behavioral characterizations* in IMGEP and QD approaches. The agent samples a goal, then tries to reach it. Here, the setup is similar to GEP-PG and ME-ES. Agents conduct an autotelic exploration (reaching self-generated cells) as an auxiliary task to facilitate the resolution of a pre-defined hard-exploration task (solving the game). Just like any other RL-IMGEP, Go-Explore needs to define its outcome representations. Here, the authors either hand-code frame downsampling or use engineered features as outcome representations but aim at learning them from data in the future (Ecoffet et al., 2021).

Unsupervised Skill Discovery

Unsupervised skill discovery is a recent line of work tackling the autonomous acquisition of repertoires of skills (Gregor et al., 2017; Eysenbach et al., 2019; Achiam et al., 2018; Campos et al., 2020; Sharma et al., 2020). Although these algorithms meet the requirements of the RL-IMGEP framework, they aim to explore the state space directly:

> "*Making progress on this problem requires specifying a learning objective that ensures that each skill individually is distinct and that the skills collectively explore large parts of the state space.*" (Eysenbach et al., 2019)

This line of work takes inspiration from the notion of *empowerment* and its mathematical formulation as the maximization of the mutual information between skill representations and behaviors (Salge et al., 2014; Jung et al., 2011; Mohamed & Rezende, 2015). The general idea is to maximize both the discriminability of the skills—different skills must visit different states—and the diversity of states visited by each skill (Gregor et al., 2017; Eysenbach et al., 2019; Achiam et al., 2018; Campos et al., 2020; Sharma et al., 2020). This is done by training in parallel a discriminator to recognize skills from states and

skill policies to visit distinct states with maximum entropy. Approaches then differ in their specific implementations of the discriminator (taking as input the terminal state (Gregor et al., 2017), any state (Eysenbach et al., 2019; Campos et al., 2020), whole trajectories (Achiam & Sastry, 2017) or transitions (Sharma et al., 2020)) and the use of a *reverse* (Gregor et al., 2017; Eysenbach et al., 2019; Achiam et al., 2018) or *forward* (Sharma et al., 2020; Campos et al., 2020) decomposition of the mutual information (see a discussion of these approaches in Campos et al., 2020).

The utility of the learned skills is often evaluated in downstream tasks including imitation learning of expert trajectories, hierarchical control in hard-exploration tasks, or fine-tuning on an evaluation task. All these approaches focus on high-dimensional control tasks from the MUJOCO library (Todorov et al., 2012), a set of environments where agents can only navigate. This choice can appear surprising: if we want to discover skills, why not consider manipulation tasks with richer affordances? The answer might be that the set of meaningful skills to be discovered in navigation tasks is simple and well defined (e.g. going to various places, going backward and forward). In contrast, the set of meaningful skills in manipulation tasks can be harder to discover (e.g. picking blocks, stacking them, using tools).

Because they explore the state space, unsupervised skill discovery approaches can learn trivial skills discriminated by meaningless features. For instance, two skills can drive the agent in the same direction but use distinct configurations of the joints. In manipulation tasks, the angular position of blocks might be enough to discriminate between skills. The task becomes even more challenging in vision-based procedurally-generated environments, where agents see new scenes every episode.

We argue that the problem comes from exploring the state space directly, a space lacking abstraction. In contrast, RL-IMGEPs explore a learned high-level outcome space that can abstract away low-level features of the states. In practice, unsupervised skill discovery approaches often integrate such biases to enhance performance. For example, one can restrict the diversity search to a subset of the state space (e.g. the x-y position of the agent Eysenbach et al., 2019; Sharma et al., 2020). One exciting avenue for future work is to combine ideas from unsupervised skill discovery and RL-IMGEPs. Agents could learn high-level representation spaces influenced by social interactions with caretakers before running an unsupervised skill discovery algorithm to explore the resulting linguistic embedding space and form skill representations.

Reinforcement Learning with Natural Language

Recent advances in language-conditioned RL can scale to real, human-generated instructions instead of relying on templated languages. Hill and colleagues, for instance, achieved zero-shot generalization from templated language to human language by augmenting the templated data with typo-noise (missed keys) and encoding it with pre-trained language models using context-sensitive word embeddings (Hill et al., 2020b).

In *Grounding Language From Play* (Lynch & Sermanet, 2020), agents are trained to achieve manipulation goals expressed either as target images or linguistic instructions. Both visual and linguistic goals are mapped to a shared multi-context latent space. The whole architecture is trained with goal-conditioned imitation learning from a dataset of

unsegmented demonstrations obtained by human teleoperation. Because the policy can be trained to target either visual or linguistic goals, it is mostly trained to pursue visual goals that do not require expensive human-generated linguistic labels. To complement this extensive self-supervised learning, language grounding only necessitates human-generated descriptions for one percent of the trajectories. This efficient method trains agents to handle high-dimensional inputs with relatively small amounts of human-generated feedback. Autotelic variants of this approach would generate their linguistic goals and learn from their own trajectories and social partner's descriptions via self-imitation learning. Whereas LGB translates linguistic descriptions into concrete goals (Chapter 6), this approach maps both linguistic descriptions and concrete goals to a shared latent embedding. Generating goals from linguistic descriptions can be more difficult but allows the mental representations of alternative futures that are key to control behavioral diversity.

Another interesting approach from Hill and colleagues aims at modeling the *fast-mapping* behaviors observed in children Hill et al. (2021). Fast-mapping refers to the ability to bind a new word to an unknown object from a single exposure. To this end, agents are given a differentiable multi-modal episodic memory. In a *discovery phase*, agents interact with objects to obtain their names and store them in episodic memories. In the *instruction phase*, agents are asked to pick up an object. Thanks to their episodic memory, they can demonstrate *fast-mapping* by transferring pick-up skills from known objects to new ones. This ability to adapt quickly to new environments within the same life is a crucial feature of animal learning and should be integrated into RL algorithms.

Chapter 8

Scaling Autotelic Agents

This chapter acknowledges the technical limits of our proposed RL-IMGEP agents and discusses how we could overcome them in future implementations. We primarily focus on scaling to more realistic inputs and learning to represent and master a richer diversity of goals and skills.

8.1 More Realistic Inputs

Empirical Chapters 2, 3, 5 and 6 have focused on environments characterized by crafted input states, full observability and simple templated languages. The objective was to define controlled environments to disentangle the study of the properties of our mechanisms from the orthogonal problems of perception, long-term memory, challenging exploration or fully-blown language understanding. However, it is crucial to scale these architectures if we ever want our agents to learn from human feedback in our social world.

The manipulation tasks introduced in Chapters 3, 5 and 6 all used crafted state features characterizing the position, orientation and velocity of the objects. Object-based representations are known to develop early in human infants (Johnson et al., 2003; Mandler, 2004; Spelke, 2003; Spelke & Kinzler, 2007; Chen, 2012) and facilitate systematic generalization and control when used in combination with relation architectures (object-body relations in IMAGINE, object-object-body in DECSTR and Karch et al., 2020). However, a more general approach would be to learn directly from lower-level sensory inputs such as vision. Here, we can consider two alternatives. One way is to learn everything end-to-end without enforcing object-based representations. This often works well in navigation tasks where goals consist in finding specific objects (e.g. Hermann et al., 2017; Chevalier-Boisvert et al., 2019) but can also work in manipulation tasks with high-quality demonstrations (Lynch & Sermanet, 2020). The alternative is to convert visual inputs to object-based representations. Recent approaches propose to replace the hand-coded feature extractors of the pre-deep-learning era with unsupervised learning algorithms extracting object representations from images (e.g. Burgess et al., 2019; Greff et al., 2019, 2020; Ding et al., 2020).

Usually coupled with visual input, *partial observability* is another challenge. Indeed, vision-based agents often receive first-person views of their environment. This type of input breaks the Markov assumption: previous observations can contain extra information

not accessible from the current observation. To overcome this issue, agents can be conditioned on state representations integrating information from past observations. For example, recurrent networks encode traces of past observations sequentially and maintain a time-extended representation in their hidden state (e.g. Chevalier-Boisvert et al., 2019; Badia et al., 2020a). Transformers, on the other hand, are conditioned on the whole trace directly (e.g. Loynd et al., 2020). Whereas current approaches receive external rewards and only use time-extended architectures to implement policies and value functions, RL-IMGEP implementations also need to represent time-extended internal reward functions. Interestingly, a recent study found that first-person views facilitates systematic generalization (Hill et al., 2020a). A reason might be that agents with an egocentric view can better align linguistic descriptions and visual states because they can isolate single objects.

In the real world, IMAGINE and DECSTR must work with natural human-generated language. As discussed in the previous chapter, pre-trained language models and demonstrations can help (Hill et al., 2020b; Lynch & Sermanet, 2020). A series of recent work use labeled human-generated demonstrations. Abramson and colleagues train a policy with *generative adversarial imitation learning* and additional auxiliary losses from a large dataset of human-generated interactions (Abramson et al., 2020). Zhou and colleagues infer a reward function with inverse RL methods and train a captioning system to provide more labels (Zhou & Small, 2020). Nguyen and colleagues model a social partner providing descriptions and use its descriptions to perform goal-conditioned self-imitation learning (Nguyen et al., 2021). Our developmental approach prevents us from using pre-trained language models but learning from labeled demonstrations and training internal captioning systems are both compatible with our requirements for Vygostkian autotelic agents. We will discuss these aspects further in Chapter 9. Finally, an original idea is to train a *translator* module to project natural language into the simple templated language used for training; a sort of *simulation-to-real* setting (Marzoev et al., 2020).

In addition to representing goals in natural language, IMAGINE must also *imagine goals in natural language*. How would that work? Our current implementation is crudely inspired by usage-based language learning theories. (Tomasello & Olguin, 1993; Tomasello, 2000a; Goldberg, 2003). The discovery and substitution of equivalent words in learned schemas is indeed observed directly in studies of child language (Tomasello & Olguin, 1993; Tomasello, 2000a). Andreas and colleagues even used similar mechanisms to generate valuable data augmentations leading to increased performance in natural language processing tasks (Andreas, 2020). As we use more natural language, the detection of templates must be improved. We can compute equivalences between sub-phrases instead of words alone. We can compute non-binary equivalence scores to better control the temperature of goals imagination: higher temperature replaces phrases with lower equivalence scores, which leads to more original goals. Computational models of construction grammar theories (Steels, 2011; Steels & Van Trijp, 2011) can be helpful tools for tracking linguistic constructions and generating new creative utterances. Indeed, some researchers already investigate the use of construction grammar theories and computational models for linguistic creativity (Hoffmann, 2018; Bergs, 2019; Hoffmann, 2020; Turner, 2020). As we generate more creative goals, we need to implement filtering systems to estimate the probability of success or to prune out goals that seem impossible. One could, for example, use learning progress measures to focus on controllable imagined goals. An alternative is

to use ensembles of reward functions and focus on imagined goals where reward functions agree, i.e. where we hope they generalize well.

Sample inefficiency is another limit of our implementations, although we could consider it a current limitation of model-free RL as a whole. Indeed, our agents require millions of interaction steps to learn to behave, which practically limits their use in the real world with humans. Our learning architectures are not tied to particular optimization algorithms and can quickly benefit from any improvement in off-policy RL or generative modeling. An idea could be to design model-based versions of our learning architectures. In model-based RL, agents learn a model of their environment (e.g. a model of its dynamics) and use it to learn faster from fewer interactions (Schmidhuber, 1990; Sutton, 1991; Dayan et al., 1995; Nguyen-Tuong & Peters, 2011; Hamrick et al., 2020). Recent improvements in model-based techniques now lead to both higher sample efficiency and higher asymptotic performance in many benchmarks, including continuous control (Chua et al., 2018; Ha & Schmidhuber, 2018; Ebert et al., 2018; Sekar et al., 2020; Charlesworth & Montana, 2020), board games (Schrittwieser et al., 2020) or Atari (Hafner et al., 2021). We could also use dynamics models to simulate trajectories towards imagined goals as a way to assess their relevance. With good world models, IMAGINE could correct its over-generalization and learn about imagined goals without any additional interaction in the environment.

8.2 A Richer Diversity of Goals

Existing works consider several types of goal representations (see a typology in Section 1.2) but still cover a small fraction of the goals humans can target. Future RL-IMGEPs must learn to represent (learn embeddings and reward function) and master (learn policies) a richer diversity of goals and skills.

Representing a Richer Diversity of Goals

Current RL-IMGEP approaches only consider *time-specific* goals, goals whose completion can be assessed from any state s. Just like in partially observable settings, time-extended goals require forms of memory. To handle the time-extended *grow* goals of IMAGINE, we had to include the initial observation in the state of the agent to give it a crude form of memory (Chapter 5)). Alternatives could involve recurrent networks, transformers or episodic memories. In addition to the existing policy and value function architectures (e.g. Chevalier-Boisvert et al., 2019; Hill et al., 2019; Loynd et al., 2020), we must also train time-extended reward function architectures to represent goals. Note that POP-IMGEP and QD approaches, because they rely on episodic optimization algorithms, are not limited by the Markov assumption and can directly target time-extended goals (e.g. Forestier & Oudeyer, 2016b, or methods from Chapter 2).

Besides task goals, humans also target *learning goals* (Ram et al., 1995). Whereas task goals express constraints on the physical state of the world, learning goals express constraints on the agent's knowledge. Agents can target learning goals as a way to organize the collection of information or help the realization of future task goals. If a person wants to fix his oven, maybe he will need to learn about electronics first. Learning-progress-based learning, for example, can be understood as a learning goal. As the agent

favors goal regions to sample task goals, it formulates the goal of learning about specific goal regions (e.g. approaches from Chapters 3 and 6). Another type of learning goals are *questions*. In *embodied question answering*, agents are trained to explore the world and answer questions about it (Das et al., 2018; Yuan et al., 2019). In the future, RL-IMGEP must learn to handle both physical and learning goals seemingly.

An idea to express a potentially infinite goal space is to formulate them as logical combinations of predicates characterizing aspects of the world. DECSTR implements a simple version of this where each predicate characterizes the presence or absence of a spatial relation between two blocks (e.g. red above blue) and goals are formulated as the logical AND function applied on the values of all predicates (Chapter 6). Predicates can be seen as simple binary classifiers asserting the veracity of a property in the current state of the world (e.g. "*the door is open*"). As the number of predicates grows, and given recursive and combinatorial logical combinations of these predicates, the set of potential goals becomes virtually infinite. As this set grows, however, agents need to derive heuristics to constrain goal selection. Indeed, the simple goal representation used in DECSTR already contains 512 possible configurations (9 binary predicates), among which only 35 can be reached. As a heuristic, DECSTR only samples goals from the set of experienced configurations. Ideally, the agent should learn predicates from experience and social interactions. One could imagine learning binary sensors as features of a discrete latent code in the pipeline of a social task. For instance, we could train a vector-quantized VAE to generate linguistic descriptions of trajectories fitted on descriptions provided by a social partner (van den Oord et al., 2017).

Autotelic agents should be able to handle multiple goal spaces at once. This can take the form of goal spaces organized hierarchically at different levels of abstractions. This hierarchical organization can serve *temporal composition* such that low-level goals can be sequentially combined to reach higher-level goals (see a discussion of hierarchical RL in the next section). Meta-diversity can be another approach (Etcheverry et al., 2020). In the outer-loop of the meta-diversity search, one aims at learning a diverse set of behavioral characterizations. In the inner-loop, the exploration mechanism aims at generating a diversity of behaviors in each existing behavioral space. In their paper, Etcheverry and colleagues train a hierarchical VAE to learn representations of synthetic lifeforms at various scales. As the diversity search stagnates (inner loop), the algorithm splits the representation space in two and grows two branches to represent the two halves of the specimens at finer scales (outer-loop). One could imagine a similar approach to learn goal representations in a DRL context. As agents learn to master skills covering existing goal spaces, an outer-loop aims at generating new goal spaces to challenge the agent. Different goal spaces focus on different aspects of the environment and each is associated with a corresponding goal-conditioned reward function.

Humans express most of their abstract goals through language, and artificial agents should do it too. As argued in Chapter 4, language offers abstraction and facilitates systematic generalization. It can express time-extended and time-specific goals, physical and learning goals, binary predicates and their logical combinations.

Mastering a Richer Diversity of Goals

An efficient way to target a wide variety of goals is to combine existing skills into more complex ones (Todorov, 2009). Skills can be combined concurrently (Saxe et al., 2017; Haarnoja et al., 2018a; van Niekerk et al., 2019; Hunt et al., 2019; Peng et al., 2019; Tasse et al., 2020) or sequentially. Logical composition is concurrent and combines skills with first-order logic (AND, OR, NOT). Averaging value functions is a simple approach to approximate the conjunction of skills (Haarnoja et al., 2018a). More generally, one can define logical operations within Boolean algebras as operations on sets (intersection, union and complement). In their paper, Tasse and colleagues formalize a homomorphism between Boolean algebras in the space of goals and the space of optimal value functions. As a result, the optimal policy for a logical combination of atomic goals is the policy that maximizes the corresponding optimal value function obtained directly from the base value functions (Tasse et al., 2020). Unfortunately, this approach makes strong assumptions restricting its use in language-conditioned settings. Our DECSTR approach from Chapter 6 can also compose skills by directly computing intersections, unions and complements of the sets of concrete goals generated from each linguistic description. However, this is only possible when the set of goals corresponding to a description can be exhaustively sampled.

Sequential combinations of skills are also necessary. A recent approach proposes a neuro-symbolic agent combining linear temporal logic with DRL (León et al., 2020). Goals are formalized in a linear temporal logic language and decomposed into a sequence of goals by a symbolic module. Each of the atomic goals is then fed to the DRL agent one by one. This method allows systematic generalization but relies on crafted goal representations. An interesting twist could be to train a parser to turn any abstract linguistic goal description into a temporal logic formulation or a decomposed sequence of goals directly targetable by a hierarchical agent. This setup would resemble recent neuro-symbolic approaches involving differentiable parsers in the context of visual question answering tasks (Mao et al., 2019).

Hierarchical reinforcement learning (HRL) can also be used for sequential skill composition (Dayan & Hinton, 1993; Parr & Russell, 1998; Sutton et al., 1998, 1999; Precup, 2000; Barto & Mahadevan, 2003). In HRL, high-level policies generate goals for low-level policies to decompose behaviors at various levels of temporal abstractions. Suppose the high-level goal is "*cooking a vegetable soup*". The high-level policy generates a plan made of a few high-level steps: "*boiling water in a sauce pan*", "*peeling the carrots and potatoes*", etc. A lower-level policy receives the intermediate goal "*boiling water in a sauce pan*" and generates a finer plan: "*filling the the pan with water*", "*putting the pan on the stove*". Finally, a policy at the lowest level generates goal-directed actions. This hierarchical task decomposition has several advantages: the higher-level policies only need to generate short sequences of sub-goals and can explore with larger steps (temporally-structured sub-goal-directed exploration); their action space is smaller and can be discretized; low-level skills can be reused and composed to form many high-level skills, etc. Low-level policies can be implemented by traditional goal-conditioned RL algorithms (Levy et al., 2019; Röder et al., 2020) and are either trained independently from the high-level policy (Kulkarni et al., 2016; Frans et al., 2018) or jointly (Levy et al., 2019; Nachum et al., 2018; Röder et al., 2020). From a set of low-level policies, one can also use planning methods

to combine skills in a hierarchical and recursive manner (Pierrot et al., 2020). Future RL-IMGEP should leverage concurrent and sequential skill compositions to conduct an efficient acquisition of skills.

Summary

In the introduction, we wrote: "*if we want to target the problem of the autonomous acquisition of open-ended repertoires of skills, we need to design embodied, situated, intrinsically motivated and autotelic agents that generate creative goals, generalize and compose skills.*" Our RL-IMGEP agents are embodied and situated, intrinsically motivated and autotelic. As demonstrated in Chapter 2, they explore well by pursuing self-generated goals (autotelic exploration). IMAGINE and DECSTR show partial systematic generalization in the linguistic space. DECSTR can compose basic skills by composing the associated sets of goals it generates with the language goal generator. IMAGINE, finally, can invent, pursue and master creative goals. In this chapter, we discussed ways to scale these approaches. We argue that the most important aspects are 1) the generation of creative (i.e. novel *and* relevant) goals in natural language; 2) the ability to target a rich diversity of goals at once and 3) the ability to compose skills both concurrently and sequentially to unleash a combinatorial explosion of the skill space. The next and last chapter of this manuscript returns to the *Vygostkian autotelic agents* proposition and discusses how richer social interactions could be turned into useful psychological tools to facilitate the acquisition of skills.

Chapter 9

Vygotskian Autotelic Agents

IMAGINE and DECSTR are first steps in the direction of Vygotskian autotelic agents (Chapter 4). In both cases, language is first a social, communicative tool. Social partners describe relevant aspects of the agents' trajectories and agents ground language in policies, reward functions or goal generators. As they do so, they *internalize* social language and turn it into a psychological tool. IMAGINE uses it to structure the imagination of creative goals that are both appropriate (built from templates of existing goals) and novel (variations of known goals). DECSTR uses it to structure the simulation of possible futures and generate behavioral diversity. In the following sections, we discuss how to enrich the social (interpsychological) and inner mental (intrapsychological) lives of future Vygotskian autotelic agents.

9.1 Social Life

Since the emergence of embodied cognition theories and embodied robotics in the 1980s, *embodiment* and *situatedness* have been widely considered necessary properties of intelligent agents (Brooks, 1990, 1991a,b; Rosch et al., 1991; Barsalou, 2008). However, computational models mainly focused on *physical situatedness*: interactions between the agent and its physical environment. Of equal importance is the need for *social and cultural situatedness*. Humans develop in rich socio-cultural environments that might be a cornerstone of their higher mental functions (Dautenhahn, 1995; Dautenhahn et al., 2002; Clark, 1998b; Brooks et al., 1999; Zlatev, 2001; Lindblom & Ziemke, 2003).

Towards Teachable Autotelic Agents

Embodied and physically situated autotelic agents form goal representations and reward functions shaped by their environment and body constraints. This can be observed in the *unsupervised skill discovery* methods reviewed in Chapter 7: robotic agents learn to move in various directions and use various configurations of their joints (Eysenbach et al., 2019; Sharma et al., 2020; Campos et al., 2020). However, without explicit inductive biases, nothing ensures that the emerging goal representations will make much sense from our point of view (e.g. an agent learning to use different joint configurations). Autotelic agents will only develop relevant skills if embedded in our socio-cultural environment

and face our social problems. As we interact with them, we will influence their goal representations, just like we do for children. In the same way, artificial agents could tap into our natural tendency to help and teach, to adapt our behaviors to the level of the learner (see the *zone of proximal development* of Vygotsky or the *scaffolding theory* of Bruner, Vygotsky, 1934; Wood et al., 1976; Bruner, 1985).

Human children are autotelic agents; they set their own goals and can learn by themselves. Nonetheless, they are *social beings*; socially situated agents whose learning and behavior are strongly shaped by social interactions with their peers (demonstrations, descriptions, encouragements, attention guidance, curricula, etc.). As we argued in a recent opinion paper called *Towards Teachable Autonomous Agents* (Sigaud et al., 2021), we need to design agents that are both *autotelic* and *teachable*. While previous research focused on either *autotelic agents* (e.g. intrinsically motivated RL, IMGEPs) or *teachable agents* (e.g. inverse RL, interactive RL), we advocate for a convergence of these lines of research.

Richer Pragmatic Frames

Teachable autotelic agents must learn from various sources of social interactions, including descriptions, demonstrations, linguistic explanations, goal suggestions and goal decompositions. The framework of *pragmatic frames* encompasses all these forms of interactions:

> "*Pragmatic frames are verbal or non-verbal patterns of goal-oriented behaviors that evolve over repeated interactions between a learner and a teacher.*" See Section 4.2.

RL algorithms have been proposed to handle several forms of pragmatic frames. Section 4.2 covered several of them, including *instruction and feedback*, *learning from demonstrations*, *learning from advice*, *learning from descriptions*.

Learning from preferences is another one. In this framework, the human teacher provides comparative judgments by ranking two trajectories of the learner. From this feedback, the learner can infer a reward function to practice its skills. This method has been used in scenarios where the design of a good reward function is difficult (e.g. learning to do a backflip Christiano et al. (2017) or learning to summarize a text Stiennon et al. (2020)), or when the original reward is too sparse (e.g. in Montezuma, Ibarz et al., 2018). It could be extended to multi-goal settings and active learning scenarios where agents generate several goal-directed behaviors and ask the teacher to provide feedback on their preference.

Recently, the *interactive agents group* at DeepMind conducted a large-scale experiment to train agents in contact with humans (Abramson et al., 2020). Although they discuss the integration of four types of pragmatic frames (question-answer, instruction-feedback, play and dialogue), they currently integrate two of them: question-answer and instruction-feedback. In these frames, the learner can fulfill both positions: 1) instructing another agent with either a physical goal (doing something in the scene) or a learning goal (asking a question about the scene) or 2) executing the instruction (acting or answering).

This paper mainly focuses on preliminary aspects of human-robot interactions: how to bootstrap agents to achieve meaningful interactions and how to evaluate the quality of the interaction. Indeed, because artificial agents act randomly at first, humans get bored and find it challenging to provide helpful feedback—how can I compare two random trajectories? The authors propose to bootstrap learning agents with a combination of auxiliary losses and imitation learning from a large set of human-human interactions. To evaluate the quality of interactions, they propose to use a set of probe tasks and ask humans to provide absolute and relative evaluations of human-agent and agent-agent interactions. From this data, they learn a scoring function that correlates well with held-out human decisions. This method is interesting because it can help provide auxiliary losses to encode social priors in agents.

Despite these successes, current implementations of interactive RL fail to reproduce all the features of natural pragmatic frames (Vollmer et al., 2016; Rohlfing et al., 2016). One can mention two major differences. First, existing approaches always consider a very restricted set of pragmatic frames, often a single one, whereas natural teacher-learner interactions encompass a rich diversity of pragmatic frames. Second, existing approaches consider *fixed* pragmatic frames: teachers and learners have fixed roles and know how to fulfill them. In the instruction-following frame, for example, interaction is segmented in episodes. Teachers always start by instructing and always wait for learners to execute before providing binary feedback. On the other hand, natural pragmatic frames are learned through social interactions and change with time. For a deeper discussion on designing richer social lives for teachable autotelic agents, interested readers can refer to our recent opinion paper (Sigaud et al., 2021).

9.2 Mental Life

Following the Vygotskian approach, one must now discuss how agents integrate these social processes into inner processes and use them to structure their cognition. Autotelic agents can internalize instructions and goal decompositions provided by social partners by learning to generate them for themselves. One way is to store instructions in a list that the agent can use to sample goals for itself. From this list, the agent can train a generative model of goals, possibly conditioned on the initial state, so that certain goals are only targeted in certain situations (e.g. Nair et al., 2020). In a recent study, Chen and colleagues trained a generative model of linguistic plans from human-generated decompositions of goals in a video game (Chen et al., 2021). We can draw a parallel between the linguistic plans generated by hierarchical agents and children's private speech. In an initial phase, the social partner generates plans to help the learner structure its experience. In the second phase, the learner has internalized these plans and can now generate plans for itself—the social speech has turned into private speech.

Agents can also internalize descriptive feedback. DECSTR, for example, learns to map descriptions to internal goal representations and, after this internalization process, generates its own goal configurations from linguistic descriptions. As it does so, it grounds the meaning of abstract descriptions (e.g. *a pyramid*) into its innate semantic representations. IMAGINE, on the other hand, stores descriptions into a list of targetable linguistic goals. From this list, it builds a generative model of goals by detecting templates

and substituting equivalent words to generate new goals. Besides, IMAGINE internalizes descriptive feedback by learning an internal reward function. IMAGINE uses it to scan lists of goals and find the descriptions of any given state. This inefficient captioning system is, in fact, the internalization of the social descriptions; IMAGINE now generates descriptions for itself. Recent works have explicitly stated the objective of learning a trajectory captioner (Cideron et al., 2020b; Zhou & Small, 2020; Nguyen et al., 2021; Chen et al., 2021). Agents can use captioners to relabel their trajectories for hindsight learning or to label the trajectories of others. Describing others' trajectories as if they were ours powers a form of *theory of mind* (Wellman, 1992; Rabinowitz et al., 2018; Jara-Ettinger, 2019). Agents infer the goals of others and, doing so, can use their trajectories as demonstration *in the wild*. Justesen and colleagues took a similar approach by inferring behavioral characterizations from demonstrations and using them to train a behavior-conditioned policy via imitation learning (Justesen et al., 2020b).

Another way to use descriptive feedback is to train language-conditioned world models. According to the *embodied simulation hypothesis*, humans generate mental simulations (e.g. visual, auditory, sensorimotor) when they hear or read language (Bergen, 2012). As agents hear linguistic descriptions of their experience, they can learn to associate both, such that hearing the word is enough to trigger a simulation of related experiences. These models can be helpful to encode compact memories and communicative intents as simple words and descriptions. When world models are synchronized, a word can trigger the simulation of past experiences in both ourselves and others—a linguistic version of Proust's madeleine. As autotelic agents generate inner speech, they can trigger the generation of memories, sensory hallucinations, imaginations of future possibilities that will, in turn, impact their future actions and inner speech, thereby generating an inner mental life.

DECSTR generates mental simulations of possible futures conditioned on linguistic descriptions. From these options, it selects one, targets it and makes it come true. The simple alignment of sensory experience and linguistic descriptions can indeed go a long way, as a recent model called DALL-E demonstrated (Ramesh et al., 2021). DALL-E is trained on a similar dataset aligning images and descriptions (although much larger). It generates photo-realistic images from linguistic descriptions alone and, more surprisingly, generalizes to new linguistic compositions such as "*a chair in the shape of an avocado*." This shows that the structure of language—i.e. its compositionality—can successfully be projected onto the continuous image space: using compositional descriptions helps compositional thinking. This is in line with the findings of Vyshedskiy: just like DALL-E, children also need early exposure to compositional language to develop the ability to compose mental images (Vyshedskiy, 2019).

Social interactions under the form of questions and answers could also be integrated to structure autotelic agents' behavior. In the framework of *embodied question answering* (Das et al., 2018; Yuan et al., 2019), agents are asked questions and explore their environment to answer them. Agents could internalize this social interaction and learn to ask questions to themselves to self-organize exploration. The generation of questions could be optimized with intrinsic rewards such as the maximization collected knowledge or visited states' novelty. TextWorld environments, because they abstract away the sensorimotor interactions to focus on high-level linguistic states and actions, offer an interesting framework to tackle these questions (Côté et al., 2018; Yuan et al., 2019).

RL-IMGEP offers a natural framework to implement these aspects. Autotelic agents first interact in socially rich environments, learn from demonstrations, plans, descriptions, advice, external curriculum, etc. As they do, they learn to integrate these processes by training language-conditioned world models, captioners, advice generators, plan generators and self-paced curricula. These internalized processes then structure their behaviors. As autotelic agents call for these internal world models, they generate an inner speech that triggers memories, imaginary sensations and future projections, affecting behavior. This internal loop forms an inner mental life.

Conclusion

Congratulations! You have now reached the end of this manuscript.

What was it about already? We have been interested in the design of artificial agents that build repertoires of skills autonomously. We have emphasized two central notions: goals and language. Our artificial agents are *autotelic*; they learn to represent, generate, pursue and master their own goals. By targeting their own goals, autotelic agents organize their exploration of the world and sculpt their learning trajectories of skills acquisition. Our agents are also *Vygotskian*; they internalize pre-existing social languages and turn them into cognitive tools to learn abstract and generalizable goal representations, organize their exploration and simulate possible futures.

How did you do that? We have taken inspiration from existing concepts and computational frameworks from psychology, developmental robotics and reinforcement learning. Building on the goal-conditioned deep reinforcement learning framework, we have endowed artificial agents with the ability to represent their own goals, select which ones to pursue and generate their own internal rewards. Our new family of algorithms called *rl-based intrinsically motivated goal exploration processes* (RL-IMGEP) is an extension of the existing IMGEP framework from developmental robotics. With deep reinforcement learning methods, our IMGEP agents handle higher dimensional inputs, require less expert knowledge and are more robust to perturbations.

So, is that it? We titled this manuscript *Towards Vygotskian Autotelic Agents* and, indeed, our proposed algorithms are only the first steps. In the future, artificial agents should represent and master a richer diversity of goals. This requires agents to represent goals in a compact and abstract way (e.g. with language), imagine new creative goals and learn to compose existing skills in an open-ended way. Future artificial agents should also learn in richer social environments and benefit from the teaching of humans in the real world. This will require agents to co-learn pragmatic frames, recognize them in the wild and learn from them. As argued by Lake and colleagues, this will require a "developmental start-up kit" (Lake et al., 2017). Agents might have to learn intuitive theories of physics and people, similar to the ones developed by children early in life. Building on this kit, artificial agents should develop a variety of world models. With causal models of physics and people, artificial agents can imagine possible futures and plan towards them. They can also represent *counterfactuals*:

> "*Cognition is about using these models to understand the world, to explain what we see, to imagine what could have happened that didn't, or what could be true that isn't, and then planning actions to make it so.*" (Lake et al., 2017)

Making up imaginary goals and planning towards them might be a good way to fine-tune these models and practice goal-reaching under-constraints (Chu & Schulz, 2020b). In all these tasks, language might be a tool of choice. People use it all the time to describe their theories, develop new ideas, plan for the future, examine counterfactuals or communicate all these to others. It is because of its properties of abstraction, compositionality and productivity.

One last crazy half-baked idea before the end? Endowing agents with a combination of linguistic world models and inner speech might be an exciting idea. As they generate inner speech, agents trigger their linguistic world models. This can include the simulation of perceptual experience (images, sounds), sensorimotor trajectories, descriptions of possible futures or past experiences. Observing these outputs, agents would produce new behaviors and inner speech. This inner loop, akin to our mental life, could help agents in their goal-directed exploration. Language could trigger memories or mnemotechnic visual representations acting as cognitive aids for the agent.

Appendices

Appendix A

Methodology

A.1 Baselines, Oracles, Ablations and Variants

Let us assume we just designed a new algorithm A to solve a particular problem P. We now need to demonstrate the "quality" of this new algorithm; why should anyone be interested? Let us assume the problem is interesting. We now need to show that: 1) A performs well on the problem, maybe even solves it; 2) all components of A contribute to its quality, e.g. increases performance or relaxes assumptions and 3) A compares favorably to existing algorithms. Now, this last point is a bit imprecise. The tendency in machine learning is to say that an algorithm is interesting if it performs better on the problem. This view is somehow limited; an algorithm can be said interesting if it is simpler (e.g. fewer components or fewer priors) or demonstrates similar performance and complexity but does so in a different way (e.g. in a more biologically plausible way).

Performance on the task might also be a multi-dimensional criterion. Standard RL algorithms can be compared in terms of final performance, sample efficiency or variability across random seeds. As we discuss in Section 1.5, we can evaluate autotelic agents' performance on downstream tasks, the diversity of discovered skills, the mastery of discovered skills, generalization, etc. As a result, discussing the importance and interest of a new algorithm can sometimes be more complex than it first appears. In this discussion, researchers often use comparisons with other algorithms that can be classified into four categories—*baselines, oracles, ablations* and *variants*:

- **Baselines** are existing related algorithms that can be applied to problem P with minimal modifications. When the considered baseline B was introduced to solve P, it is an interesting and essential comparison to make. When B was proposed to solve another problem P', its adaptation to solve P might introduce biases, which becomes more probable the more P' differs from P. This can lead to the problem of *under-tuning* when researchers spend less effort on tuning hyperparameters of A than they spent tuning those of B.
- **Oracles** O are algorithms that benefit from extra-knowledge compared to A. It can be a complete hard-coded solution to P or a variant of A that replaces some components with improved ones benefiting from external knowledge. It is used as a higher bound of A's performance. The goal for A is to perform as closely as possible to O.

- **Variants** V are alternative versions of A that replace some components with others to act more *like* pre-existing algorithms. It is a way to construct baselines with minimal contrasts to A. When a baseline B is originally introduced to solve P', its adaptation to P involves making B *more like* A. Variants take the opposite stance, starting from A and making it *more like* B by introducing the very components that B claimed to be important. This strategy has two advantages. First, it reduces under-tuning, because V shares most of its components and hyper-parameters with A, thus benefits from its tuning. Second, V and A are minimally contrasted such that the comparison focuses on the specific contrasted components. Indeed, DRL algorithms often involve many components, and comparing A to B would tell us "which is best" but would tell very little about the impact of each component. Because the problem P that we consider—the autonomous acquisition of open-ended repertoires—is often significantly different from the problems P' targeted in related works, we follow the strategy of replacing modified B baselines with minimally-contrastive variants V.

- **Ablations** A' are modified versions of A that remove or replace components of A. This is used to provide minimal contrasts and investigate the impact of each component of A. To prove that all components of A contribute to its quality, we must ensure that A performs better than its ablations. If a component of A helps relax one assumption about the problem, the ablation of A that removes that component is important to discuss the impact of this relaxing (e.g. does it impact performance?).

In this research, we often look at various measures to qualify the agents' behaviors and learning patterns. We use these four types of comparisons to discuss in depth the impact of each component of our algorithms and how they compare to related algorithms.

A.2 Statistical Comparisons

Reproducibility in machine learning, and reinforcement learning (RL) in particular, has become a severe issue in recent years. As pointed out in two recent papers, reproducing the results of an RL paper can turn out to be much more complicated than expected (Islam et al., 2017; Henderson et al., 2018). In their thorough investigation, Henderson and colleagues showed that these difficulties can be caused by differences in codebases, hyperparameters (e.g. size of the network, activation functions) or the number of random seeds used in the original study (Henderson et al., 2018). They state the obvious: the claim that an algorithm performs better than another should be supported by evidence and tested with statistical tests. However, the RL literature rarely uses statistical testing.

Building on these observations, we conducted a comparative study of statistical tests in terms of false-positive rates and statistical powers under various assumptions on the performance distributions, the variance, etc. This study led us to write the *Hitchhiker's Guide to Statistical Comparisons of RL Algorithms* (Colas et al., 2019b). For the sake of concision, we only report the conclusions and guidelines here. Interested readers can refer to the paper (Colas et al., 2019b).

Definitions

Statistical difference testing. Statistical difference testing offers a principled way to compare the central measures of performance μ_1, μ_2 of two algorithms.[1] We first define two hypotheses:

1. the *null hypothesis* $\mathcal{H}_0$: $\Delta\mu = \mu_1 - \mu_2 = 0$,
2. the *alternative hypothesis* $\mathcal{H}_a$: $|\Delta\mu| > 0$.

When performing a test, one initially assumes the null hypothesis. Then, we measure the performance of the two algorithms and obtain two samples x_1 and x_2. The next step is to estimate the probability to observe two samples whose empirical central difference is at least as extreme as the observed one ($|\Delta\overline{x}| = |\overline{x_1} - \overline{x_2}|$) under the null hypothesis $\mathcal{H}_0$ (i.e. given $\Delta\mu = 0$). This probability is called the *p-value*. If the p-value is very low, the test rejects $\mathcal{H}_0$ and concludes that a true underlying difference ($\mathcal{H}_a$) is likely. When the p-value is high, the test does not have enough evidence to conclude. This could be due to the lack of true difference, or the lack of statistical power (too few measurements given how noisy they are). The significance level α (usually ≤ 0.05) draws the line between rejection and conservation of $\mathcal{H}_0$: if p-value $< \alpha$, then $\mathcal{H}_0$ is rejected.

	True $\mathcal{H}_0$	True $\mathcal{H}_a$
Predicted $\mathcal{H}_0$	True negative $1 - \alpha^*$	False negative β^*
Predicted $\mathcal{H}_a$	False positive α^*	True positive $1 - \beta^*$

Table A.1: Hypothesis testing

Statistical errors. Note that having a p-value of 0.05 still results in 1 chance out of 20 to claim a difference that does not exist. This is called a *type-I error* or a *false positive*. The false-positive rate is usually noted α, just like the significance level. Indeed, statistical tests with significance level α are supposed to enforce a false-positive rate inferior to α. Because our experiments demonstrate it is not always the case, we prefer to note the false-positive rate α^*. False negatives occur when the statistical test fails to recognize a true difference in the measures of the central performance. The false-negative rate is noted β^*.

Trade-off between false positive and statistical power. Ideally, we would like to set $\alpha = 0$ to ensure the lowest possible false-positive rate α^*. However, decreasing the confidence level makes the statistical test more conservative. The test requires even larger empirical differences $\Delta\overline{x}$ to reject $\mathcal{H}_0$, which decreases the probability of true positives $1 - \beta^*$. This probability of true positives, also called the test's *statistical power*, is the

[1] *Central measure* refers to either the mean or the median.

probability to reject $\mathcal{H}_0$ when $\mathcal{H}_a$ holds. Several factors impact it. Larger effect sizes, lower variances and larger sample sizes all lead to stronger statistical power because they make the underlying difference easier to detect, thus bring more evidence for the rejection of $\mathcal{H}_0$. The significance level conditions the amount of evidence required to pass the test, thus negatively impacts the statistical power. Generally, the sample size is chosen as a function of the expected effect size to obtain a theoretical statistical power of $1-\beta^* = 0.8$. Different tests have different statistical powers depending on the assumptions they make, whether they are met, how the p-value is derived, etc.

Relative effect size. The *relative effect size* ϵ is the absolute effect size $|\Delta\mu|$, scaled by the pooled standard deviation σ_{pool}, such that $\epsilon = |\Delta\mu|/\sigma_{pool}$. The relative effect size is independent of the spread of the considered distributions.

Comparison Methodology

Measuring the performance of RL algorithms. Before using any statistical test, one must obtain measures of performance. RL algorithms should ideally be evaluated *offline*. The algorithm performance after t steps is measured as the average of the returns over E evaluation episodes conducted independently from training, usually using a deterministic version of the current policy (e.g. $E = 20$). Evaluating agents by averaging performance over several training episodes results in a much less interpretable performance measure and should be stated clearly. The collection of performance measures across time forms a learning curve.

Representing learning curves. Learning curves for several runs can be rendered on a plot. One can represent the median or the mean. Whereas the empirical median directly represents the center of the collected sample, the empirical mean tries to model the sample as coming from a Gaussian distribution and, under this assumption, represents the maximum likelihood estimate of that center. One must also represent error bars. The standard deviation (SD) represents the variability of the performance measures but is only representative when the values are approximately normally distributed. When performance measures are normally distributed, one can use the standard error of the mean (SE) or confidence intervals to represent estimates of the uncertainty on the mean estimate.

Robust comparisons. Which test, which sample sizes? Our results advocate for the use of Welch's t-test, which shows lower false-positive rates and similar statistical powers than other tests. The number of random seeds to achieve a statistical power of 0.8 depends on the expected relative effect size:

$$\begin{aligned}\epsilon &= 0.5 \rightarrow N \approx 100,\\ \epsilon &= 1 \rightarrow N \approx 20,\\ \epsilon &= 2 \rightarrow N \approx 5, 10.\end{aligned}$$

The analysis of a study case comparing SAC and TD3 algorithms required a sample size between $N = 5$ and $N = 10$ for a relatively strong effect $\epsilon = 0.93$ when comparing the means, and about 5 more seeds when comparing the medians ($\epsilon = 0.80$).

A word on multiple comparisons. When performing multiple comparisons (e.g. between different pairs of algorithms evaluated in the same setting), the probability of having at least one false positive increases linearly with the number of comparisons n_c. This probability is called the *family-wise error rate* (FWER). To correct for this effect, one must apply corrections. The Bonferroni correction for instance adapts the confidence level $\alpha_{Bonf.} = \alpha/n_c$ (Bonferroni, 1936). This ensures that the FWER stays below the initial α but drastically impacts the statistical power of the test.

Comparing full learning curves. Instead of only comparing the final performance measures of the two algorithms after T timesteps in the environment, we can compare performance across learning by performing tests at each evaluation step. This might reveal differences in the speed of convergence and can provide more robust comparisons. Figure A.1 demonstrates this with runs for SAC and TD3, using a t-test ($N = 5$). The correction to apply when comparing two learning curves depends 1) on the number of comparisons, 2) on the criterion that is used to conclude whether an algorithm is better than the other. The criterion used to draw a conclusion should be decided before running any test. Picking the decision criterion after data collection can lead to cherry-picking: e.g. selecting the only criterion leading to a positive outcome among a set of C criteria. An example of criterion could be: "*if when comparing the last* 100 *performance measures of the two algorithms, more than* 5 *comparisons show a significant difference in favor of Algorithm 1, then it is better than Algorithm 2*". In that case, the number of comparisons performed is $N_c = 100$, and the criterion is $N_{rejection} > N_{crit} = 5$. We want to constrain the probability FWER that our criterion is met by pure chance to a confidence level $\alpha = 0.05$. This probability is given by the repartition function of the binomial law with parameters $n = 100$ and $p = \alpha$:

$$\text{FWER}(\alpha,\, N_c) = P(x > N_c) = 1 - \sum_{k=0}^{N_c} \binom{100}{k} \alpha^k \,(1-\alpha)^{100-k}.$$

This probability stays below 0.05 for $N_{crit} > 10$. For $N_{crit} = 5$, the probability is 0.56: the criterion is too easy to pass. One can use a stronger criterion $N_{crit} > 10$, or correct the value of α such that $\text{FWER}(\alpha_{\text{corrected}},\, N_{crit} = 5) < 0.05$, here $\alpha_{\text{corrected}} = 0.019$ works.

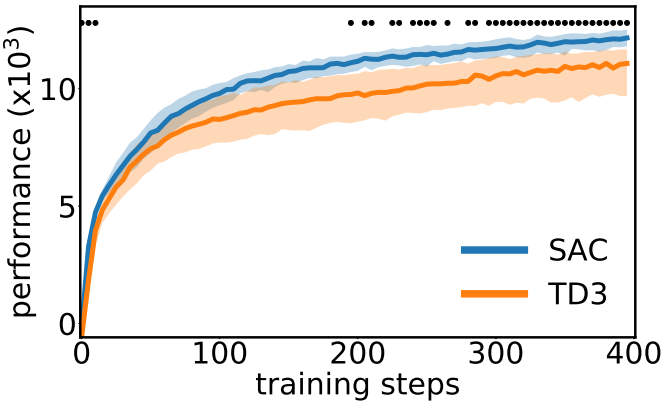


Figure A.1: Comparing two real RL algorithms: SAC and TD3. Example of learning curves representations. Here $N = 5$ for both samples, the median and the 80^{th} percentiles are represented. Black dots indicate statistically significant differences using a t-test and $\alpha = 0.01$.

Appendix B

Appendix of IMAGINE

This annex presents the additional methods, results and discussion, as well as implementation details for the IMAGINE study in Chapter 5.

- Section B.1 gives a complete description of the *Playground* environment.
- Section B.2 presents a focus on generalization and studies different types of generalization.
- Section B.3 presents a focus on exploration and how it is influenced by goal imagination.
- Section B.4 presents a focus on the goal imagination mechanism we use in IMAGINE.
- Section B.5 presents a focus on the *modular-attention* architecture.
- Section B.6 presents a focus on the benefits of learning the reward function.
- Section B.7 provides additional visualization of the goal embeddings and the attention vectors.
- Section B.8 discusses the comparison with goal-as-state approaches.
- Section B.9 gives all necessary implementation details.

B.1 Complete Description of the Playground Environment

Environment Description

The environment is a 2D square: $[-1.2, 1.2]^2$. The agent is a disc of diameter 0.05 with an initial position (0, 0). Objects have sizes uniformly sampled from [0.2, 0.3]. Their initial positions are randomized so that they are not in contact with each other. The agent has an three-dimensional action space $[-1, 1]^3$. The first two actions control the agent's continuous 2D translation (bounded to 0.15 in any direction). The third action controls the gripper (closed is positive). The agent can grasp objects by touching them and closing its gripper, unless it already has an object in hand. Objects include 10 animals, 10 plants, 10 pieces of furniture and 2 supplies. Admissible categories are *animal, plant, furniture,*

supply and *living_thing* (animal or plant), see Figure B.1. Objects are assigned a color attribute (red, blue or green). Their precise color is a continuous RGB code uniformly sampled from RGB subspaces associated with their attribute color. Each scene contains three of these procedurally-generated objects (see paragraph about the Social Partner below).

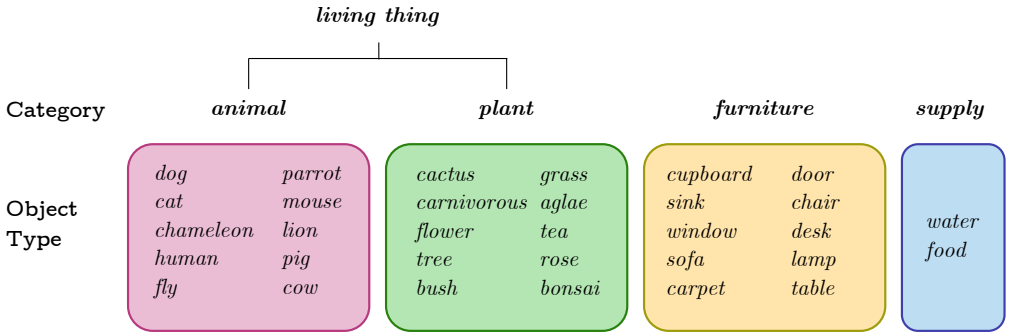


Figure B.1: Objects types and categories in Playground.

Agent Perception

At time step t, we can define an observation $\mathbf{o}_t$ as the concatenation of body observations (2D-position, gripper state) and objects' features. These two types of features form affordances between the agent and the objects around. These affordances are necessary to understand the meaning of object interactions like *grasp*. The state $\mathbf{s}_t$ used as input of the models is the concatenation of $\mathbf{o}_t$ and $\mathbf{\Delta o}_t = \mathbf{o}_t - \mathbf{o}_0$ to provide a sense of time. This is required to acquire the understanding and behavior related to the *grow* predicate, as the agent needs to observe and produce a change in the object's size.

Social Partner

SP has two roles:

- *Scene organization*: SP organize the scene according to the goal selected by the agent. When the agent selects a goal, it communicates it to SP. If the goal starts with the word *grow*, SP adds a procedurally-generated supply (water or food for animals, water for plants) of any size and color to the scene. If the goal contains an object (e.g. *red cat*), SP adds a corresponding object to the scene (with a procedurally generated size and RGB color). The remaining objects are generated procedurally. As a result, the objects required to fulfill a goal are always present and the scene contains between 1 (*grow* goals) and 3 (*go* goals) random objects. Note that all objects are procedurally generated (random initial position, RGB color, and size).

- *Scene description*: SP provides linguistic descriptions of interesting outcomes experienced by the agent at the end of episodes. It takes the final state of an episode ($\mathbf{s}_T$) as input and returns matching linguistic descriptions: $\mathcal{D}_{\text{SP}}(\mathbf{s}_T) \subset \mathcal{D}^{\text{SP}}$. When SP

provides *descriptions*, the agent considers them as targetable *goals*. This mapping $\mathcal{D}^{\text{SP}} \rightarrow \mathcal{G}^{\text{train}}$ simply consists in removing the first *'you'* token (e.g. turning *'you grasp red door'* into the goal *'grasp red door'*). Given the set of previously discovered goals ($\mathcal{G}_{\text{known}}$) and new descriptions $\mathcal{D}_{\text{SP}}(\mathbf{s}_T)$, the agent infers the set of goals that were not achieved: $\mathcal{G}_{\text{na}}(\mathbf{s}_T) = \mathcal{G}_{\text{known}} \setminus \mathcal{D}_{\text{SP}}(\mathbf{s}_T)$, where $\setminus$ indicates the complement.

Grammar

We now present the grammar generating descriptions for the set of goals achievable in Playground ($\mathcal{G}^A$). ***Bold*** and { } refer to sets of words while *italic* refers to particular words:

1. Go: *(e.g. go bottom left)*
   - *go* + ***zone***
2. Grasp: *(e.g. grasp any animal)*
   - *grasp* + ***color*** ∪ {*any*} + ***object type*** ∪ ***object category***
   - *grasp* + *any* + ***color*** + *thing*
3. Grow: *(e.g. grow blue lion)*
   - *grow* + ***color*** ∪ {*any*} + ***living thing*** ∪ {*living_thing, animal, plant*}
   - *grow* + *any* + ***color*** + *thing*

Word sets are defined by:

- ***zone*** = {*center, top, bottom, right, left, top left, top right, bottom left, bottom right*}
- ***color*** = {*red, blue, green*}
- ***object type*** = ***living thing*** ∪ ***furniture*** ∪ ***supply***
- ***object category*** = {*living_thing, animal, plant, furniture, supply*}
- ***living thing*** = ***animal*** ∪ ***plant***
- ***animal*** = {*dog, cat, chameleon, human, fly, parrot, mouse, lion, pig, cow*}
- ***plant*** = {*cactus, carnivorous, flower, tree, bush, grass, algae, tea, rose, bonsai*}
- ***furniture*** = {*door, chair, desk, lamp, table, cupboard, sink, window, sofa, carpet*}
- ***supply*** = {*water, food*}
- ***predicate*** = {*go, grasp, grow*}

We partition this set of achievable goals into a training ($\mathcal{G}^{\text{train}}$) and a testing ($\mathcal{G}^{\text{test}}$) set. Goals from $\mathcal{G}^{\text{test}}$ are used to evaluate the ability of agents to explore outcomes beyond the ones described by SP. The next section introduces this testing set and focuses on generalization. Note that some goals might be syntactically valid but not achievable. This includes all goals of the form *grow* + ***color*** $\cup$ {*any*} + ***furniture*** $\cup$ {*furniture*} (e.g. *grow red lamp*).

B.2 Focus on Generalization

Table B.1 provides the exhaustive list of goals used to test each type of generalization.

Table B.1: Testing goals in $\mathcal{G}^{\text{test}}$, by type.

Type 1	*Grasp blue door*, *grasp green dog*, *grasp red tree*, *grow green dog*
Type 2	*Grasp any flower*, *grasp blue flower*, *grasp green flower*, *grasp red flower*, *grow any flower*, *grow blue flower*, *grow green flower*, *grow red flower*
Type 3	*Grasp any animal*, *grasp blue animal*, *grasp green animal*, *grasp red animal*
Type 4	*Grasp any fly*, *grasp blue fly*, *grasp green fly*, *grasp red fly*
Type 5	*Grow any algae*, *grow any bonsai*, *grow any bush*, *grow any cactus* *grow any carnivorous*, *grow any grass*, *grow any living_thing*, *grow any plant* *grow any rose*, *grow any tea*, *grow any tree*, *grow blue algae* *grow blue bonsai*, *grow blue bush*, *grow blue cactus*, *grow blue carnivorous* *grow blue grass*, *grow blue living_thing*, *grow blue plant*, *grow blue rose* *grow blue tea*, *grow blue tree*,*grow green algae*, *grow green bonsai* *grow green bush*, *grow green cactus*, *grow green carnivorous*, *grow green grass* *grow green living_thing*, *grow green plant*, *grow green rose*, *grow green tea* *grow green tree*, *grow red algae*, *grow red bonsai*, *grow red bush* *grow red cactus*, *grow red carnivorous*, *grow red grass*, *grow red living_thing* *grow red plant*, *grow red rose*, *grow red tea*, *grow red tree*

B.3 Focus on Exploration

Interesting Interactions

Interesting interactions are trajectories of the agent that humans could infer as goal-directed. If an agent brings water to a plant and grows it, it makes sense for a human. If it then tries to do this for a lamp, it also feels goal-directed, even though it does not work. This type of behavior characterizes the penchant of agents to interact with objects around them, to try new things.

Sets of Interesting Interactions

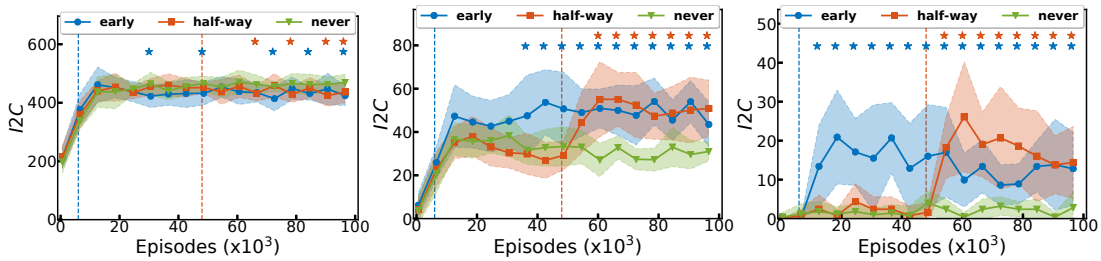
We consider three sets of interactions: 1) interactions related to training goals; 2) interactions related to testing goals; 3) the extra set. This *extra set* contains interactions where the agent brings water or food to a piece of furniture or to another supply. Although such behaviors do not achieve any of the goals, we consider them as interesting exploratory behaviors. Indeed, they testify that agents try to achieve meaningful imagined goals that are meaningful from their point of view, i.e. corresponding to a meaningful form of generalization after discovering that animals or plants can be grown (e.g. *grow any door*).

The Interesting Interaction Count Metric

We count the number of interesting interactions over all final transitions from the last 600 episodes (1 epoch). Agents do not need to target these interactions; we simply report the number of times they are experienced. Indeed, the agent does not have to target a particular interaction for the trajectory to be interesting from an exploratory point of view. The HER mechanism ensures that these trajectories can be replayed to learn about any goal, imagined or not. Computed on the extra set, the *Interesting Interaction Count* (I2C) is the number of times the agent was found to bring supplies to a piece of furniture or other supplies over the last epoch:

$$\text{I2C}_{\text{extra}} = \sum_{i \in \mathcal{I}_{\text{extra}}} \sum_{t=1}^{600} \delta_{i,t},$$

where $\delta_{i,\,t} = 1$ if interaction i was achieved in episode t, 0 otherwise and $\mathcal{I}_{\text{extra}}$ is the set of interesting interactions (here from the extra set) performed during an epoch. Imaginative agents achieve higher scores in the testing and extra sets of interactions, while maintaining similar exploration scores on the training set, see Figures B.2a to B.2c.



(a) (b) (c)

Figure B.2: Exploration metrics (a) Interesting interaction count (I2C) on training set, (b) I2C on testing set, (c) I2C on extra set. Goal imagination starts early (vertical blue line), half-way (vertical orange line) or does not start (*no imagination* baseline in green).

B.4 Focus on Goal Imagination

Algorithm 4 presents the algorithm underlying our goal imagination mechanism. This mechanism is inspired by the *Construction Grammar* (CG) literature and generates new sentences by composing known ones (Goldberg, 2003). It computes sets of equivalent words by searching for sentences with an edit distance of 1: sentences where only one word differs. These words are then labeled equivalent and can be substituted in known sentences. Note that the goal imagination process filters goals that are already known. Imagining goals from $\mathcal{G}^{\text{train}}$ before they are discovered drives the exploration of IMAGINE agents. In our setup, however, this effect remains marginal as IMAGINE discovers all goals from $\mathcal{G}^{\text{train}}$ in the first epochs (see Figure B.4).

Algorithm 4 Goal Imagination. The edit distance between two sentences refers to the number of words to modify to transform one sentence into the other.

Input: $\mathcal{G}_{\text{known}}$ (discovered goals)
Initialize: empty list of sets of equivalent words *word_eq*, empty list of template sentences *goal_templates*, empty set of imagined goals $\mathcal{G}_{\text{im}}$.
▷ Computing word equivalences
for g_{NL} in $\mathcal{G}_{\text{known}}$ **do**
    new_goal_templates = True
    for g_m in *goal_templates* **do**
        if edit_distance(g_{NL}, g_m) < 2 **then** ▷ g_{NL} belongs to template g_m
            new_goal_templates = False
            if edit_distance(g_{NL}, g_m) == 1 **then**
                $w_1, w_2 \leftarrow$ *get_non_matching_words*(g_{NL}, g_m)
                if w_1 and w_2 not in any of *word_eq* sets **then** ▷ $(w_1\, w_2)$: new equivalent set
                    word_eq.add($\{w_1, w_2\}$)
                else
                    for *eq_set* in *word_eq* **do** ▷ update equivalence sets
                        if $w_1 \in$ *eq_set* or $w_2 \in$ *eq_set* **then**
                            eq_set = *eq_set* $\cup \{w_1, w_2\}$
    if *new_goal_templates* **then** ▷ if g_{NL} belongs to no known templates
        goal_templates.add(g_{NL})
▷ Generating new sentences
for g in *goal_templates* **do**
    for w in g **do**
        for *eq_set* in *word_eq* **do**
            if $w \in$ *eq_set* **then**
                for w' in *eq_set* **do**
                    $g_{im} \leftarrow$ replace(g, w, w')
                    if $g_{im} \notin \mathcal{G}_{\text{known}}$ **then**
                        $\mathcal{G}_{\text{im}} = \mathcal{G}_{\text{im}} \cup \{g_{im}\}$
$\mathcal{G}_{\text{im}} = \mathcal{G}_{\text{im}} \setminus \mathcal{G}_{\text{known}}$ ▷ filtering known goals.

Table B.2: All imaginable goals $\mathcal{G}^{\text{im}}$ generated by the Construction Grammar Heuristic.

Goals from $\mathcal{G}^{\text{train}}$	$\mathcal{G}^{\text{train}}$. (Note that known goals are filtered from the set of imagined goals. However, any goal from $\mathcal{G}^{\text{train}}$ can be imagined before it is encountered in the interaction with SP.)
Goals from $\mathcal{G}^{\text{test}}$	All goals from Type 1, 3, 4 and 5, see Table B.1
Syntactically incorrect goals	*Go bottom top*, *go left right*, *grasp red blue thing*, *grow blue red thing*, *go right left*, *go top bottom*, *grasp green blue thing*, *grow green red thing*, *grasp green red thing* *grasp blue green thing*, *grasp blue red thing*, *grasp red green thing*.
Syntactically correct but unachievable goals	*Go center bottom*, *go center top*, *go right center*, *go right bottom*, *go right top*, *go left center*, *go left bottom*, *go left top*, *grow green cupboard*, *grow green sink*, *grow blue lamp*, *go center right*, *grow green window*, *grow blue carpet*, *grow red supply*, *grow any sofa*, *grow red sink*, *grow any chair*, *go top center*, *grow blue table*, *grow any door*, *grow any lamp*, *grow blue sink*, *go bottom center*, *grow blue door*, *grow blue supply*, *grow green carpet*, *grow blue furniture*, *grow green supply*, *grow any window*, *grow any carpet*, *grow green furniture*, *grow green chair*, *grow green food*, *grow any cupboard*, *grow red food*, *grow any table*, *grow red lamp*, *grow red door*, *grow any food*, *grow blue window*, *grow green sofa*, *grow blue sofa*, *grow blue desk*, *grow any sink*, *grow red cupboard*, *grow green door*, *grow red furniture*, *grow blue food*, *grow red desk*, *grow red table*, *grow blue chair*, *grow red sofa*, *grow any furniture*, *grow red window*, *grow any desk*, *grow blue cupboard*, *grow red chair*, *grow green desk*, *grow green table*, *grow red carpet*, *go center left*, *grow any supply*, *grow green lamp*, *grow blue water*, *grow red water*, *grow any water*, grow green water, *grow any water*, grow green water.

Imagined Goals

We run the goal imagination mechanism based on the construction grammar heuristic (CGH) from $\mathcal{G}^{\text{train}}$. After filtering goals from $\mathcal{G}^{\text{train}}$, this produces 136 new imagined sentences. Table B.2 presents the list of these goals while Figure B.3 presents a Venn diagram of the various goal sets. Among these 136 goals, 56 belong to the testing set $\mathcal{G}^{\text{test}}$. This results in a coverage of 87.5% of $\mathcal{G}^{\text{test}}$, and a precision of 45%. In goals that do not belong to $\mathcal{G}^{\text{test}}$, goals of the form *grow* + {*any*} ∪ **color** + **furniture** ∪ **supplies** (e.g. *grow any lamp*) are *meaningful* to humans, but are not achievable in the environment (*impossible*).

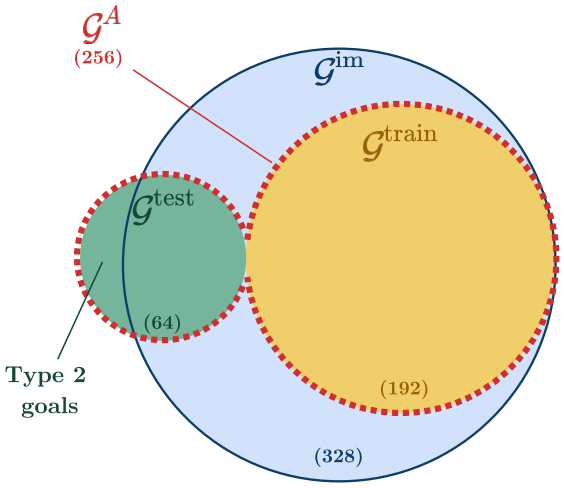


Figure B.3: Venn diagram of goal spaces.

Variants of Goal Imagination Mechanisms

Main Section 5.5.3 investigates variants of our goal imagination mechanism:

1. *Lower coverage:* To reduce the coverage of CGH while maintaining the same precision, we simply filter half of the goals that would have been imagined by CGH. This filtering is probabilistic, resulting in different imagined sets for different runs. It happens online, meaning that the coverage is always half of the coverage that CGH would have had at the same time in training.

2. *Lower precision:* To reduce precision while maintaining the same coverage, we sample a random sentence for each goal imagined by CGH. We control for vocabulary (words of $\mathcal{G}^{\text{train}}$) and sentence length. Goals from $\mathcal{G}^{\text{test}}$ are still imagined via the CGH mechanism. This variant only doubles the imagination of sentences that do not belong to $\mathcal{G}^{\text{test}}$.

3. *Oracle:* Perfect precision and coverage are achieved by filtering the output of CGH, keeping only goals from $\mathcal{G}^{\text{test}}$. Once the 56 goals that CGH can imagine are imagined, the oracle variant adds the 8 remaining goals: those including the word *flower* (Type 2 generalization).

4. *Random goals:* Each time CGH would have imagined a new goal, it is replaced by a randomly generated sentence, using words from the words of $\mathcal{G}^{\text{train}}$.

Note that all variants imagine goals at the same speed as the CGH algorithm. They simply filter or add noise to its output, see Figure B.4.

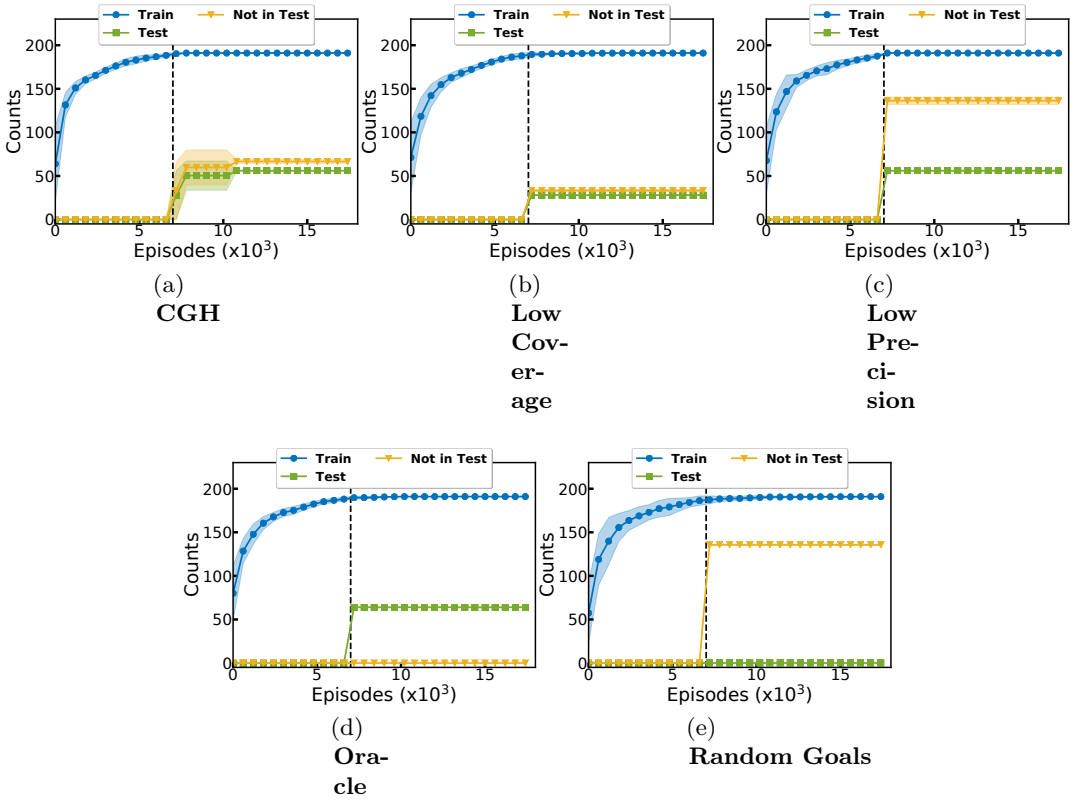


Figure B.4: Evolution of known goals for various goal imagination mechanisms. All graphs show the evolution of the number of goals from $\mathcal{G}^{\text{train}}$, $\mathcal{G}^{\text{test}}$ and others in the list of known goals $\mathcal{G}_{\text{known}}$. We zoom on the first epochs, as most goals are discovered and invented early. Vertical dashed line indicates the onset of goal imagination. (a) CGH; (b) Low Coverage; (c) Low precision; (d) Oracle; (e) Random Goals.

Effect of Low Coverage on Generalization

In Main Section 5.5.3, we compare our goal imagination mechanism to a *Low Coverage* variant that only covers half of the proportion of $\mathcal{G}^{\text{test}}$ covered by CGH (44%). Figure B.5 shows that the generalization performance on goals from $\mathcal{G}^{\text{test}}$ that the agent imagined (n-shot generalization, blue) are not significantly higher than the generalization performance on goals from $\mathcal{G}^{\text{test}}$ that were not imagined (zero-shot generalization). As they are both significantly higher than the *no imagination* baseline, this implies that training on imagined goals boosts zero-shot generalization on similar goals that were not imagined.

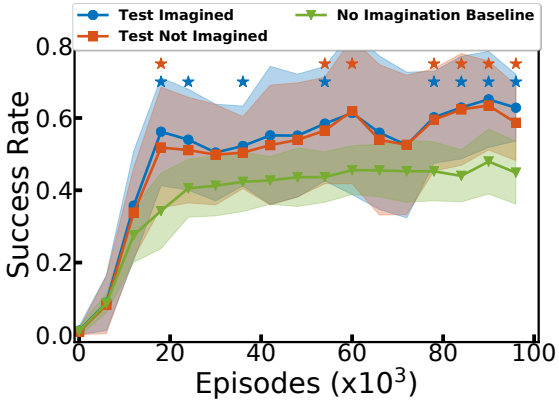


Figure B.5: Zero-shot versus n-shot. We look at the *Low Coverage* variant of our goal imagination mechanism that only covers 43.7% the test set with a 45% precision. We report success rates on testing goals of Type 5 (*grow + plant*) and compare with the *no imagination* baseline (green). We split in two: imagined goals (blue), and others (orange).

Impacts of Various Goal Imagination Mechanisms on Exploration

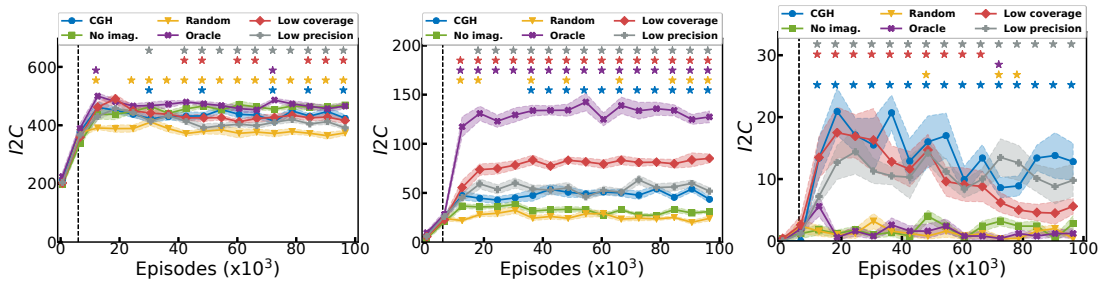
Figure B.6 presents the I2C exploration scores on the training, testing and extra sets for the different goal imagination mechanisms introduced in Main Section 5.5.3. Let us discuss each of these scores:

1. *Training interactions.* In Figure B.6a, we see that decreasing the precision (Low Precision and Random Goal conditions) affects exploration on interactions from the training set, where it falls below the exploration of the *no imagination* baseline. This is due to the addition of meaningless goals forcing agents to allocate less time to meaningful interactions in comparison.

2. *Testing interactions.* In Figure B.6b, we see that the highest exploration scores on interactions from the test set comes from the oracle. Because it shows high coverage and precision, it spends more time on interactions from the testing set. What is more surprising is the exploration score of the low coverage condition, higher than the exploration score of CGH. With equal precision, CGH should show better exploration, as it covers more test goals. However, the *Low Coverage* condition,

by spending more time exploring each of its imagined goals (it imagined fewer), probably learned to master them better, increasing the robustness of its behavior towards those. This insight advocates for the use of goal selection methods based on learning progress. Agents could estimate their learning progress on imagined goals using their internal reward function and its zero-shot generalization. Focusing on goals associated with high learning progress might help agents filter goals they can learn about from others.

3. *Extra interactions.* Figure B.6c shows that only the goal imagination mechanisms that invent goals not covered by the testing set manage to boost exploration in this extra set. The oracle perfectly covers the testing set but does not generate goals related to other objects (e.g. *grow any lamp*).



(a) **I2C on $\mathcal{G}^{\text{train}}$** (b) **I2C on $\mathcal{G}^{\text{test}}$** (c) **I2C on $\mathcal{G}^{\text{extra}}$**

Figure B.6: Exploration metrics for different goal imagination mechanisms: (a) Interesting interaction count (I2C) on training set, (b) I2C on testing set, (c) I2C on extra set. Goal imagination starts early (vertical line), except for the *no imagination* baseline (green). Standard errors of the mean plotted for clarity (as usual, 10 seeds).

B.5 Focus on Architectures

This section compares our proposed object-based modular architecture MA for the policy and reward function to a flat architecture that does not use inductive biases for efficient skill transfer. We hypothesize that only the object-based modular architectures enable a sufficient generalization performance for the goal imagination to impact generalization and exploration. Indeed, when generalization abilities are low, agents cannot evaluate their performance on imagined goals, thus cannot improve.

Preliminary Study of the Reward Function Architecture

We first compared modular and flat architectures for the reward function ($\text{MA}^{\mathcal{R}}$ vs $\text{FA}^{\mathcal{R}}$ in Figure B.7). This experiment was conducted in a supervised setting, independently from policy learning. We use a dataset of $50{\times}10^3$ trajectories and associated goal descriptions collected using a pre-trained policy. To closely match the training conditions of IMAGINE, we train the reward function on the final states s_T and test it on any states s_t, $t = [\![1, T]\!]$ of other episodes. Table B.3 provides the F_1-score computed at convergence on $\mathcal{G}^{\text{train}}$ and $\mathcal{G}^{\text{test}}$ for the two architectures.

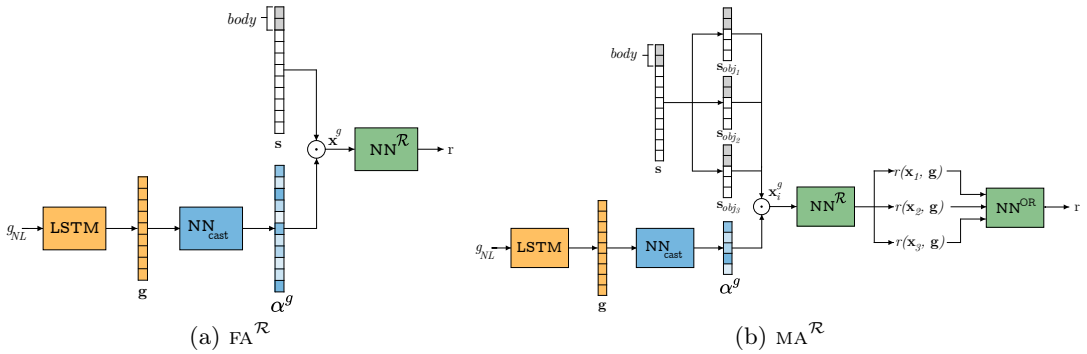


Figure B.7: Reward function architectures: (a) *Flat-attention* reward function ($\text{FA}^{\mathcal{R}}$) and (b) *Modular-attention* reward function ($\text{MA}^{\mathcal{R}}$). We use $\text{MA}^{\mathcal{R}}$ for all experiments except for the experiment in Table B.3

Table B.3: Reward function architectures performance.

	$F_{1\text{train}}$	$F_{1\text{test}}$
$\text{MA}^{\mathcal{R}}$	0.98 ± 0.02	0.64 ± 0.22
$\text{FA}^{\mathcal{R}}$	0.60 ± 0.10	0.22 ± 0.05

$\text{MA}^{\mathcal{R}}$ outperforms $\text{FA}^{\mathcal{R}}$ on both the training and testing sets. In addition to its poor generalization performance, $\text{FA}^{\mathcal{R}}$'s performance on the training set is too low to support policy learning. As a result, we always use $\text{MA}^{\mathcal{R}}$ for the reward function.

Policy Architecture Comparison

Let us now compare MA and FA architectures for the policy and critic (see Figure B.8). MA significantly outperforms FA on the training and testing sets at convergence (Table B.4) and generalizes better across learning as well (Figure B.9a). When combined with imagination, only MA demonstrates a significant boost in systematic generalization (see Main Table 5.2). Figure B.9c and B.9d support similar conclusions for exploration: only modular architectures enable goal imagination to boost exploration on the testing and extra sets of interactions.

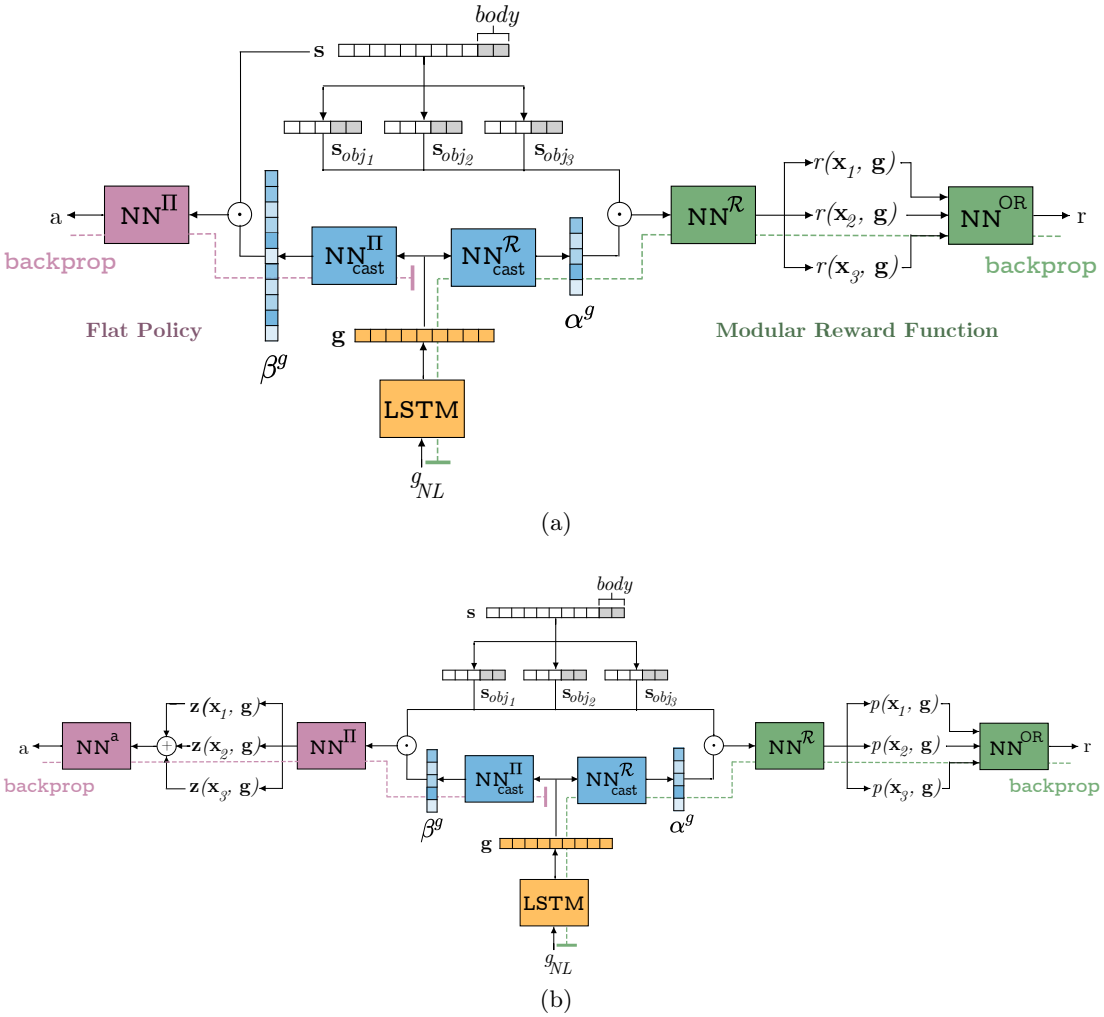


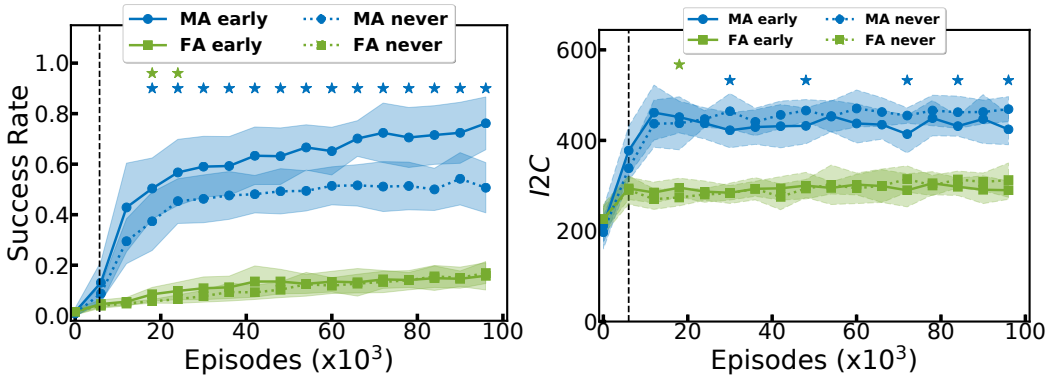
Figure B.8: Policy and reward function architectures: (a) *Modular-attention* (MA) reward + *Flat-attention* (FA) policy. (b) MA reward + MA policy. In both figures, the reward function is represented on the right in green, the policy on the left in pink, the language encoder at the bottom in yellow and the attention mechanisms at the center in blue.

In preliminary experiments, we tested a *Flat-Concatenation* (FC) architecture where the gated attention mechanism was replaced by a simple concatenation of the goal encoding and the state vector. We did not found a significant difference with respect to

FA. We chose to keep the attention mechanism, as it improves model interpretability (see Additional Visualization B.7).

Table B.4: Policy and critic architecture performance, using $\text{MA}^{\mathcal{R}}$ for the reward function. Differences between MA and FA are both significant, p-values $< 10^{-10}$.

	$\overline{\text{SR}}_{\text{train}}$	$\overline{\text{SR}}_{\text{test}}$
MA	0.95 ± 0.05	0.76 ± 0.10
FA	0.40 ± 0.13	0.16 ± 0.06



(a) $\overline{\text{SR}}_{\text{test}}$ (b) $\text{I2C}_{\text{train}}$

(c) I2C_{test} (d) $\text{I2C}_{\text{extra}}$

Figure B.9: Policy architecture comparison: (a) $\overline{\text{SR}}$ on $\mathcal{G}^{\text{test}}$ for the FA and MA architectures when the agent starts imagining goals early (plain, after the black vertical dashed line) or never (dashed). (b, c, d) I2C on interactions from the training, testing and extra sets respectively. Imagination is performed using CGH. Stars indicate significant differences between CGH and the corresponding *no imagination* baseline.

B.6 Focus on Reward Function

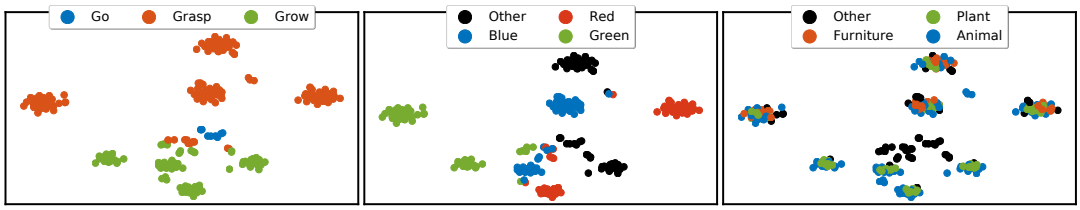
The reward function learned by IMAGINE can be used for several purposes. This paper leverages some of these ideas (the first two), while others are left for future work (the last two):

- **Behavior adaptation.** As Main Section 5.5.1 showed, the reward function enables agents to adapt their behavior towards imagined goals. Whereas the zero-shot generalization pushed agents to grow plants with food and water with equal probability, the reward function helped agents to correct that behavior towards more water.
- **Guiding hindsight experience replay (HER).** In multi-goal RL with discrete sets of goals, HER is traditionally used to modify transitions sampled from the replay buffer. It replaces originally targeted goals with others randomly selected from the set of goals (Andrychowicz et al., 2017; Mankowitz et al., 2018). This enables transfer across skills by reinterpreting trajectories in the light of new goals. In that case, a reward function is required to compute the reward associated with that new transition (new goal). To improve on random goal replay, we favor goal substitution towards goals that actually match the state and have a higher chance of being rewarded. In IMAGINE, we scan a set of 40 goal candidates for each transition, and select substitute goals that match the scene when possible, with probability $p = 0.5$.
- **Exploring like Go-Explore.** In Go-Explore (Ecoffet et al., 2021), agents first reach a goal state, then start exploring from there. We could reproduce that behavior in our IMAGINE agents with our internal reward function. The reward function would scan each state during the trajectory. When the agent thinks it reached the goal, it can can switch to another, add noise on its goal embedding, or increase the exploration noise on actions. This might enable agents to explore sequences of goal-directed behaviors. We leave the study of this mechanism for future work.
- **Filtering of imagined goals.** When generating imagined goals, agents also generate meaningless goals. Ideally, we would like agents to filter these from meaningful goals. Goals are said meaningful if the reward function generalizes well enough for the agent to progress on the corresponding skill. In an ensemble of reward functions, most functions would agree on the interpretation of imagined goals close to those found in the training set. On the other hand, they might disagree on others. Agents could use such ensemble to filter appropriate goals from others by scanning a dataset of interactions and evaluating the agreement for given imagined goals. Having an efficient filtering mechanism would drastically improve the efficiency of goal imagination, as Main Section 5.5.3 showed that the ratio of meaningful goals determines generalization performance. This is also left for future work.

B.7 Additional Visualizations

Visualizing Goal Embedding

To analyze the goal embeddings learned by the language encoder L_e, we perform a t-SNE using 2 components, perplexity 20, a learning rate of 10 for 5000 iterations. Figure B.10 presents the resulting projection for a particular run. The embedding seems to be organized mainly in terms of motor predicates (B.10a) and colors (B.10b). Object types or categories do not seem to be strongly represented (B.10c).



(a) (b) (c)

Figure B.10: t-SNE of Goal Embedding. The same t-SNE is presented, with different color codes (a) predicates, (b) colors, (c) object categories.

Visualizing Attention Vectors

In the *modular-attention* architectures for the reward function and policy, we train attention vectors to gate object-specific features via a term-by-term product. In each architecture, the attention vector is shared across objects (permutation invariance). Figure B.11 presents examples of attention vectors for the reward function (B.11a) and for the policy (B.11b) at the end of training. These attention vectors highlight relevant parts of the object-specific sub-state depending on the linguistic goal:

- When the sentence refers to a particular object type (e.g. *dog*) or category (e.g. *living thing*), the attention vector suppresses the corresponding object type(s) and highlights the complement set of object types. If the object does not match the object type or category described in the sentence, the output of the term-by-term product between object types and attention will be close to 1. Conversely, if the object is of the required type, the attention suppression ensures that the output stays close to zero. Although it might not be intuitive for humans, it efficiently detects whether the considered object is the one the sentence refers to.
- When the sentence refers to a navigation goal (e.g. *go top*), the attention highlights the agent's position (here y).
- When the sentence is a *grow* goal, the reward function focuses on the difference in object's size, while the policy further highlights the object's position.

The attention vectors use information about the goal to highlight or suppress parts of the input using the different strategies described above depending on the type of input

(object categories, agent's position, size difference, etc). This type of gated attention improves the interpretability of the reward function and policy.

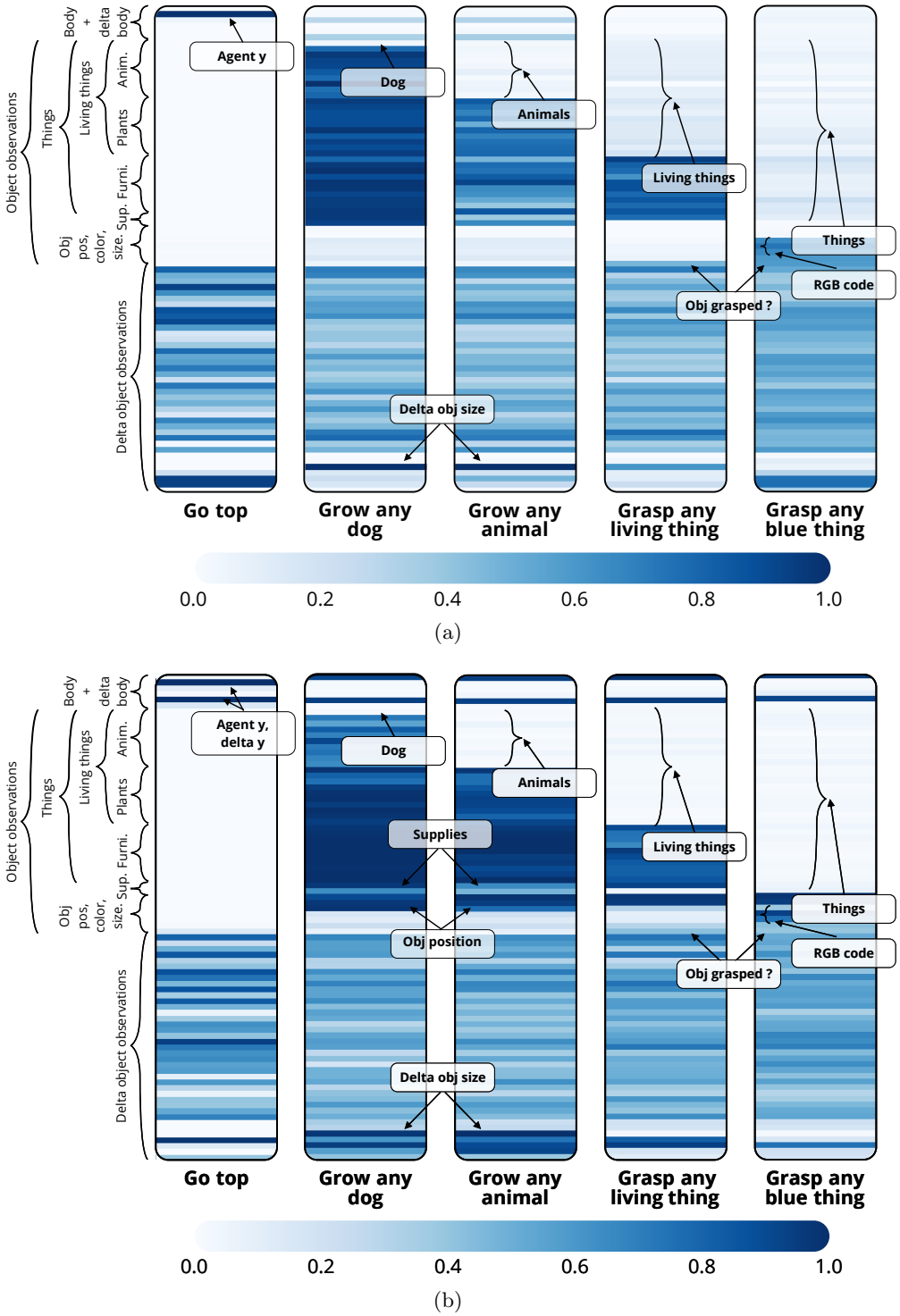


Figure B.11: Attention vectors (a) $\boldsymbol{\alpha}^g$ for the reward function (1 seed). (b) $\boldsymbol{\beta}^g$ for the policy (1 seed).

B.8 Comparing IMAGINE to Goal-as-State Approaches

In the goal-conditioned RL literature, some works have proposed goal generation mechanisms to facilitate the acquisition of skills over large sets of goals (Nair et al., 2018b; Pong et al., 2020; Nair et al., 2020). Some of them had a special interest in exploration and proposed to bias goal sampling towards goals from low-density areas (Pong et al., 2020). One might then think that IMAGINE should be compared to these approaches. However, there are a few catches:

1. Nair et al. (2018b, 2020); Pong et al. (2020) use generative models of states to sample state-based goals. However, our environment is procedurally generated. This means that sampling a given state from the generative model has a very low probability to *match* the scene. If the present objects are three red cats, the agent has no chance to reach a goal specifying dogs and lions' positions, colors, and sizes. Indeed, most of the state space is made of object features that cannot be acted upon (colors, types, sizes of most objects). One could imagine using SP to organize the scene, but we would need to ask SP to find the three objects specified by the generated goal, in the exact colors (RGB codes) and size. By doing so, there would be no distracting object for the agent to discover and learn about. A second option is to condition the goal generation on the scene as it is done in Nair et al. (2020). The question of whether it might work in procedurally-generated environments remains open.

2. Assuming a perfect goal generator that only samples valid goals that do not ask for a change of object color or type, the agent would then need to bring each object to its target position and to grow objects to their very specific goal size. These goals are not the same as those targeted by IMAGINE, they are too specific. These approaches—like most goal-conditioned RL approaches—represent goals as particular states (e.g. block positions in manipulation tasks, visual states in navigation tasks) (Schaul et al., 2015; Andrychowicz et al., 2017; Nair et al., 2018b; Pong et al., 2020). In contrast, language-conditioned agents represent abstract goals, usually defined by specific constraints on states (e.g. *grow any plant* requires the size of at least one plant to increase) (Chan et al., 2019; Jiang et al., 2019; Cideron et al., 2020b). For this reason, *goal-as-state* and *abstract-goal* approaches do not tackle the same problem. The former target specific coordinates, and cannot be instructed to reach abstract goals, while the latter are not trained to reach specific states.

For these reasons, we argue that the goal-conditioned approaches that use state-based goals cannot be easily or fairly compared to our approach IMAGINE.

B.9 Implementation Details

Reward Function Inputs and Hyperparameters

Supplementary Section B.5 details the architecture of the reward function. The following provides extra details about the inputs. The object-dependent sub-state $\mathbf{s}_{obj(i)}$ contains information about both the agent's body and the corresponding object i: $\mathbf{s}_{obj(i)} = [\mathbf{o}_{body}, \Delta\mathbf{o}_{body}, \mathbf{o}_{obj(i)}, \Delta\mathbf{o}_{obj(i)}]$ where $\mathbf{o}_{body}$ and $\mathbf{o}_{obj(i)}$ are body- and obj_i-dependent observations, and $\Delta\mathbf{o}^t_{body} = \mathbf{o}^t_{body} - \mathbf{o}^0_{body}$ and $\Delta\mathbf{o}^t_{obj(i)} = \mathbf{o}^t_{obj(i)} - \mathbf{o}^0_{obj(i)}$ measure the difference between the initial and current observations. The second input is the attention vector $\boldsymbol{\alpha}^g$ that is integrated with $\mathbf{s}_{obj(i)}$ through an Hadamard product to form the model input: $\mathbf{x}^g_i = \mathbf{s}_{obj(i)} \odot \boldsymbol{\alpha}^g$. This attention vector is a simple mapping from $\mathbf{g}$ to a vector of the size of $\mathbf{s}_{obj(i)}$ contained in $[0,1]^{size(\mathbf{s}_{obj(i)})}$. This cast is implemented by a one-layer neural network with sigmoid activations $\texttt{NN}^{\text{cast}}$ such that $\boldsymbol{\alpha}^g = \texttt{NN}^{\text{cast}}(\mathbf{g})$.

For the three architectures, the number of hidden units of the `LSTM` and the sizes of the hidden layers of fully connected networks are fixed to 100. `NN` parameters are initialized using He initialization (He et al., 2015) and we use one-hot word encodings. The `LSTM` is implemented using `rnn.BasicLSTMCell` from TensorFlow 1.15 based on Zaremba et al. (2014). The states are initially set to zero. The `LSTM`'s weights are initialized uniformly from $[-0.1, 0.1]$ and the biases initially set to zero. The `LSTM` use a *tanh* activation function whereas the `NN` are using ReLU activation functions in their hidden layers and sigmoids at there output.

Reward Function Training Schedule

The architecture is trained via backpropagation using the Adam Optimizer (Kingma & Ba, 2015). The data is fed to the model in batches of 512 examples. Each batch is constructed so that it contains at least one instance of each goal description g_{NL} (goals discovered so far). We also use a modular buffer to impose a ratio of positive rewards of 0.2 for each description in each batch. When trained in parallel with the policy, the reward function is updated once every 1200 episodes. Each update corresponds to up to 100 training epochs (100 batches). We implement a stopping criterion based on the F_1-score computed from a held-out test set uniformly sampled from the last episodes (20% of the last 1200 episodes (2 epochs)). The update is stopped when the F_1-score on the held-out set does not improve for 10 consecutive training epochs.

RL Implementation and Hyperparameters

In the policy and critic architectures, we use hidden layers of size 256 and ReLU activations. Attention vectors are cast from goal embeddings using single-layer neural networks with sigmoid activations. We use the He initialization scheme for (He et al., 2015) and train them via backpropagation using the Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) (Kingma & Ba, 2015).

Our learning algorithm is built on top of the OpenAI Baselines implementation of

HER-DDPG.[1] We leverage a parallel implementation with 6 actors. Actors share the same policy and critic parameters but maintain their own memory and conduct their own updates independently. Updates are then summed to compute the next set of parameters broadcast to all actors. Each actor is updated for 50 epochs with batches of size 256 every 2 episodes of environment interactions. Using hindsight replay, we enforce a ratio $p = 0.5$ of transitions associated with positive rewards in each batch. We use the same hyperparameters as Plappert et al. (2018a).

Computing Resources

The RL experiments contain 8 conditions of 10 seeds each and 4 conditions with 5 seeds (SP study). Each run leverages 6 CPUs (6 actors) for about 36h for a total of 2.5 cpu years. Experiments presented in this paper require machines with at least 6 CPU cores.

[1] The OpenAI Baselines implementation of HER-DDPG can be found at https://github.com/openai/baselines, our implementation can be found at https://sites.google.com/view/imagine-drl.[2]

Appendix C

Appendix of LGB

C.1 Pseudo-Code

Algorithm 5 presents the high-level pseudo-code of LGB algorithms for each of the three phases.

Algorithm 5 LGB algorithm

▷ Goal → Behavior phase
Require Env E
Initialize policy Π, goal sampler G_s, buffer B
loop
  $g \leftarrow G_s$.sample()
  $\{s, a, s', g, c_p, c'_p\}_{[\![1,T]\!]} \leftarrow E$.rollout$(\Pi, g)$
  G_s.update(c_p^T)
  B.update$(\{s, a, s', g, c_p, c'_p\}_{[\![1,T]\!]})$
  Π.update(B)
return Π, G_s

▷ Language → Goal phase
Require Π, E, G_s, social partner SP
Initialize language goal generator LGG
dataset $\leftarrow SP$.interact(E, Π, G_s)
LGG.update(dataset)
return LGG

▷ Language → Behavior phase
Require $E, \Pi,$ LGG, SP
loop
  descr. $\leftarrow SP$.listen()
  loop ▷ Strategy switching loop
    $g \leftarrow$ LGG.sample(descr, c^0)
    $c_p^T \leftarrow E$.rollout(g)
    if $g == c_p^T$ **then break**

C.2 Semantic Predicates and Application to Fetch Manipulate

In this paper, we restrict the semantic representations to the use of the *close* and *above* binary predicates applied to $M = 3$ objects. The resulting semantic configurations are formed by:

$$c_p = [c(o_1, o_2),\ c(o_1, o_3),\ c(o_2, o_3),\ a(o_1, o_2),\ a(o_2, o_1),\ a(o_1, o_3),\ a(o_3, o_1),\ a(o_2, o_3),\ a(o_3, o_2)],$$

where *c()* and *a()* refer to the *close* and *above* predicates respectively and (o_1, o_2, o_3) are the red, green and blue blocks respectively.

Symmetry and asymmetry of *close* and *above* predicates. We consider objects o_1 and o_2:

- *close* is symmetric: "o_1 is **close** to o_2" $\Leftrightarrow$ "o_2 is **close** to o_1." The corresponding semantic mapping function is based on the Euclidean distance, which is symmetric.
- *above* is asymmetric: "o_1 is **above** o_2" $\Rightarrow$ **not** "o_2 is **above** o_1." The corresponding semantic mapping function evaluates the sign of the difference of the object Z-axis coordinates.

C.3 The DECSTR Algorithm

C.3.1 Autotelic Reinforcement Learning

The sensorimotor learning phase (G→B) of DECSTR alternates between two steps:

- **Data acquisition.** A DECSTR agent has no prior on the set of reachable semantic configurations. Its first goal is sampled uniformly from the semantic configuration space. Using this goal, it starts interacting with its environment, generating trajectories of sensory states s, actions a and configurations c_p. The last configuration c_p^T achieved in the episode after T time steps is considered stable and is added to the set of reachable configurations. As it interacts with the environment, the agent explores the configuration space, discovers reachable configurations and selects new targets.
- **Internal models updates.** A DECSTR agent updates two models: its curriculum strategy and its policy. The curriculum strategy can be seen as an active goal sampler. It biases the selection of goals to target and goals to learn about. The policy is the module controlling the agent's behavior and is updated via RL.

Policy Updates with a Goal-Conditioned Soft Actor-Critic

Policy updates are performed with Soft Actor-Critic (SAC) (Haarnoja et al., 2018b), a state-of-the-art off-policy actor-critic algorithm. We also use hindsight experience replay

(HER) (Andrychowicz et al., 2017). This mechanism enables agents to learn from failures by reinterpreting past trajectories in the light of goals different from the ones originally targeted. HER was designed for continuous goal spaces, but can be directly transposed to discrete goals (Chapter 3). In the transitions to be fed to SAC, we simply replace the originally targeted goal configuration with a configuration achieved later in the trajectory. We also use our automatic curriculum strategy: the LP-C-based probabilities are used to sample goals to learn about. When a goal g is sampled, we search the experience buffer for the collection of episodes that ended in the configuration $c_p = g$. From these episodes, we sample a transition uniformly. This hindsight substitute of g has high chances of being the sampled one. At least, it is a configuration on the path towards this goal, as it is sampled from a trajectory leading to it. The HER mechanism is thus biased towards goals sampled by the agent.

Object-Centered Inductive Biases

In the proposed *Fetch Manipulate* environment, the three blocks share the same set of attributes (position, velocity, color identifier). Thus, it is natural to encode a *relational inductive bias* in our architecture. The behavior with respect to a pair of objects should be independent of the position of the objects in the inputs. The architecture used for the policy is depicted in Figure 6.3.

A shared network (NN_{shared}) encodes the concatenation of: 1) agent's body features; 2) object pair features; 3) current configuration (c_p) and 4) current goal g. This is done independently for all object pairs. No matter the location of the features of the object pair in the initial observations, this shared network ensures that the same behavior will be performed, thus skills are transferred between object pairs. A sum is then used to aggregate these outputs, before a final network (NN_{policy}) maps the aggregation to actions a. The critic follows the same architecture, where a final network NN_{critic} maps the aggregation to an action-value Q. Parallel encoding of each pair-specific inputs can be seen as different modules trying to reach the goal by only seeing these pair-specific inputs. The intuition is that modules dealing with the pair that should be acted upon to reach the goal will supersede others in the sum aggregation.

Although in principle our architecture could work with combinations of objects (3 modules), we found permutations to work better in practice (6 modules). With combinations, the shared network would need to learn to put block A on block B to achieve a predicate $above(o_i, o_j)$, and would need to learn the reverse behavior (put B on A) to achieve the symmetric predicate $above(o_j, o_i)$. With permutations, the shared network can simply learn one of these behaviors (e.g. A on B). Considering the predicate $above(o_A, o_B)$, at least one of the modules has objects organized so that this behavior is the good one: if the permutation (o_B, o_A) is not the right one, permutation (o_A, o_B) is. The symmetry bias is explained in Section 6.2.2. It leverages the symmetry of the behaviors required to achieve the predicates $above(o_i, o_j)$ and $above(o_j, o_i)$. As a result, the two goal configurations are:

$$g_1 = [c(o_1, o_2),\ c(o_1, o_3),\ c(o_2, o_3),\ a(o_1, o_2),\ a(o_1, o_3),\ a(o_2, o_3)],$$

$$g_2 = [c(o_1, o_2),\ c(o_1, o_3),\ c(o_2, o_3),\ a(o_2, o_1),\ a(o_3, o_1),\ a(o_3, o_2)],$$

where g_1 is used in association with object permutations (o_i, o_j) with $i < j$ and g_2 is used in association with object permutations (o_j, o_i) with $i < j$. As a result, the shared network automatically ensures transfer between predicates based on symmetric behaviors.

Implementation Details

This part includes details necessary to reproduce results. The code is available at https://sites.google.com/view/decstr/.

Parallel implementation of SAC-HER. We use a parallel implementation of SAC (Haarnoja et al., 2018b). Each of the 24 parallel workers maintains its own replay buffer of size 10^6 and performs its own updates. Updates are summed over the 24 actors and the updated network is broadcast to all workers. Each worker alternates between 2 episodes of data collection and 30 updates with batch size 256. To form an epoch, this cycle is repeated 50 times and followed by the offline evaluation of the agent on each reachable goal. An epoch is thus made of $50 \times 2 \times 24 = 2400$ episodes.

Goal sampler updates. The agent performs self-evaluations with probability $self\_eval = 0.1$. During these runs, the agent targets uniformly sampled discovered configurations without exploration noise. This enables the agent to self-evaluate on each goal. Goals are organized into buckets. Main Section 6.2.2 presents our automatic bucket generation mechanism. Once buckets are formed, we compute C, LP, and P, based on windows of the past $W = 1800$ self-evaluation interactions for each bucket.

Modular architecture. The shared network of our modular architecture NN_{shared} is a 1-hidden layer network of hidden size 256. After all pair-specific inputs have been encoded through this module, their outputs (of size 84) are summed. The sum is then passed through a final network with a hidden layer of size 256 to compute the final actions (policy) or action-values (critic). All networks use $ReLU$ activations and the Xavier initialization. We use Adam optimizers, with learning rates 10^{-3}. The list of hyperparameters is provided in Table C.1.

Computing resources

The sensorimotor learning experiments contain 8 conditions: 2 of 10 seeds and 6 of 5 seeds. Each run leverages 24 CPUs (24 actors) for about 72h for a total of 9.8 CPU years. Experiments presented in this paper require machines with at least 24 cpu cores. The language grounding phase runs on a single CPU and trains in a few minutes.

C.3.2 Language-Conditioned Goal Generator

Language-Conditioned Goal Generator Training

We use a conditional Variational Auto-Encoder (C-VAE) (Sohn et al., 2015). Conditioned on the initial configuration and a sentence describing the expected transformation

Table C.1: Sensorimotor learning hyperparameters used in DECSTR.

Hyperparam.	Description	Values.
nb_mpis	Number of workers	24
nb_cycles	Number of repeated cycles per epoch	50
nb_rollouts_per_mpi	Number of rollouts per worker	2
nb_updates	Number of updates per cycle	30
start_bias_init	Epoch from which initializations are biased	100
W	Curriculum window size	1800
self_eval	Self evaluation probability	0.1
N_b	Number of buckets	5
replay_strategy	HER replay strategy	*future*
k_replay	Ratio of HER data to data from normal experience	4
batch_size	Size of the batch during updates	256
γ	Discount factor to model uncertainty about future decisions	0.98
τ	Polyak coefficient for target critics smoothing	0.95
lr_actor	Actor learning rate	10^{-3}
lr_critic	Critic learning rate	10^{-3}
α	Entropy coefficient used in SAC	0.2
automatic_entropy	Automatically tune the entropy coefficient	*False*

of one object relation, it generates compatible goal configurations. After the first phase of goal-directed sensorimotor training, the agent interacts with a hard-coded social partner as described in Main Section 6.2.2. From these interactions, we obtain a dataset of 5000 triplets: initial configuration, final configuration and sentence describing one change of predicate from the initial to the final configuration. The list of sentences used by the synthetic social partner is provided in Table C.2. Note that *red*, *green* and *blue* refer to objects o_1, o_2, o_3 respectively.

Content of the Test Sets

We describe the 5 test sets:

1. Test set 1 is made of input pairs (c_i, s) from the training set, but tests the coverage of all compatible final configurations $\mathcal{C}_f$, 80% of which are not found in the training set. In that sense, it is partly a test set.

2. Test set 2 contains two input pairs: {[0 1 0 0 0 0 0 0 0], *put blue close_to green*} and {[0 0 1 0 0 0 0 0 0], *put green below red*} corresponding to 7 and 24 compatible final configurations respectively.

3. Test set 3 corresponds to all pairs including the initial configuration $c_i = [1\,1\,0\,0\,0\,0\,0\,0\,0]$ (29 pairs), with an average of 13 compatible final configurations.

4. Test set 4 corresponds to all pairs including one of the sentences *put green on_top_of red* and *put blue far_from red*, i.e. 20 pairs with an average of 9.5 compatible final configurations.

5. Test set 5 includes all pairs that include both the initial configuration of test set 3 and one of the sentences of test set 4, i.e. 2 pairs with 6 and 13 compatible goals respectively. Note that pairs of set 5 are removed from sets 3 and 4.

Table C.2: List of instructions. Each of them specifies a shift of one predicate, either from false to true ($0 \rightarrow 1$) or true to false ($1 \rightarrow 0$). **block A** and **block B** represent two different blocks from {red, blue, green}.

Transition type	Sentences
Close $0 \rightarrow 1$ ($\times 3$)	*Put* ***block A*** *close_ to* ***block B***, *Bring* ***block A*** *and* ***block B*** *together*, *Get* ***block A*** *and* ***block B*** *close_from each_ other*, *Get* ***block A*** *close_ to* ***block B***.
Close $1 \rightarrow 0$ ($\times 3$)	*Put* ***block A*** *far_from* ***block B***, *Get* ***block A*** *far_from* ***block B***, *Get* ***block A*** *and* ***block B*** *far_from each_ other*, *Bring* ***block A*** *and* ***block B*** *apart*,
Above $0 \rightarrow 1$ ($\times 6$)	*Put* ***block A*** *above* ***block B***, *Put* ***block A*** *on_ top_ of* ***block B***, *Put* ***block B*** *under* ***block A***, *Put* ***block B*** *below* ***block A***.
Above $1 \rightarrow 0$ ($\times 6$)	*Remove* ***block A*** *from_ above* ***block B***, *Remove* ***block A*** *from* ***block B***, *Remove* ***block B*** *from_ below* ***block A***, *Put* ***block B*** *and* ***block A*** *on_ the_ same_ plane*, *Put* ***block A*** *and* ***block B*** *on_ the_ same_ plane*.

Table C.3: List of abstract instructions. 'construction' refers to either a stack or a pyramid, 'put X on top' refers to any construction where the top block has color X.

Abstract instructions
Put them all close, Get them all close,
Make a tower, Build a tower, Stack some blocks,
Make a stack of two, Make a tower of two, Build a stack of two,
Build a tower of two, Stack two blocks,
Make a stack of three, Make a tower of three, Build a stack of three,
Build a tower of three, Stack three blocks,
Make a pyramid, Build a pyramid,
Make a construction, Build a construction,
Put red on top, Get red on top, Put green on top, Get green on top,
Put blue on top, Get blue on top.

Testing on Logical Expressions of Instructions

To evaluate DECSTR on logical functions of instructions, we generate three types of expressions:

1. 100 instructions of the form "A and B" where A and B are basic instructions corresponding to shifts of the form *above* $0 \rightarrow 1$ (see Table C.2). These intersections correspond to stacks of 3 or pyramids.
2. 200 instructions of the form "A and B" where A and B are *above* and *close* instructions respectively. B can be replaced by "not B" with probability 0.5.
3. 200 instructions of the form "(A and B) or (C and D))," where A, B, C, D are basic instructions: A and C are *above* instructions while B and D are *close* instructions. Here also, any instruction can be replaced by its negation with probability 0.5.

Implementation Details

The encoder is a fully connected neural network with two layers of size 128 and $ReLU$ activations. It takes as input the concatenation of the final binary configuration and its two conditions: the initial binary configuration and an embedding of the description. The description is embedded with a recurrent network with embedding size 100, *tanh* non-linearities and biases. The encoder outputs the mean and log-variance of the latent distribution of size 27. The decoder is also a fully connected network with two hidden layers of size 128 and $ReLU$ activations. It takes as input the latent code z and the same conditions as the encoder. As it generates binary vectors, the last layer uses *sigmoid* activations. We train the architecture with a mixture of Kullback-Leibler divergence loss (KD_{loss}) w.r.t a standard Gaussian prior and a binary Cross-Entropy loss (BCE_{loss}). The combined loss is $BCE_{\text{loss}} + \beta \times KD_{\text{loss}}$ with $\beta = 0.6$. We use an Adam optimizer, a learning rate of 5×10^{-4}, a batch size of 128 and optimize for 150 epochs. As training is fast (≈ 2 min on a single cpu), we conducted a quick hyperparameter search over β, layer sizes, learning rates and latent sizes (see Table C.4). We found robust results for various layer sizes, various β below 1. and latent sizes above 9.

Table C.4: LGG hyperparameter search. In bold are the selected hyperparameters.

Hyperparam.	Values.
β	[0.5, **0.6**, 0.7, 0.8, 0.9, 1.]
layers size	[**128**, 256]
learning rate	[0.01, **0.005**, 0.001]
latent sizes	[9, 18, **27**]

C.4 Baselines and Oracle

The language-conditioned LB baseline is fully described in the main document.

C.4.1 Expert Buckets Oracle

In the EXPERT BUCKETS oracle, the automatic bucket generation of DECSTR is replaced with an expert-predefined set of buckets using *a priori* measures of similarity and difficulty. To define these buckets, one needs prior knowledge of the set of unreachable configurations, which are ruled out. The 5 predefined buckets contain all configurations characterized by:

- Bucket 1: a single *close* relation between a pair of objects and no *above* relations (4 configurations).
- Bucket 2: 2 or 3 *close* relations and no *above* relations (4 configurations).
- Bucket 3: 1 stack of 2 blocks and a third block that is either away or close to the base, but is not close to the top of the stack (12 configurations).
- Bucket 4: 1 stack of 2 blocks and the third block close to the stack, as well as pyramid configurations (9 configurations).
- Bucket 5: stacks of 3 blocks (6 configurations).

These buckets are the only difference between the EXPERT BUCKETS baseline and DECSTR.

C.4.2 LGB-C Baseline

The LGB-C baseline represent goals not as semantic configurations but as particular 3D targets positions for each block, as defined for example in Lanier et al. (2019) and Li et al. (2020). The goal vector size is also 9 and contains the 3D target coordinates of the three blocks. This baseline also implements decoupling and, thus, can be compared to DECSTR in the three phases. We keep as many modules as possible common with DECSTR to minimize the amount of confounding factors and reduce the *under-fitting* bias. The goal selection is taken from DECSTR, but converts semantic configuration into specific randomly-sampled target coordinates for the blocks, see Figure C.1. The agent is not conditioned on its current semantic configuration nor its semantic goal configuration. For this reason, we do not apply the symmetry bias. The binary reward is positive when the maximal distance between a block and its target position is below 5 cm, i.e. the size of a block (similar to Andrychowicz et al., 2017). To make this baseline competitive, we integrate methods from a state-of-the-art block manipulation algorithm (Lanier et al., 2019). The agent receives positive rewards of 1, 2, 3 when the corresponding number of blocks is well placed. We also introduce the multi-criteria HER from Lanier et al. (2019). Finally, we add an additional object-centered inductive bias by only considering, for each Deep Sets module, the 3D target positions of the corresponding pair.That is, for each object pair, we ignore the 3D positions of the remaining object, yielding to a vector of size 6. Language grounding is based on a C-VAE similar to the one used by DECSTR. We only replace the cross-entropy loss with a mean-squared loss due to the continuous nature of the target goal coordinates. We use the exact same training and testing sets as with semantic goals.

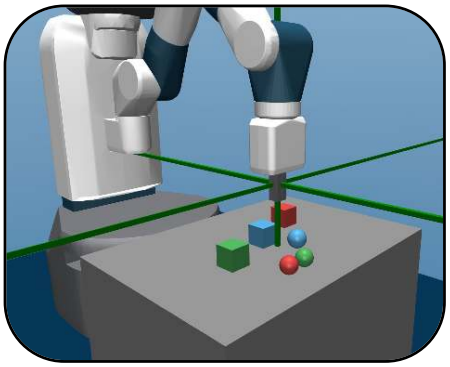


Figure C.1: The LGB-C baseline samples target positions for each block (example for a pyramid here).

C.5 Additional Results

C.5.1 DECSTR Learning Trajectories

Figure C.2 shows the evolution of internal estimations of the competence C, the learning progress LP and the associated sampling probabilities P. Note that these metrics are computed online by DECSTR, as it self-evaluates on random discovered configurations. Learning trajectories seem to be uniform across different runs, and buckets are learned in increasing order. This confirms that the time of discovery is a good proxy for goal difficulty. In that case, configurations discovered first end up in the lower index buckets and are indeed learned first. Note that a failing automatic bucket generation would assign goals to random buckets. This would result in uniform measures of learning progress across different buckets, which would be equivalent to uniform goal sampling. As Main Figure 6.4c shows, DECSTR performs much better than the *random goals* conditions. This proves that our automatic bucket algorithm generates useful goal clustering.

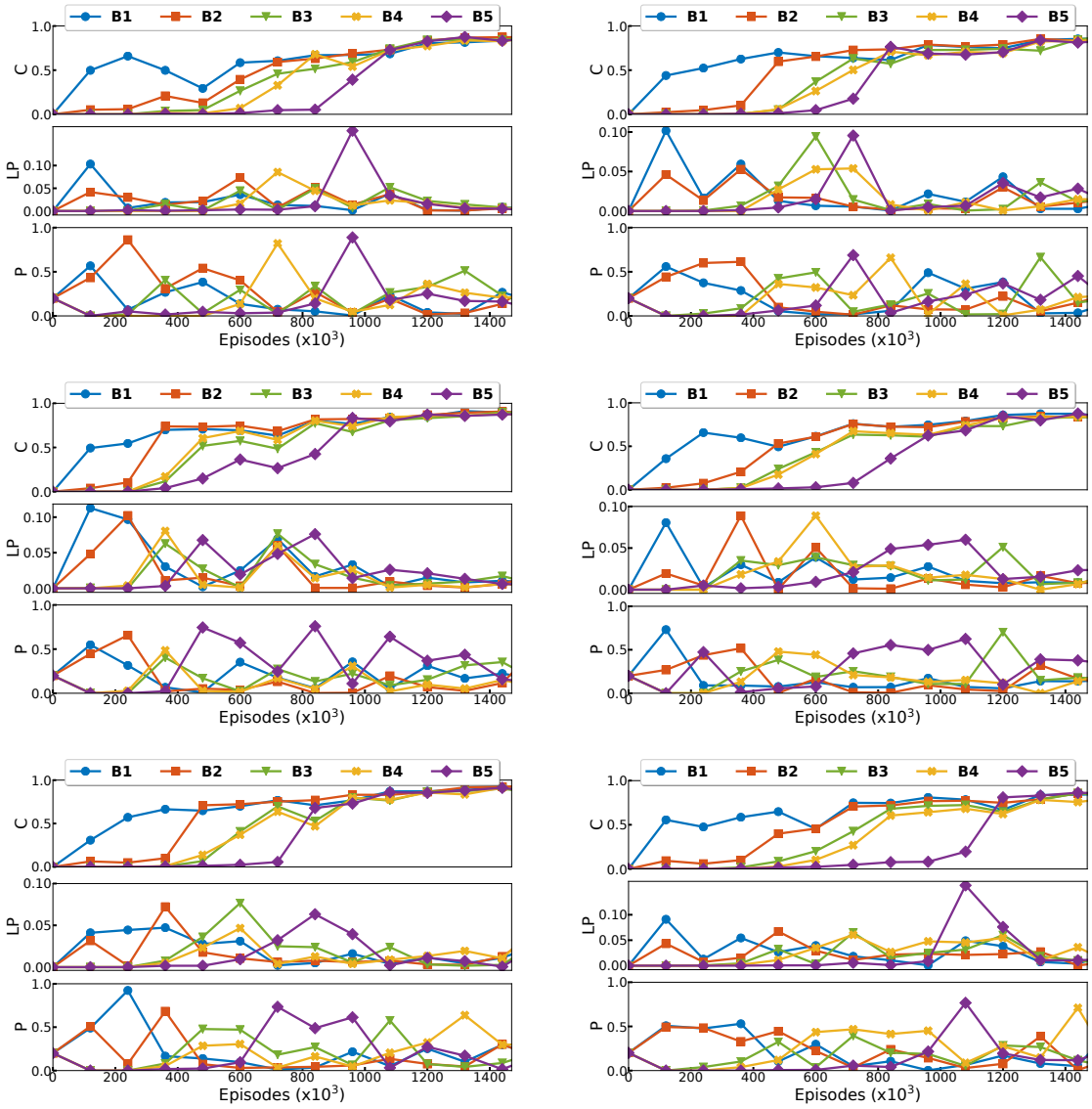


Figure C.2: Learning trajectories of 6 DECSTR agents.

Appendix D

References for Figure 1.14

Table D.1: References for the labels in Figure 1.14.

Label	Reference	Label	Reference
ACTRCE	Chan et al. (2019)	Agent57	Badia et al. (2020a)
AGILE	Bahdanau et al. (2019b)	AlfWorld	Shridhar et al. (2021)
AMIGO	Campero et al. (2021)	Ask Your Human	Chen et al. (2021)
Asym. Self-Play	Sukhbaatar et al. (2018)	BabyAI	Chevalier-Boisvert et al. (2019)
CLIC	Fournier et al. (2019)	CMA-ES	Hansen (2016)
Count-based DQN	Bellemare et al. (2016)	CURIOUS	Chapter 3
CWYC	Blaes et al. (2019)	DADS	Sharma et al. (2020)
DCEM	Hill et al. (2020b)	DDPG	Lillicrap et al. (2016)
DECSTR	Chapter 6	Deep GA	Santucci et al. (2013)
DIAYN	Eysenbach et al. (2019)	Disagreement	Pathak et al. (2019)
DISCERN	Warde-Farley et al. (2019)	DQN	Mnih et al. (2015)
EDL	Campos et al. (2020)	ES	Rechenberg (1973)
Evo-ES	Gajewski et al. (2019)	Feudal RL	Dayan & Hinton (1993)
GA	Holland (1992)	GEP-PG	Chapter 2
GMM-IMGEP	Moulin-Frier et al. (2014)	Go-Explore	Ecoffet et al. (2021)
Goal-GAN	Florensa et al. (2018)	GR-IMGEP	Kovač et al. (2020)
HAC	Levy et al. (2019)	HACOB	Forestier & Oudeyer (2016a)
Hermann	Hermann et al. (2017)	HER	Andrychowicz et al. (2017)
HIGHER	Cideron et al. (2020b)	HILBERT	Pierrot et al. (2020)
HIR	Jiang et al. (2019)	HIRO	Nachum et al. (2018)
HOLMES	Etcheverry et al. (2020)	ICM	Pathak et al. (2017)
IMAGINE	Chapter 5	Kaelbling	Kaelbling (1993)
Lang-DQN	Narasimhan et al. (2015)	Lanier	Lanier et al. (2019)
LfP	Lynch & Sermanet (2020)	MACOB	Forestier & Oudeyer (2016b)
MAP-Elites	Mouret & Clune (2015)	MAX	Shyam et al. (2019)
ME-ES	Chapter 2	MUGL	Laversanne-Finot et al. (2018)
MULEX	Beyer et al. (2019)	NGU	Badia et al. (2020b)
NS	Lehman & Stanley (2008)	NS-ES	Conti et al. (2018)
NS-LC	Lehman & Stanley (2011b)	NSR-ES	Conti et al. (2018)
Q-Learning	Watkins (1989)	ReNN	Li et al. (2020)
RIDE	Raileanu & Rocktäschel (2020)	RIG	Nair et al. (2018b)
RND	Burda et al. (2019)	SAGG-RIAC	Baranes & Oudeyer (2013)
SAC	Haarnoja et al. (2018b)	Scalable ES	Salimans et al. (2017)
Schmidhuber	Schmidhuber (1991a)	Setter-Solver	Racanière et al. (2020)
SGIM	Nguyen et al. (2011)	SGIM-ACTS	Nguyen & Oudeyer (2012)
SGIM-D	Nguyen & Oudeyer (2014)	Skew-Fit	Pong et al. (2020)
TextWorld	Côté et al. (2018)	UGL	Péré et al. (2018)
UVFA	Schaul et al. (2015)	VIME	Houthooft et al. (2016a)

Bibliography

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In Brodley, C. E. (ed.), *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

Abramson, J., Ahuja, A., Brussee, A., Carnevale, F., Cassin, M., Clark, S., Dudzik, A., Georgiev, P., Guy, A., Harley, T., Hill, F., Hung, A., Kenton, Z., Landon, J., Lillicrap, T. P., Mathewson, K., Muldal, A., Santoro, A., Savinov, N., Varma, V., Wayne, G., Wong, N., Yan, C., and Zhu, R. Imitating Interactive Intelligence. *CoRR*, abs/2012.05672, 2020.

Achiam, J. and Sastry, S. Surprise-Based Intrinsic Motivation for Deep Reinforcement Learning. *CoRR*, abs/1703.01732, 2017.

Achiam, J., Edwards, H., Amodei, D., and Abbeel, P. Variational Option Discovery Algorithms. *CoRR*, abs/1807.10299, 2018.

Akakzia, A., Colas, C., Oudeyer, P.-Y., Chetouani, M., and Sigaud, O. Grounding Language to Autonomously-Acquired Skills Via Goal Generation. In *ICLR 2021*, 2021.

Al-Omari, M., Duckworth, P., Hogg, D. C., and Cohn, A. G. Natural language acquisition and grounding for embodied robotic systems. In Singh, S. P. and Markovitch, S. (eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 4349–4356. AAAI Press, 2017.

Alet, F., Schneider, M. F., Lozano-Pérez, T., and Kaelbling, L. P. Meta-learning curiosity algorithms. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Ammanabrolu, P. and Hausknecht, M. J. Graph constrained reinforcement learning for natural language action spaces. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Andreas, J. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7556–7566, Online, 2020. Association for Computational Linguistics.

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 39–48. IEEE Computer Society, 2016.

Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. 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. 166–175. PMLR, 2017.

Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. 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. 5048–5058, 2017.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. VQA: visual question answering. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 2425–2433. IEEE Computer Society, 2015.

Arkin, R. C., Arkin, R. C., and others. *Behavior-Based Robotics*. MIT press, 1998.

Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M., and Yoshida, C. Cognitive Developmental Robotics: A Survey. *IEEE transactions on autonomous mental development*, 1(1):12–34, 2009.

Aubret, A., Matignon, L., and Hassas, S. A Survey on Intrinsic Motivation in Reinforcement Learning. *CoRR*, abs/1908.06976, 2019.

Back, T., Hoffmeister, F., and Schwefel, H.-P. A Survey of Evolution Strategies. In *Proceedings of the fourth international conference on genetic algorithms*, volume 2. Morgan Kaufmann Publishers San Mateo, CA, 1991.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., and Blundell, C. Agent57: Outperforming the atari human benchmark. 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. 507–517. PMLR, 2020a.

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., and Blundell, C. Never give up: Learning directed exploration strategies. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b.

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, 2019a.

Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., de Vries, H., and Courville, A. C. Systematic generalization: What is required and can it be learned? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b.

Bain, M. and Sammut, C. A Framework for Behavioural Cloning. In *Machine Intelligence 15*, pp. 103–129, 1995.

Baldassarre, G. What Are Intrinsic Motivations? A Biological Perspective. In *2011 IEEE international conference on development and learning (ICDL)*, volume 2, pp. 1–8. IEEE, 2011.

Baldassarre, G. Intrinsic Motivations and Open-Ended Development in Animals, Humans, and Robots: An Overview. *Frontiers in Psychology*, pp. 5, 2014.

Baldassarre, G. and Mirolli, M. *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer, 2013.

Baranes, A. and Oudeyer, P.-Y. Proximo-Distal Competence Based Curiosity-Driven Exploration. In *Learning, in" International Conference on Epigenetic Robotics, Italie*. Citeseer, 2009a.

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, 2009b.

Baranes, A. and Oudeyer, P.-Y. Intrinsically Motivated Goal Exploration for Active Motor Learning in Robots: A Case Study. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, 2010.

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.

Barsalou, L. W. Grounded Cognition. *Annu. Rev. Psychol.*, 59:617–645, 2008.

Barto, A., Mirolli, M., and Baldassarre, G. Novelty or Surprise? *Frontiers in psychology*, 4:907, 2013.

Barto, A. G. Intrinsic Motivation and Reinforcement Learning. In *Intrinsically motivated learning in natural and artificial systems*, pp. 17–47. Springer, 2013.

Barto, A. G. and Mahadevan, S. Recent Advances in Hierarchical Reinforcement Learning. *Discrete event dynamic systems*, 13(1):41–77, 2003.

Barto, A. G. and Simsek, O. Intrinsic Motivation for Reinforcement Learning Systems. In *Proceedings of the Thirteenth Yale Workshop on Adaptive and Learning Systems*, pp. 113–118. Citeseer, 2005.

Barto, A. G., Singh, S., and Chentanez, N. Intrinsically Motivated Learning of Hierarchical Collections of Skills. In *Proceedings of the 3rd International Conference on Development and Learning*, pp. 112–19. Piscataway, NJ, 2004.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V. F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., G\"{u}l\c{c}ehre, \., Song, H. F., Ballard, A. J., Gilmer, J., Dahl, G. E., Vaswani, A., Allen, K. R., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. Relational Inductive Biases, Deep Learning, and Graph Networks. *CoRR*, abs/1806.01261, 2018.

Beaulieu, S., Frati, L., Miconi, T., Lehman, J., Stanley, K. O., Clune, J., and Cheney, N. Learning to Continually Learn. In Giacomo, G. D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., and Lang, J. (eds.), *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 992–1001. IOS Press, 2020.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

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.

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.

Benureau, F. C. and Oudeyer, P.-Y. Behavioral Diversity Generation in Autonomous Exploration Through Reuse of Past Experience. *Frontiers in Robotics and AI*, 3:8, 2016.

Bergen, B. K. *Louder Than Words: The New Science of How the Mind Makes Meaning*. Basic Books (AZ), 2012.

Bergs, A. What, If Anything, Is Linguistic Creativity? *Gestalt Theory*, 41(2):173–183, 2019.

Berk, L. E. Why Children Talk to Themselves. *Scientific American*, 271(5):78–83, 1994.

Berlyne, D. E. Novelty and Curiosity as Determinants of Exploratory Behaviour. *British Journal of Psychology*, 41(1):68, 1950.

Berlyne, D. E. Curiosity and Exploration. *Science*, 153(3731):25–33, 1966.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H. P. d. O., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. Dota 2 with Large Scale Deep Reinforcement Learning. *CoRR*, abs/1912.06680, 2019.

Berseth, G., Geng, D., Devin, C., Finn, C., Jayaraman, D., and Levine, S. SMiRL: Surprise Minimizing RL in Dynamic Environments. *CoRR*, abs/1912.05510, 2019.

Beyer, L., Vincent, D., Teboul, O., Gelly, S., Geist, M., and Pietquin, O. MULEX: Disentangling Exploitation from Exploration in Deep RL. *CoRR*, abs/1907.00868, 2019.

Bisk, Y., Yuret, D., and Marcu, D. Natural language communication with robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 751–761, San Diego, California, 2016. Association for Computational Linguistics.

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, 2020. Association for Computational Linguistics.

Blaes, S., Pogancic, M. V., Zhu, J., and Martius, G. Control what you can: Intrinsically motivated task-planning agent. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 12520–12531, 2019.

Bonawitz, E., Shafto, P., Gweon, H., Chang, I., Katz, S., and Schulz, L. The Double-Edged Sword of Pedagogy: Modeling the Effect of Pedagogical Contexts on Preschoolers' Exploratory Play. pp. 6, 2009.

Bonferroni, C. Teoria Statistica Delle Classi E Calcolo Delle Probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62, 1936.

Bornstein, M. H., Tamis-LeMonda, C. S., Tal, J., Ludemann, P., Toda, S., Rahn, C. W., Pêcheux, M.-G., Azuma, H., and Vardi, D. Maternal Responsiveness to Infants in Three Societies: The United States, France, and Japan. *Child development*, 63(4):808–821, 1992.

Bougie, N. and Ichise, R. Fast and Slow Curiosity for High-Level Exploration in Reinforcement Learning. *Applied Intelligence*, 51(2):1086–1107, 2021.

Braitenberg, V. *Vehicles: Experiments in Synthetic Psychology*. MIT press, 1986.

Branavan, S., Zettlemoyer, L., and Barzilay, R. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1268–1277, Uppsala, Sweden, 2010. Association for Computational Linguistics.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *CoRR*, abs/1606.01540, 2016.

Brooks, R. A. Elephants Don't Play Chess. *Robotics and autonomous systems*, 6(1-2): 3–15, 1990.

Brooks, R. A. Intelligence Without Representation. *Artificial intelligence*, 47(1-3):139–159, 1991a.

Brooks, R. A. New Approaches to Robotics. *Science*, 253(5025):1227–1232, 1991b.

Brooks, R. A., Breazeal, C., Marjanović, M., Scassellati, B., and Williamson, M. M. The Cog Project: Building a Humanoid Robot. In Goos, G., Hartmanis, J., van Leeuwen, J., and Nehaniv, C. L. (eds.), *Computation for Metaphors, Analogy, and Agents*, volume 1562, pp. 52–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

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., 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. 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.

Bruner, J. Child's Talk: Learning to Use Language. *Child Language Teaching and Therapy*, 1(1):111–114, 1985.

Bruner, J. The Narrative Construction of Reality. *Critical Inquiry*, 18(1):1–21, 1991.

Burda, Y., Edwards, H., Storkey, A. J., and Klimov, O. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. MONet: Unsupervised Scene Decomposition and Representation. *CoRR*, abs/1901.11390, 2019.

Campero, A., Raileanu, R., Kuttler, H., Tenenbaum, J. B., Rocktäschel, T., and Grefenstette, E. Learning with AMIGo: Adversarially Motivated Intrinsic Goals. In *International Conference on Learning Representations*, 2021.

Campos, V., Trott, A., Xiong, C., Socher, R., Giró-i-Nieto, X., and Torres, J. Explore, discover and learn: Unsupervised discovery of state-covering skills. 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. 1317–1327. PMLR, 2020.

Cangelosi, A. and Schlesinger, M. *Developmental Robotics: From Babies to Robots*. MIT press, 2015.

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. *IEEE Transactions on Autonomous Mental Development*, 2(3):167–195, 2010.

Carruthers, P. Modularity, Language, and the Flexibility of Thought. *Behavioral and Brain Sciences*, 25(6):705–719, 2002.

Caruana, R. Multitask Learning. *Machine learning*, 28(1):41–75, 1997.

Chan, B. W.-C. Lenia: Biology of Artificial Life. *Complex Syst.*, 28(3), 2019.

Chan, H., Wu, Y., Kiros, J., Fidler, S., and Ba, J. ACTRCE: Augmenting Experience via Teacher's Advice For Multi-Goal Reinforcement Learning. *CoRR*, abs/1902.04546, 2019.

Chaplot, D. S., Sathyendra, K. M., Pasumarthi, R. K., Rajagopal, D., and Salakhutdinov, R. Gated-attention architectures for task-oriented language grounding. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 2819–2826. AAAI Press, 2018.

Charlesworth, H. and Montana, G. Plangan: Model-based planning with sparse rewards and multiple goals. 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.

Chatzilygeroudis, K., Vassiliades, V., and Mouret, J.-B. Reset-Free Trial-and-Error Learning for Robot Damage Recovery. *Robotics and Autonomous Systems*, 100:236–250, 2018.

Chen, D. L. and Mooney, R. J. Learning to interpret natural language navigation instructions from observations. In Burgard, W. and Roth, D. (eds.), *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011.

Chen, V., Gupta, A., and Marino, K. Ask Your Humans: Using Human Instructions to Improve Generalization in Reinforcement Learning. In *International Conference on Learning Representations*, 2021.

Chen, Z. Object-Based Attention: A Tutorial Review. *Attention, Perception, & Psychophysics*, 74(5):784–802, 2012.

Chenu, A., Perrin-Gilbert, N., Doncieux, S., and Sigaud, O. Selection-Expansion: A Unifying Framework for Motion-Planning and Diversity Search Algorithms. *CoRR*, abs/2104.04768, 2021.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. Baby-Ai: First Steps Towards Grounded Language Learning with a Human in the Loop. In *International Conference on Learning Representations*, 2019.

Chiel, H. J. and Beer, R. D. The Brain Has a Body: Adaptive Behavior Emerges from Interactions of Nervous System, Body and Environment. *Trends in Neurosciences*, 20 (12):553–557, 1997.

Chitnis, R., Silver, T., Tenenbaum, J. B., Kaelbling, L. P., and Lozano-Pérez, T. GLIB: Exploration via Goal-Literal Babbling for Lifted Operator Learning. *CoRR*, abs/2001.08299, 2020.

Chomsky, N. *Syntactic structures*. Mouton, 1957.

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.

Chu, J. and Schulz, L. Exploratory Play, Rational Action, and Efficient Search. In Denison, S., Mack, M., Xu, Y., and Armstrong, B. C. (eds.), *Proceedings of the 42th Annual Meeting of the Cognitive Science Society - Developing a Mind: Learning in Humans, Animals, and Machines, CogSci 2020, virtual, July 29 - August 1, 2020*. cognitivesciencesociety.org, 2020a.

Chu, J. and Schulz, L. E. Play, Curiosity, and Cognition. *Annual Review of Developmental Psychology*, 2(1):317–343, 2020b.

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.

Chulef, A. S., Read, S. J., and Walsh, D. A. A Hierarchical Taxonomy of Human Goals. *Motivation and Emotion*, pp. 42, 2001.

Cideron, G., Pierrot, T., Perrin, N., Beguir, K., and Sigaud, O. QD-RL: Efficient Mixing of Quality and Diversity in Reinforcement Learning. *CoRR*, abs/2006.08505, 2020a.

Cideron, G., Seurin, M., Strub, F., and Pietquin, O. Higher: Improving Instruction Following with Hindsight Generation for Experience Replay. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 225–232, 2020b.

Clark, A. Magic Words: How Language Augments Human Computation. In Carruthers, P. and Boucher, J. (eds.), *Language and Thought*, pp. 162–183. Cambridge University Press, 1 edition, 1998a.

Clark, A. *Being There: Putting Brain, Body, and World Together Again*. MIT press, 1998b.

Clark, A. Language, Embodiment, and the Cognitive Niche. *Trends in Cognitive Sciences*, 10(8):370–374, 2006.

Clark, A. and Grush, R. Towards a Cognitive Robotics. *Adaptive Behavior*, 7(1):5–16, 1999.

Clark, A., Davies, M., Dennett, D. C., Frankish, K., Goldin-Meadow, S., Gomez, J.-C., Laurence, S., and others. *Language and Thought: Interdisciplinary Themes*. Cambridge University Press, 1998.

Co-Reyes, J. D., Gupta, A., Sanjeev, S., Altieri, N., Andreas, J., DeNero, J., Abbeel, P., and Levine, S. Guiding policies with language via meta-learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. End-to-End Driving Via Conditional Imitation Learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9. IEEE, 2018.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. Active Learning with Statistical Models. *Journal of artificial intelligence research*, 4:129–145, 1996.

Colas, C., Sigaud, O., and Oudeyer, P. GEP-PG: decoupling exploration and exploitation in deep reinforcement learning algorithms. 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. 1038–1047. PMLR, 2018a.

Colas, C., Sigaud, O., and Oudeyer, P.-Y. How Many Random Seeds? Statistical Power Analysis in Deep Reinforcement Learning Experiments. *CoRR*, abs/1806.08295, 2018b.

Colas, C., Oudeyer, P., Sigaud, O., Fournier, P., and Chetouani, M. CURIOUS: intrinsically motivated modular multi-goal reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1331–1340. PMLR, 2019a.

Colas, C., Sigaud, O., and Oudeyer, P.-Y. A Hitchhiker's Guide to Statistical Comparisons of Reinforcement Learning Algorithms. *CoRR*, abs/1904.06979, 2019b.

Colas, C., Hejblum, B. P., Rouillon, S., Thiébaut, R., Oudeyer, P.-Y., Moulin-Frier, C., and Prague, M. EpidemiOptim: A Toolbox for the Optimization of Control Policies in Epidemiological Models. *CoRR*, abs/2010.04452, 2020a.

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*, 2020b.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y. Intrinsically Motivated Goal-Conditioned Reinforcement Learning: a Short Survey. *CoRR*, abs/2012.09830, 2020c.

Colas, C., Madhavan, V., Huizinga, J., and Clune, J. Scaling Map-Elites to Deep Neuroevolution. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 67–75, 2020d.

Colas, C., Karch, T., Moulin-Frier, C., and Oudeyer, P.-Y. Language as a Cognitive Tool: Dall-E, Humans and Vygotskian RL Agents. *Blog post - Flowers Team Inria*, 2021.

Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K. O., and Clune, J. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking 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. 5032–5043, 2018.

Copeland, B. J. and Proudfoot, D. Artificial Intelligence: History, Foundations, and Philosophical Issues. *Artificial Intelligence*, pp. 54, 2007.

Cordeschi, R. *The Discovery of the Artificial: Behavior, Mind and Machines Before and Beyond Cybernetics*, volume 28. Springer Science & Business Media, 2002.

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

Cully, A. Autonomous Skill Discovery with Quality-Diversity and Unsupervised Descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 81–89, 2019.

Cully, A. and Demiris, Y. Quality and Diversity Optimization: A Unifying Modular Framework. *IEEE Transactions on Evolutionary Computation*, 2017.

Cully, A. and Mouret, J.-B. Evolving a Behavioral Repertoire for a Walking Robot. *Evol. Comput.*, 24(1):59–88, 2016.

Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. Robots That Can Adapt Like Animals. *Nature*, 521(7553):503, 2015.

Cummins, R. Systematicity. *The Journal of Philosophy*, 93(12):591–614, 1996.

Czikszentmihalyi, M. *Flow: The Psychology of Optimal Experience*. New York: Harper & Row, 1990.

Côté, M.-A., K\'{a}d\'{a}r, \., Yuan, X., Kybartas, B., 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 Cazenave, T., Saffidine, A., and Sturtevant, N. (eds.), *Computer Games - 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers*, volume 1017 of *Communications in Computer and Information Science*, pp. 41–75. Springer, 2018.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. Embodied question answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 1–10. IEEE Computer Society, 2018.

Dautenhahn, K. Getting to Know Each Other—Artificial Social Intelligence for Autonomous Robots. *Robotics and Autonomous Systems*, 16(2-4):333–356, 1995.

Dautenhahn, K. and Billard, A. Studying Robot Social Cognition Within a Developmental Psychology Framework. In *1999 Third European Workshop on Advanced Mobile Robots (Eurobot'99). Proceedings (Cat. No. 99EX355)*, pp. 187–194. IEEE, 1999.

Dautenhahn, K., Ogden, B., and Quick, T. From Embodied to Socially Embedded Agents – Implications for Interaction-Aware Robots. *Cognitive Systems Research*, 3(3):397–428, 2002.

Dayan, P. and Hinton, G. E. Feudal Reinforcement Learning. In *Advances in neural information processing systems*, pp. 271–278, 1993.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The Helmholtz Machine. *Neural Computation*, 7(5):889–904, 1995.

Deisenroth, M. P., Rasmussen, C. E., and Fox, D. Learning to Control a Low-Cost Manipulator Using Data-Efficient Reinforcement Learning. *Robotics: Science and Systems VII*, pp. 57–64, 2011.

Denervaud, S., Knebel, J.-F., Hagmann, P., and Gentaz, E. Beyond Executive Functions, Creativity Skills Benefit Academic Outcomes: Insights from Montessori Education. *PloS one*, 14(11):e0225319, 2019.

Denervaud, S., Knebel, J.-F., Immordino-Yang, M. H., and Hagmann, P. Effects of Traditional Versus Montessori Schooling on 4-to 15-Year Old Children's Performance Monitoring. *Mind, Brain, and Education*, 14(2):167–175, 2020.

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, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

Ding, D., Hill, F., Santoro, A., and Botvinick, M. M. Object-Based Attention for Spatio-Temporal Reasoning: Outperforming Neuro-Symbolic Models with Flexible Distributed Architectures. *CoRR*, abs/2012.08508, 2020.

Ding, Y., Florensa, C., Abbeel, P., and Phielipp, M. Goal-conditioned imitation learning. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 15298–15309, 2019.

Doan, T., Mazoure, B., Durand, A., Pineau, J., and Hjelm, R. D. Attraction-Repulsion Actor-Critic for Continuous Control Reinforcement Learning. *CoRR*, abs/1909.07543, 2019.

Dominey, P. F. Emergence of Grammatical Constructions: Evidence from Simulation and Grounded Agent Experiments. *Connection Science*, 17(3-4):289–306, 2005.

Dénervaud, S. and Gentaz, E. Les Effets De La « Méthode Montessori » Sur Le Développement Psychologique Des Enfants: Une Synthèse Des Recherches Scientifiques Quantitatives. pp. 6, 2015.

Ebert, F., Finn, C., Dasari, S., Xie, A., Lee, A. X., and Levine, S. Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control. *CoRR*, abs/1812.00568, 2018.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. First Return, Then Explore. *Nature*, 590(7847):580–586, 2021.

Eiben, A. and Smith, J. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

Elliot, A. J. and Fryer, J. W. The Goal Construct in Psychology. *Handbook of motivation science*, 18:235–250, 2008.

Etcheverry, M., Moulin-Frier, C., and Oudeyer, P. Hierarchically organized latent modules for exploratory search in morphogenetic systems. 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.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Eysenbach, B., Geng, X., Levine, S., and Salakhutdinov, R. R. Rewriting history with inverse RL: hindsight inference for policy improvement. 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.

Fillmore, C. J. Frames and the Semantics of Understanding. *Quaderni di semantica*, 6(2): 222–254, 1985.

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.

Florensa, C., Degrave, J., Heess, N., Springenberg, J. T., and Riedmiller, M. A. Self-supervised Learning of Image Embedding for Continuous Control. *CoRR*, abs/1901.00943, 2019.

Fodor, J. A. and Pylyshyn, Z. W. Connectionism and Cognitive Architecture: A Critical Analysis. *Cognition*, 28(1-2):3–71, 1988.

Fontaine, M. C., Togelius, J., Nikolaidis, S., and Hoover, A. K. Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space. In Coello, C. A. C. (ed.), *GECCO '20: Genetic and Evolutionary Computation Conference, Cancún Mexico, July 8-12, 2020*, pp. 94–102. ACM, 2020.

Forestier, S. *Intrinsically Motivated Goal Exploration in Child Development and Artificial Intelligence: Learning and Development of Speech and Tool Use*. These de doctorat, Bordeaux, 2019. URL https://www.theses.fr/2019BORD0247.

Forestier, S. and Oudeyer, P.-Y. Curiosity-Driven Development of Tool Use Precursors: A Computational Model. In *38th Annual Conference of the Cognitive Science Society (CogSci 2016)*, pp. 1859–1864, 2016a.

Forestier, S. and Oudeyer, P.-Y. Modular Active Curiosity-Driven Discovery of Tool Use. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 3965–3972. IEEE, 2016b.

Forestier, S. and Oudeyer, P.-Y. Overlapping Waves in Tool Use Development: A Curiosity-Driven Computational Model. In *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pp. 238–245. IEEE, 2016c.

Forestier, S., Mollard, Y., and Oudeyer, P.-Y. Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning. *CoRR*, abs/1708.02190, 2017.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. Noisy networks for exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Fournier, P., Sigaud, O., Chetouani, M., and Oudeyer, P.-Y. Accuracy-based Curriculum Learning in Deep Reinforcement Learning. *CoRR*, abs/1806.09614, 2018.

Fournier, P., Colas, C., Chetouani, M., and Sigaud, O. CLIC: Curriculum Learning and Imitation for Object Control in Non-Rewarding Environments. *IEEE Transactions on Cognitive and Developmental Systems*, 2019.

Frans, K., Ho, J., Chen, X., Abbeel, P., and Schulman, J. Meta learning shared hierarchies. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adverserial inverse reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Fu, J., Korattikara, A., Levine, S., and Guadarrama, S. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. 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. 1582–1591. PMLR, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062. PMLR, 2019.

Gajewski, A., Clune, J., Stanley, K. O., and Lehman, J. Evolvability ES: Scalable and Direct Optimization of Evolvability. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 107–115, 2019.

Gentner, D. Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive Science*, 7(2):155–170, 1983.

Gentner, D. Language as Cognitive Tool Kit: How Language Supports Relational Thought. *American Psychologist*, 71(8):650–657, 2016.

Gentner, D. and Goldin-Meadow, S. (eds.). *Language in Mind: Advances in the Study of Language and Thought*. MIT Press, Cambridge, Mass, 2003.

Gentner, D. and Hoyos, C. Analogy and Abstraction. *Topics in Cognitive Science*, 9(3): 672–693, 2017.

Giddins, G. *Visions of Jazz: The First Century*. Oxford University Press, 2000.

Glenberg, A. M. and Kaschak, M. P. Grounding Language in Action. *Psychonomic Bulletin & Review*, 9(3):558–565, 2002.

Goldberg, A. E. *Constructions: A Construction Grammar Approach to Argument Structure*. Cognitive theory of language and culture. University of Chicago Press, Chicago, 1995.

Goldberg, A. E. The Emergence of the Semantics of Argument Structure Constructions. pp. 9, 1999.

Goldberg, A. E. Constructions: A New Theoretical Approach to Language. *Trends in cognitive sciences*, 7(5):219–224, 2003.

Goldberg, A. E. *Constructions at Work: The Nature of Generalization in Language*. Oxford linguistics. Oxford University Press, Oxford ; New York, 2006.

Golinkoff, R. M., Can, D. D., Soderstrom, M., and Hirsh-Pasek, K. (Baby)Talk to Me: The Social Context of Infant-Directed Speech and Its Effects on Early Language Acquisition. *Current Directions in Psychological Science*, 24(5):339–344, 2015.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative Adversarial Networks. *CoRR*, abs/1406.2661, 2014.

Gopnik, A. *The Philosophical Baby: What Children's Minds Tell Us About Truth, Love & the Meaning of Life*. Random House, 2009.

Gopnik, A., Meltzoff, A. N., and Kuhl, P. K. *The Scientist in the Crib: Minds, Brains, and How Children Learn.* William Morrow & Co, 1999.

Gottlieb, J. and Oudeyer, P.-Y. Towards a Neuroscience of Active Sampling and Curiosity. *Nature Reviews Neuroscience*, 19(12):758–770, 2018.

Gottlieb, J., Oudeyer, P.-Y., Lopes, M., and Baranes, A. Information-Seeking, Curiosity, and Attention: Computational and Neural Mechanisms. *Trends in cognitive sciences*, 17(11):585–593, 2013.

Goyal, P., Niekum, S., and Mooney, R. J. Using natural language for reward shaping in reinforcement learning. In Kraus, S. (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 2385–2391. ijcai.org, 2019.

Green, E. J. and Quilty-Dunn, J. What Is an Object File? *The British Journal for the Philosophy of Science*, 2017.

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. Multi-object representation learning with iterative variational inference. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2424–2433. PMLR, 2019.

Greff, K., Steenkiste, S. v., and Schmidhuber, J. On the Binding Problem in Artificial Neural Networks. *CoRR*, abs/2012.05208, 2020.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational Intrinsic Control. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

Grizou, J., Points, L. J., Sharma, A., and Cronin, L. Exploration of Self-Propelling Droplets Using a Curiosity Driven Robotic Assistant. *CoRR*, abs/1904.12635, 2019.

Ha, D. and Schmidhuber, J. World Models. *CoRR*, abs/1803.10122, 2018.

Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., and Levine, S. Composable Deep Reinforcement Learning for Robotic Manipulation. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pp. 6244–6251. IEEE, 2018a.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 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. 1856–1865. PMLR, 2018b.

Haber, N., Mrowca, D., Wang, S., Li, F., and Yamins, D. L. 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.

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. Mastering Atari with Discrete World Models. In *International Conference on Learning Representations*, 2021.

Hamrick, J. B., Friesen, A. L., Behbahani, F., Guez, A., Viola, F., Witherspoon, S., Anthony, T., Buesing, L., Velickovic, P., and Weber, T. On the Role of Planning in Model-Based Deep Reinforcement Learning. *CoRR*, abs/2011.04021, 2020.

Hansen, N. The CMA Evolution Strategy: A Tutorial. *CoRR*, abs/1604.00772, 2016.

Harari, Y. N. *Sapiens: A Brief History of Humankind*. Random House, 2014.

Harnad, S. The Symbol Grounding Problem. *Physica D*, 42:335–346, 1990.

Hartikainen, K., Geng, X., Haarnoja, T., and Levine, S. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 1026–1034. IEEE Computer Society, 2015.

Hebb, D. O. The Organization of Behavior; a Neuropsycholocigal Theory. *A Wiley Book in Clinical Psychology*, 62:78, 1949.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3207–3214. AAAI Press, 2018.

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W. M., Jaderberg, M., Teplyashin, D., Wainwright, M., Apps, C., Hassabis, D., and Blunsom, P. Grounded Language Learning in a Simulated 3D World. *CoRR*, abs/1706.06551, 2017.

Hermer-Vazquez, L. Language, Space, and the Development of Cognitive Flexibility in Humans: The Case of Two Spatial Memory Tasks. *Cognition*, 79(3):263–299, 2001.

Hester, T., Vecerík, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., Dulac-Arnold, G., Agapiou, J. P., Leibo, J. Z., and Gruslys, A. Deep q-learning from demonstrations. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18),*

and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pp. 3223–3230. AAAI Press, 2018.

Hill, F., Lampinen, A. K., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A. Emergent Systematic Generalization in a Situated Agent. *CoRR*, abs/1910.00571, 2019.

Hill, F., Lampinen, A. K., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A. Environmental drivers of systematicity and generalization in a situated agent. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020a.

Hill, F., Mokra, S., Wong, N., and Harley, T. Human Instruction-Following with Deep Reinforcement Learning via Transfer-Learning from Text. *CoRR*, abs/2005.09382, 2020b.

Hill, F., Tieleman, O., Glehn, T. v., Wong, N., Merzic, H., and Clark, S. Grounded Language Learning Fast and Slow. In *International Conference on Learning Representations*, 2021.

Hinaut, X. and Dominey, P. F. Real-Time Parallel Processing of Grammatical Structure in the Fronto-Striatal System: A Recurrent Network Simulation Study Using Reservoir Computing. *PloS one*, 8(2), 2013.

Ho, J. and Ermon, S. Generative adversarial imitation learning. 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. 4565–4573, 2016.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Comput.*, 9(8): 1735–1780, 1997.

Hoffmann, T. Creativity and Construction Grammar: Cognitive and Psychological Issues. *Zeitschrift für Anglistik und Amerikanistik*, 66(3):259–276, 2018.

Hoffmann, T. Construction Grammar and Creativity: Evolution, Psychology, and Cognitive Science. *Cognitive Semiotics*, 13(1), 2020.

Holland, J. H. Genetic Algorithms. *SCIENTIFIC AMERICAN*, pp. 9, 1992.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P. Curiosity-Driven Exploration in Deep Reinforcement Learning Via Bayesian Neural Networks. *CoRR*, abs/1605.09674, 2016a.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P. VIME: variational information maximizing exploration. 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. 1109–1117, 2016b.

Hu, H., Yarats, D., Gong, Q., Tian, Y., and Lewis, M. Hierarchical decision making by generating and following natural language instructions. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 10025–10034, 2019.

Hull, C. L. *Principles of Behavior*, volume 422. Appleton-century-crofts New York, 1943.

Hunt, J. Intrinsic Motivation and Its Role in Psychological Development. In *Nebraska symposium on motivation*, volume 13, pp. 189–282. University of Nebraska Press, 1965.

Hunt, J. J., Barreto, A., Lillicrap, T. P., and Heess, N. Composing entropic policies using divergence correction. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2911–2920. PMLR, 2019.

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. Reward learning from human preferences and demonstrations in atari. 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. 8022–8034, 2018.

Islam, R., Henderson, P., Gomrokchi, M., and Precup, D. Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. In *Proceedings of the ICML 2017 workshop on Reproducibility in Machine Learning (RML)*, 2017.

Janner, M., Narasimhan, K., and Barzilay, R. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics*, 6: 49–61, 2018.

Jara-Ettinger, J. Theory of mind as inverse reinforcement learning. *Current Opinion in Behavioral Sciences*, 29:105–110, 2019.

Jiang, Y., Gu, S., Murphy, K., and Finn, C. Language as an abstraction for hierarchical deep reinforcement learning. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9414–9426, 2019.

Johnson, S. P., Amso, D., and Slemmer, J. A. Development of Object Concepts in Infancy: Evidence for Early Learning in an Eye-Tracking Paradigm. *Proceedings of the National Academy of Sciences*, 100(18):10568–10573, 2003.

Jung, T., Polani, D., and Stone, P. Empowerment for Continuous Agent - Environment Systems. *Adapt. Behav.*, 19(1):16–39, 2011.

Jung, W., Park, G., and Sung, Y. Population-guided parallel policy search for reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Justesen, N., Bontrager, P., Togelius, J., and Risi, S. Deep Learning for Video Game Playing. *IEEE Trans. Games*, 12(1):1–20, 2020a.

Justesen, N., Duque, M. G., Jaramillo, D. C., Mouret, J.-B., and Risi, S. Learning a Behavioral Repertoire from Demonstrations. In *IEEE Conference on Games, CoG 2020, Osaka, Japan, August 24-27, 2020*, pp. 383–390. IEEE, 2020b.

Kaelbling, L. P. Learning to Achieve Goals. In *IJCAI*, pp. 1094–1099. Citeseer, 1993.

Kahneman, D. *Thinking, Fast and Slow*. Macmillan, 2011.

Kahneman, D. and Tversky, A. The Simulation Heuristic. Technical report, Stanford Univ Ca Dept Of Psychology, 1981.

Kaplan, F. and Oudeyer, P.-Y. Maximizing Learning Progress: An Internal Reward System for Development. In *Embodied artificial intelligence*, pp. 259–270. Springer, 2004.

Kaplan, F. and Oudeyer, P.-Y. In Search of the Neural Circuits of Intrinsic Motivation. *Frontiers in neuroscience*, 1:17, 2007.

Karch, T., Colas, C., Teodorescu, L., Moulin-Frier, C., and Oudeyer, P.-Y. Deep Sets for Generalization in RL. *CoRR*, abs/2003.09443, 2020.

Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., Tsarkov, D., Wang, X., van Zee, M., and Bousquet, O. Measuring compositional generalization: A comprehensive method on realistic data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Khadka, S., Majumdar, S., Nassar, T., Dwiel, Z., Tumer, E., Miret, S., Liu, Y., and Tumer, K. Collaborative evolutionary reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3341–3350. PMLR, 2019.

Kidd, C. and Hayden, B. Y. The Psychology and Neuroscience of Curiosity. *Neuron*, 88 (3):449–460, 2015.

Kidd, C., Piantadosi, S. T., and Aslin, R. N. The Goldilocks Effect: Human Infants Allocate Attention to Visual Sequences That Are Neither Too Simple nor Too Complex. *PloS one*, 7(5):e36399, 2012.

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.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. 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.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Kollar, T., Tellex, S., Roy, D., and Roy, N. Toward Understanding Natural Language Directions. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 259–266, 2010.

Kovač, G., Laversanne-Finot, A., and Oudeyer, P.-Y. GRIMGEP: Learning Progress for Robust Goal Sampling in Visual Deep Reinforcement Learning. *CoRR*, abs/2008.04388, 2020.

Kuhlmann, G., Stone, P., Mooney, R., and Shavlik, J. Guiding a Reinforcement Learner with Natural Language Advice: Initial Results in Robocup Soccer. In *The AAAI-2004 workshop on supervisory control of learning and adaptive systems*. San Jose, CA, 2004.

Kulick, J., Toussaint, M., Lang, T., and Lopes, M. Active learning for teaching a robot grounded relational symbols. In Rossi, F. (ed.), *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pp. 1451–1457. IJCAI/AAAI, 2013.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction 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. 3675–3683, 2016.

Lake, B. M. and Murphy, G. L. Word Meaning in Minds and Machines. *CoRR*, abs/2008.01766, 2020.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building Machines That Learn and Think Like People. *Behavioral and Brain Sciences*, 40:e253, 2017.

Lanier, J. B., McAleer, S., and Baldi, P. Curiosity-Driven Multi-Criteria Hindsight Experience Replay. *CoRR*, abs/1906.03710, 2019.

Laversanne-Finot, A., Péré, A., and Oudeyer, P.-Y. Curiosity Driven Exploration of Learned Disentangled Goal Spaces. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pp. 487–504. PMLR, 2018.

LeCun, Y., Bengio, Y., and others. Convolutional Networks for Images, Speech, and Time Series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lehman, J. and Stanley, K. O. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *ALIFE*, pp. 329–336, 2008.

Lehman, J. and Stanley, K. O. Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary computation*, 19(2):189–223, 2011a.

Lehman, J. and Stanley, K. O. Evolving a Diversity of Virtual Creatures Through Novelty Search and Local Competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 211–218, 2011b.

Lehman, J., Risi, S., and Clune, J. Creative Generation of 3D Objects with Deep Learning and Innovation Engines. In *Proceedings of the 7th International Conference on Computational Creativity*, 2016.

Lehman, J., Chen, J., Clune, J., and Stanley, K. O. Es Is More Than Just a Traditional Finite-Difference Approximator. In Aguirre, H. E. and Takadama, K. (eds.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, pp. 450–457. ACM, 2018a.

Lehman, J., Chen, J., Clune, J., and Stanley, K. O. Safe Mutations for Deep and Recurrent Neural Networks Through Output Gradients. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 117–124, Kyoto Japan, 2018b. ACM.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *CoRR*, abs/2005.01643, 2020.

Levy, A., Konidaris, G. D., Jr., R. P., and Saenko, K. Learning multi-level hierarchies with hindsight. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

León, B. G., Shanahan, M., and Belardinelli, F. Systematic Generalisation Through Task Temporal Logic and Deep Reinforcement Learning. *CoRR*, abs/2006.08767, 2020.

Li, R., Jabri, A., Darrell, T., and Agrawal, P. Towards Practical Multi-Object Manipulation using Relational Reinforcement Learning. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pp. 4051–4058. IEEE, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. 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.

Lin, L.-J. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Machine learning*, 8(3-4):293–321, 1992.

Lindblom, J. and Ziemke, T. Social Situatedness of Natural and Artificial Intelligence: Vygotsky and Beyond. *Adaptive Behavior*, 11(2):79–96, 2003.

Linke, C., Ady, N. M., White, M., Degris, T., and White, A. Adapting Behaviour via Intrinsic Reward: A Survey and Empirical Study. *CoRR*, abs/1906.07865, 2019.

Loewenstein, G. The Psychology of Curiosity: A Review and Reinterpretation. *Psychological Bulletin*, 116, 1994.

Lonini, L., Forestier, S., Teulière, C., Zhao, Y., Shi, B. E., and Triesch, J. Robust Active Binocular Vision Through Intrinsically Motivated Learning. *Frontiers in neurorobotics*, 7:20, 2013.

Lopes, M. and Oudeyer, P.-Y. The Strategic Student Approach for Life-Long Exploration and Learning. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pp. 1–8. IEEE, 2012.

Lopes, M., Lang, T., Toussaint, M., and Oudeyer, P. Exploration in model-based reinforcement learning by empirically estimating learning progress. In Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pp. 206–214, 2012.

Loynd, R., Fernandez, R., Çelikyilmaz, A., Swaminathan, A., and Hausknecht, M. J. Working memory graphs. 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. 6404–6414. PMLR, 2020.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J. N., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. A survey of reinforcement learning informed by natural language. In Kraus, S. (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 6309–6317. ijcai.org, 2019.

Lupyan, G. What Do Words Do? Toward a Theory of Language-Augmented Thought. In *Psychology of Learning and Motivation*, volume 57, pp. 255–297. Elsevier, 2012.

Lynch, C. and Sermanet, P. Grounding Language in Play. *CoRR*, abs/2005.07648, 2020.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning Latent Plans from Play. In Kaelbling, L. P., Kragic, D., and Sugiura, K. (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132. PMLR, 2020.

MacGlashan, J., Babes-Vroman, M., desJardins, M., Littman, M., Muresan, S., Squire, S., Tellex, S., Arumugam, D., and Yang, L. Grounding English Commands to Reward Functions. In *Robotics: Science and Systems XI*. Robotics: Science and Systems Foundation, 2015.

MacMahon, M., Stankiewicz, B., and Kuipers, B. Walk the Talk: Connecting Language, Knowledge, and Action in Route Instructions. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pp. 1475–1482. AAAI Press, 2006.

Madden, C., Hoen, M., and Dominey, P. F. A Cognitive Neuroscience Perspective on Embodied Language for Human–Robot Cooperation. *Brain and Language*, 112(3): 180–188, 2010.

Madotto, A., Namazifar, M., Huizinga, J., Molino, P., Ecoffet, A., Zheng, H., Papangelis, A., Yu, D., Khatri, C., and Tür, G. Exploration based language learning for text-based games. In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pp. 1488–1494. ijcai.org, 2020.

Maheswaranathan, N., Metz, L., Tucker, G., Choi, D., and Sohl-Dickstein, J. Guided evolutionary strategies: augmenting random search with surrogate gradients. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4264–4273. PMLR, 2019.

Majumdar, S., Khadka, S., Miret, S., McAleer, S., and Tumer, K. Evolutionary reinforcement learning for sample-efficient multiagent coordination. 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. 6651–6660. PMLR, 2020.

Mandler, J. M. Preverbal Representation and Language. *Language and space*, pp. 365, 1999.

Mandler, J. M. Thought Before Language. *Trends in Cognitive Sciences*, 8(11):508–513, 2004.

Mandler, J. M. On the Spatial Foundations of the Conceptual System and Its Enrichment. *Cognitive science*, 36(3):421–451, 2012.

Mankowitz, D. J., Zídek, A., Barreto, A., Horgan, D., Hessel, M., Quan, J., Oh, J., Hasselt, H. v., Silver, D., and Schaul, T. Unicorn: Continual Learning with a Universal, Off-Policy Agent. *CoRR*, abs/1802.08294, 2018.

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. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Marcus, G. F. Rethinking Eliminative Connectionism. *Cognitive psychology*, 37(3):243–282, 1998.

Martius, G., Der, R., and Ay, N. Information Driven Self-Organization of Complex Robotic Behaviors. *PloS one*, 8(5):e63400, 2013.

Marzoev, A., Madden, S., Kaashoek, M. F., Cafarella, M. J., and Andreas, J. Unnatural Language Processing: Bridging the Gap Between Synthetic and Natural Language Data. *CoRR*, abs/2004.13645, 2020.

Matheron, G., Perrin, N., and Sigaud, O. PBCS: Efficient Exploration and Exploitation Using a Synergy Between Reinforcement Learning and Motion Planning. In Farkas, I., Masulli, P., and Wermter, S. (eds.), *Artificial Neural Networks and Machine Learning - ICANN 2020 - 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15-18, 2020, Proceedings, Part II*, volume 12397 of *Lecture Notes in Computer Science*, pp. 295–307. Springer, 2020.

McClelland, J. L., Hill, F., Rudolph, M., Baldridge, J., and Schütze, H. Extending Machine Language Models toward Human-Level Language Understanding. *arXiv preprint arXiv:1912.05877*, 2019.

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.

McGovern, A. and Barto, A. G. Automatic discovery of subgoals in reinforcement learning using diverse density. In Brodley, C. E. and Danyluk, A. P. (eds.), *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pp. 361–368. Morgan Kaufmann, 2001.

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, 2016.

Meyer, J.-A. Artificial Life and the Animat Approach to Artificial Intelligence. In *Artificial Intelligence*, pp. 325–354. Elsevier, 1996.

Miessner, B. F. *Radiodynamics: The Wireless Control of Torpedoes and Other Mechanisms*. D. Van Nostrand Company, 1916.

Mikolov, T., Joulin, A., and Baroni, M. A Roadmap Towards Machine Intelligence. In Gelbukh, A. F. (ed.), *Computational Linguistics and Intelligent Text Processing - 17th International Conference, CICLing 2016, Konya, Turkey, April 3-9, 2016, Revised Selected Papers, Part I*, volume 9623 of *Lecture Notes in Computer Science*, pp. 29–61. Springer, 2016.

Minervini, P., Bosnjak, M., Rocktäschel, T., and Riedel, S. Towards Neural Theorem Proving at Scale. *CoRR*, abs/1807.08204, 2018.

Mintz, T. H. Frequent Frames as a Cue for Grammatical Categories in Child Directed Speech. *Cognition*, 90(1):91–117, 2003.

Mirolli, M. and Parisi, D. Towards a Vygotskyan Cognitive Robotics: The Role of Language as a Cognitive Tool. *New Ideas in Psychology*, 29(3):298–311, 2011.

Misra, D., Langford, J., and Artzi, Y. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1004–1015, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

Mitchell, T. R. Motivation: New Directions for Theory, Research, and Practice. *Academy of management review*, 7(1):80–88, 1982.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., and others. Human-Level Control Through Deep Reinforcement Learning. *nature*, 518(7540):529–533, 2015.

Moerland, T. M., Broekens, J., and Jonker, C. M. Model-based Reinforcement Learning: A Survey. *CoRR*, abs/2006.16712, 2020.

Mohamed, S. and Rezende, D. J. Variational information maximisation for intrinsically motivated reinforcement learning. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2125–2133, 2015.

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*, 4:1006, 2014.

Mouret, J.-B. and Clune, J. Illuminating Search Spaces by Mapping Elites. *arXiv preprint arXiv:1504.04909*, 2015.

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, 2018.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Overcoming Exploration in Reinforcement Learning with Demonstrations. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pp. 6292–6299. IEEE, 2018a.

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: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 9209–9220, 2018b.

Nair, A., Bahl, S., Khazatsky, A., Pong, V., Berseth, G., and Levine, S. Contextual Imagined Goals for Self-Supervised Robotic Learning. In *Conference on Robot Learning*, pp. 530–539, 2020.

Narasimhan, K., Kulkarni, T., and Barzilay, R. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1–11, Lisbon, Portugal, 2015. Association for Computational Linguistics.

Nasiriany, S., Pong, V., Lin, S., and Levine, S. Planning with goal-conditioned policies. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14814–14825, 2019.

Newell, A. and Simon, H. The Logic Theory Machine–a Complex Information Processing System. *IRE Transactions on information theory*, 2(3):61–79, 1956.

Newell, A. and Simon, H. A. Computer Science as Empirical Inquiry: Symbols and Search. In *ACM Turing award lectures*, pp. 1975. 1976.

Newell, A., Shaw, J. C., and Simon, H. A. Report on a General Problem Solving Program. In *IFIP congress*, volume 256, pp. 64. Pittsburgh, PA, 1959.

Nguyen, A. M., Yosinski, J., and Clune, J. Innovation Engines: Automated Creativity and Improved Stochastic Optimization Via Deep Learning. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 959–966, 2015.

Nguyen, K., Misra, D., Schapire, R. E., Dudík, M., and Shafto, P. Interactive Learning from Activity Description. *CoRR*, abs/2102.07024, 2021.

Nguyen, M. and Oudeyer, P.-Y. Socially Guided Intrinsic Motivation for Robot Learning of Motor Skills. *Autonomous Robots*, 36(3):273–294, 2014.

Nguyen, S. M. and Oudeyer, P.-Y. Active Choice of Teachers, Learning Strategies and Goals for a Socially Guided Intrinsic Motivation Learner. *Paladyn*, 3(3):136–146, 2012.

Nguyen, S. M., Baranes, A., and Oudeyer, P.-Y. Bootstrapping Intrinsically Motivated Learning with Human Demonstrations. *CoRR*, abs/1112.1937, 2011.

Nguyen-Tuong, D. and Peters, J. Model Learning for Robot Control: A Survey. *Cognitive processing*, 12(4):319–340, 2011.

Nieder, A. Prefrontal Cortex and the Evolution of Symbolic Reference. *Current Opinion in Neurobiology*, 19(1):99–108, 2009.

Oh, J., Singh, S. P., 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.

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.

Osband, I., Blundell, C., Pritzel, A., and Roy, B. V. Deep exploration via bootstrapped DQN. 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. 4026–4034, 2016.

Oudeyer, P.-Y. and Kaplan, F. What Is Intrinsic Motivation? A Typology of Computational Approaches. *Frontiers in neurorobotics*, 1:6, 2009.

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.

Parisi, D. Robots with Language. *Frontiers in Neurorobotics*, 2010.

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.

Parr, R. and Russell, S. Reinforcement Learning with Hierarchies of Machines. *Advances in neural information processing systems*, pp. 1043–1049, 1998.

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, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2778–2787. PMLR, 2017.

Pathak, D., Gandhi, D., and Gupta, A. Self-supervised exploration via disagreement. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5062–5071. PMLR, 2019.

Peng, X. B., Chang, M., Zhang, G., Abbeel, P., and Levine, S. MCP: learning composable hierarchical control with multiplicative compositional policies. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3681–3692, 2019.

Péré, A., Forestier, S., Sigaud, O., and Oudeyer, P. Unsupervised learning of goal spaces for intrinsically motivated goal exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. Film: Visual reasoning with a general conditioning layer. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3942–3951. AAAI Press, 2018.

Pfeifer, R. and Bongard, J. *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT press, 2006.

Piaget, J. *The Language and Thought of the Child*. Routledge, 1926.

Piaget, J. *The Development of Thought: Equilibration of Cognitive Structures*. Viking, 1977.

Piaget, J. and Cook, M. *The Origins of Intelligence in Children*, volume 8. International Universities Press New York, 1952.

Pierrot, T., Perrin, N., Behbahani, F., Laterre, A., Sigaud, O., Beguir, K., and Freitas, N. d. Learning Compositional Neural Programs for Continuous Control. *CoRR*, abs/2007.13363, 2020.

Pitis, S., Chan, H., Zhao, S., Stadie, B. C., and Ba, J. Maximum entropy gain exploration for long horizon multi-goal 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. 7750–7761. PMLR, 2020.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., and Zaremba, W. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. *CoRR*, abs/1802.09464, 2018a.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter space noise for exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018b.

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.

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., 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, 2020b.

Pourchot, A. and Sigaud, O. CEM-RL: combining evolutionary and gradient-based methods for policy search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Precup, D. *Temporal Abstraction in Reinforcement Learning*. PhD Thesis, The University of Massachussetts, 2000.

Pugh, J. K., Soros, L., Szerlip, P. A., and Stanley, K. O. Confronting the Challenge of Quality Diversity. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 967–974. ACM, 2015.

Pugh, J. K., Soros, L. B., and Stanley, K. O. Searching for Quality Diversity When Diversity Is Unaligned with Quality. In *International Conference on Parallel Problem Solving from Nature*, pp. 880–889. Springer, 2016.

Qian, H. and Yu, Y. Derivative-Free Reinforcement Learning: A Review. *CoRR*, abs/2102.05710, 2021.

Rabinowitz, N. C., Perbet, F., Song, H. F., Zhang, C., Eslami, S. M. A., and Botvinick, M. Machine theory of mind. 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. 4215–4224. PMLR, 2018.

Racanière, S., Lampinen, A. K., Santoro, A., Reichert, D. P., Firoiu, V., and Lillicrap, T. P. Automated curriculum generation through setter-solver interactions. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language Models Are Unsupervised Multitask Learners. *OpenAI Blog*, 1(8):9, 2019.

Raffin, A. and Stulp, F. Generalized State-Dependent Exploration for Deep Reinforcement Learning in Robotics. *CoRR*, abs/2005.05719, 2020.

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.

Ram, A., Leake, D. B., and Leake, D. *Goal-Driven Learning*. MIT press, 1995.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-Shot Text-to-Image Generation. *CoRR*, abs/2102.12092, 2021.

Rechenberg, I. Evolutionsstrategie—Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Information. *Stuttgart-Bad Cannstatt: Friedrich Frommann Verlag*, 1973.

Reinke, C., Etcheverry, M., and Oudeyer, P.-Y. Intrinsically Motivated Exploration for Automated Discovery of Patterns in Morphogenetic Systems. *CoRR*, abs/1908.06663, 2019.

Riedmiller, M. A., Hafner, R., Lampe, T., Neunert, M., Degrave, J., de Wiele, T. V., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing solving sparse reward tasks from scratch. 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. 4341–4350. PMLR, 2018.

Risi, S. and Togelius, J. Procedural Content Generation: From Automatically Generating Game Levels to Increasing Generality in Machine Learning. *CoRR*, abs/1911.13071, 2019.

Rohlfing, K. J., Wrede, B., Vollmer, A.-L., and Oudeyer, P.-Y. An Alternative to Mapping a Word onto a Concept in Language Acquisition: Pragmatic Frames. *Frontiers in Psychology*, 7, 2016.

Rolf, M. and Steil, J. J. Efficient Exploratory Learning of Inverse Kinematics on a Bionic Elephant Trunk. *IEEE transactions on neural networks and learning systems*, 25(6): 1147–1160, 2013.

Rolf, M., Steil, J. J., and Gienger, M. Goal Babbling Permits Direct Learning of Inverse Kinematics. *IEEE Transactions on Autonomous Mental Development*, 2(3):216–229, 2010.

Romac, C., Portelas, R., Hofmann, K., and Oudeyer, P.-Y. TeachMyAgent: a Benchmark for Automatic Curriculum Learning in Deep RL. *CoRR*, abs/2103.09815, 2021.

Rosch, E., Thompson, L., and Varela, F. J. *The Embodied Mind: Cognitive Science and Human Experience*. MIT Press, 1991.

Rosenblatt, F. *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

Rosenblatt, F. Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

Ruis, L., Andreas, J., Baroni, M., Bouchacourt, D., and Lake, B. M. A Benchmark for Systematic Generalization in Grounded Language Understanding. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.-F., and Lin, H.-T. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Rumelhart, D. E., Smolensky, P., McClelland, J. L., and Hinton, G. Sequential Thought Processes in Pdp Models. *Parallel distributed processing: explorations in the microstructures of cognition*, 2:3–57, 1986.

Runco, M. A. and Jaeger, G. J. The Standard Definition of Creativity. *Creativity Research Journal*, 24(1):92–96, 2012.

Ryan, R. M. and Deci, E. L. Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology*, 25(1):54–67, 2000a.

Ryan, R. M. and Deci, E. L. Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being. *American Psychologist*, pp. 67, 2000b.

Röder, F., Eppe, M., Nguyen, P. D. H., and Wermter, S. Curious Hierarchical Actor-Critic Reinforcement Learning. In Farkas, I., Masulli, P., and Wermter, S. (eds.), *Artificial Neural Networks and Machine Learning - ICANN 2020 - 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15-18, 2020, Proceedings, Part II*, volume 12397 of *Lecture Notes in Computer Science*, pp. 408–419. Springer, 2020.

Salakhutdinov, R. Deep learning. In Macskassy, S. A., Perlich, C., Leskovec, J., Wang, W., and Ghani, R. (eds.), *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pp. 1973. ACM, 2014.

Salge, C., Glackin, C., and Polani, D. Empowerment–An Introduction. In Prokopenko, M. (ed.), *Guided Self-Organization: Inception*, pp. 67–114. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

Salimans, T. and Chen, R. Learning Montezuma's Revenge from a Single Demonstration. *NeurIPS*, 2018.

Salimans, T., Ho, J., Chen, X., and Sutskever, I. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *CoRR*, abs/1703.03864, 2017.

Salomon, R. Evolutionary Algorithms and Gradient Search: Similarities and Differences. *IEEE Transactions on Evolutionary Computation*, 2(2):45–55, 1998.

Sampson, G. Two Ideas of Creativity. *Evidence, Experiment and Argument in Linguistics and Philosophy of Language. Bern: Peter Lang*, pp. 15–26, 2016.

Santucci, V. G., Baldassarre, G., and Mirolli, M. Which Is the Best Intrinsic Motivation Signal for Learning Multiple Skills? *Frontiers in neurorobotics*, 7:22, 2013.

Santucci, V. G., Baldassarre, G., and Mirolli, M. GRAIL: A Goal-Discovering Robotic Architecture for Intrinsically-Motivated Learning. *IEEE Transactions on Cognitive and Developmental Systems*, 8(3):214–231, 2016.

Santucci, V. G., Oudeyer, P.-Y., Barto, A., and Baldassarre, G. Intrinsically Motivated Open-Ended Learning in Autonomous Robots. *Frontiers in Neurorobotics*, 13:115, 2020.

Saxe, A. M., Earle, A. C., and Rosman, B. Hierarchy through composition with multitask lmdps. 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. 3017–3026. PMLR, 2017.

Schaal, S. and others. Learning from Demonstration. *Advances in neural information processing systems*, pp. 1040–1046, 1997.

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 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1312–1320. JMLR.org, 2015.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. 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.

Schmidhuber, J. Making the World Differentiable: On Using Self-Supervised Fully Recurrent N eu al Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments. 1990.

Schmidhuber, J. Curious Model-Building Control Systems. In *Neural Networks, 1991. 1991 IEEE International Joint Conference on*, pp. 1458–1463. IEEE, 1991a.

Schmidhuber, J. A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991b.

Schmidhuber, J. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61: 85–117, 2015.

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, 2020.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1889–1897. JMLR.org, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. Parameter-Exploring Policy Gradients. *Neural Networks*, 23(4):551–559, 2010.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. 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. 8583–8592. PMLR, 2020.

Shah, J. Y. and Gardner, W. L. (eds.). *Handbook of Motivation Science*. Guilford Press, New York, 2008.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Shi, L., Li, S., Zheng, Q., Yao, M., and Pan, G. Efficient Novelty Search Through Deep Reinforcement Learning. *IEEE Access*, 8:128809–128818, 2020.

Shridhar, M., Yuan, X., Cote, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *International Conference on Learning Representations*, 2021.

Shyam, P., Jaskowski, W., and Gomez, F. Model-based active exploration. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5779–5788. PMLR, 2019.

Siegler, R. S. *Emerging Minds: The Process of Change in Children's Thinking*. Oxford University Press, 1998.

Sigaud, O., Colas, C., Akakzia, A., Chetouani, M., and Oudeyer, P.-Y. Towards Teachable Autonomous Agents. *Arxiv*, pp. 14, 2021.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. A. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 387–395. JMLR.org, 2014.

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*, 2017.

Simonton, D. K. Creative Productivity and Aging: An Age Decrement—or Not? 2012.

Simsek, Ö. and Barto, A. G. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In Brodley, C. E. (ed.), *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

Simsek, Ö. and Barto, A. G. Skill characterization based on betweenness. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 1497–1504. Curran Associates, Inc., 2008.

Singer, D. G. and Singer, J. L. *The House of Make-Believe: Children's Play and the Developing Imagination*. Harvard University Press, 2009.

Singh, S., Lewis, R. L., and Barto, A. G. Where Do Rewards Come From? In *Proceedings of the annual conference of the cognitive science society*, pp. 2601–2606. Cognitive Science Society, 2009.

Singh, S., Lewis, R. L., Barto, A. G., and Sorg, J. Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

Skinner, B. F. *Science and Human Behavior*. 1953.

Smith, L. and Gasser, M. The Development of Embodied Cognition: Six Lessons from Babies. *Artificial life*, 11(1-2):13–29, 2005.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 3483–3491, 2015.

Spelke, E. S. (ed.). *What Makes Us Smart*. MIT Press, Cambridge, Mass, 2003.

Spelke, E. S. and Kinzler, K. D. Core Knowledge. *Developmental science*, 10(1):89–96, 2007.

Stanley, K. O. and Miikkulainen, R. Evolving Neural Networks Through Augmenting Topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

Steels, L. The Artificial Life Roots of Artificial Intelligence. *Artificial life*, 1(1_2):75–110, 1993.

Steels, L. The Autotelic Principle. In *Embodied artificial intelligence*, pp. 231–242. Springer, 2004.

Steels, L. Semiotic Dynamics for Embodied Agents. *IEEE Intelligent Systems*, 21(3): 32–38, 2006.

Steels, L. (ed.). *Design Patterns in Fluid Construction Grammar*. Number v. 11 in Constructional approaches to language. John Benjamins Pub. Co, Amsterdam ; Philadelphia, 2011.

Steels, L. *Experiments in Cultural Language Evolution*, volume 3. John Benjamins Publishing, 2012.

Steels, L. and Van Trijp, R. How to Make Construction Grammars Fluid and Robust. In Steels, L. (ed.), *Constructional Approaches to Language*, volume 11, pp. 301–330. John Benjamins Publishing Company, Amsterdam, 2011.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to Summarize with Human Feedback. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021. Curran Associates, Inc., 2020.

Stout, A. and Barto, A. G. Competence Progress Intrinsic Motivation. In *2010 IEEE 9th International Conference on Development and Learning*, pp. 257–262, 2010.

Stulp, F. and Sigaud, O. Robot Skill Learning: From Reinforcement Learning to Evolution Strategies. *Paladyn Journal of Behavioral Robotics*, 4(1):49–61, 2013.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *CoRR*, abs/1712.06567, 2017.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., and Fergus, R. Intrinsic motivation and automatic curricula via asymmetric self-play. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Sussman, G. J. A Computational Model of Skill Acquisition. Technical report, HIT Technical Report AI TR-297, 1973.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991.

Sutton, R. S. and Barto, A. G. *Introduction to Reinforcement Learning*, volume 135. MIT press Cambridge, 1998.

Sutton, R. S., Precup, D., and Singh, S. P. Intra-Option Learning About Temporally Abstract Actions. In *ICML*, volume 98, pp. 556–564, 1998.

Sutton, R. S., Precup, D., and Singh, S. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. Horde: A Scalable Real-Time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768, 2011.

Szabó, Z. G. Compositionality. In Zalta, E. N. (ed.), *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2020 edition, 2020.

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.

Tasse, G. N., James, S., and Rosman, B. A Boolean Task Algebra for Reinforcement Learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.-F., and Lin, H.-T. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Tate, A. and others. Interacting Goals and Their Use. In *IJCAI*, volume 10, pp. 215–218, 1975.

Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S., and Roy, N. Approaching the Symbol Grounding Problem with Probabilistic Graphical Models. *AI magazine*, 32(4):64–76, 2011.

Thelen, E. and Smith, L. B. *A Dynamic Systems Approach to the Development of Cognition and Action*. MIT press, 1996.

Thorndike, E. L. Animal Intelligence: An Experimental Study of the Associative Processes in Animals. *The Psychological Review: Monograph Supplements*, 2(4):i, 1898.

Thorndike, E. L. A Proof of the Law of Effect. *Science*, 77(1989):173–175, 1933.

Thrun, S. Exploration in Active Learning. *Handbook of Brain Science and Neural Networks*, pp. 381–384, 1995.

Thrun, S. B. Efficient Exploration in Reinforcement Learning. pp. 44, 1992.

Todorov, E. Compositionality of optimal control laws. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, pp. 1856–1864. Curran Associates, Inc., 2009.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A Physics Engine for Model-Based Control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Tomasello, M. *The Cultural Origins of Human Cognition*. Harvard University Press, 1999.

Tomasello, M. The Item-Based Nature of Children's Early Syntactic Development. *Trends in cognitive sciences*, 4(4):156–163, 2000a.

Tomasello, M. Primate Cognition: Introduction to the Issue. *Cognitive science*, 24(3): 351–361, 2000b.

Tomasello, M. *Constructing a Language*. Harvard university press, 2009.

Tomasello, M. and Olguin, R. Twenty-Three-Month-Old Children Have a Grammatical Category of Noun. *Cognitive development*, 8(4):451–464, 1993.

Tomasello, M., Carpenter, M., Call, J., Behne, T., and Moll, H. Understanding and Sharing Intentions: The Origins of Cultural Cognition. *Behavioral and brain sciences*, 28(5):675–691, 2005.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In Lang, J. (ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 4950–4957. ijcai.org, 2018.

Turing, A. M. *Intelligent Machinery*. NPL. Mathematics Division, 1948.

Turing, A. M. Computing Machinery and Intelligence. *Mind*, 59(236):433–460, 1950.

Turner, M. The Role of Creativity in Multimodal Construction Grammar. *Zeitschrift für Anglistik und Amerikanistik*, 66(3):357–370, 2018.

Turner, M. Constructions and Creativity. *Cognitive Semiotics*, 13(1):20202019, 2020.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation 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. 6306–6315, 2017.

Van Eecke, P. and Beuls, K. Exploring the Creative Potential of Computational Construction Grammar. *Zeitschrift für Anglistik und Amerikanistik*, 66(3):341–355, 2018.

van Niekerk, B., James, S., Earle, A. C., and Rosman, B. Composing value functions in reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6401–6409. PMLR, 2019.

Vassiliades, V., Chatzilygeroudis, K., and Mouret, J.-B. Using Centroidal Voronoi Tessellations to Scale up the Multidimensional Archive of Phenotypic Elites Algorithm. *IEEE Transactions on Evolutionary Computation*, 22(4):623–630, 2017.

Vecerík, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. A. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. *CoRR*, abs/1707.08817, 2017.

Veeriah, V., Oh, J., and Singh, S. Many-Goals Reinforcement Learning. *CoRR*, abs/1806.09605, 2018.

Venkattaramanujam, S., Crawford, E., Doan, T., and Precup, D. Self-Supervised Learning of Distance Functions for Goal-Conditioned Reinforcement Learning. *CoRR*, abs/1907.02998, 2019.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical 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. 3540–3549. PMLR, 2017.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster Level in Starcraft Ii Using Multi-Agent Reinforcement Learning. *Nature*, 2019.

Vollmer, A.-L., Wrede, B., Rohlfing, K. J., and Oudeyer, P.-Y. Pragmatic Frames for Teaching and Learning in Human–Robot Interaction: Review and Challenges. *Frontiers in neurorobotics*, 10:1–10, 2016.

Vygotsky, L. S. Tool and Symbol in Child Development. In *Mind in Society*, pp. 19–30. Harvard University Press, 1930.

Vygotsky, L. S. Play and Its Role in the Mental Development of the Child. *Soviet Psychology*, 5(3):6–18, 1933.

Vygotsky, L. S. *Thought and Language*. MIT press, 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:e38546, 2019.

Walter, W. G. An Imitation of Life. *Scientific american*, 182(5):42–45, 1950.

Warde-Farley, D., de Wiele, T. V., Kulkarni, T. D., Ionescu, C., Hansen, S., and Mnih, V. Unsupervised control through non-parametric discriminative rewards. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Watkins, C. J. C. H. *Learning with Delayed Rewards*. PhD Thesis, Psychology Department, University of Cambridge, England, 1989.

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.

Weizenbaum, J. ELIZA—a Computer Program for the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 9(1):36–45, 1966.

Wellman, H. M. *The Child's Theory of Mind*. The MIT Press, 1992.

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., and Thelen, E. Autonomous Mental Development by Robots and Animals. *Science*, 291(5504):599–600, 2001.

White, R. W. Motivation Reconsidered: The Concept of Competence. *Psychological review*, pp. 37, 1959.

Whitehead, A. N. Symbolism, Its Meaning and Effect. 1927.

Whorf, B. L. *Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf*. MIT press, 1956.

Wiener, N. *Cybernetics or Control and Communication in the Animal and the Machine*. Technology Press, 1948.

Wierstra, D., Schaul, T., Peters, J., and Schmidhuber, J. Natural Evolution Strategies. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3381–3387. IEEE, 2008.

Williams, R. J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine learning*, 8(3-4):229–256, 1992.

Winograd, T. Understanding Natural Language. *Cognitive psychology*, 3(1):1–191, 1972.

Wood, D., Bruner, J. S., and Ross, G. The Role of Tutoring in Problem Solving. *Journal of Child Psychology and Psychiatry*, 17(2):89–100, 1976.

Xu, D., Nair, S., Zhu, Y., Gao, J., Garg, A., Fei-Fei, L., and Savarese, S. Neural Task Programming: Learning to Generalize Across Hierarchical Tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8. IEEE, 2018.

Yoshida, H. and Smith, L. B. Sound Symbolism and Early Word Learning in Two Languages. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 25, 2003.

Yuan, X., Côté, M.-A., Fu, J., Lin, Z., Pal, C., Bengio, Y., and Trischler, A. Interactive language learning by question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2796–2813, Hong Kong, China, 2019. Association for Computational Linguistics.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. Deep sets. 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. 3391–3401, 2017.

Zaremba, W., Sutskever, I., and Vinyals, O. Recurrent Neural Network Regularization. *CoRR*, abs/1409.2329, 2014.

Zhang, J., Wetzel, N., Dorka, N., Boedecker, J., and Burgard, W. Scheduled Intrinsic Drive: A Hierarchical Take on Intrinsically Motivated Exploration. *CoRR*, abs/1903.07400, 2019.

Zhang, Y., Abbeel, P., and Pinto, L. Automatic Curriculum Learning Through Value Disagreement. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.-F., and Lin, H.-T. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Zhao, R. and Tresp, V. Energy-Based Hindsight Experience Prioritization. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pp. 113–122. PMLR, 2018.

Zhong, V., Rocktäschel, T., and Grefenstette, E. RTFM: generalising to new environment dynamics via reading. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Zhou, L. and Small, K. Inverse Reinforcement Learning with Natural Language Goals. *CoRR*, abs/2008.06924, 2020.

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A. Target-Driven Visual Navigation in Indoor Scenes Using Deep Reinforcement Learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3357–3364. IEEE, 2017.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum Entropy Inverse Reinforcement Learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

Zlatev, J. The Epigenesis of Meaning in Human Beings, and Possibly in Robots. *Minds and Machines*, 11(2):155–195, 2001.

Zwaan, R. and Madden, C. Embodied Sentence Comprehension. *Grounding Cognition: The Role of Perception and Action in Memory, Language, and Thinking*, pp. 224–245, 2005.