Teaching Predicate-based Autotelic Agents

Learning Goal Representations with a Social Caregiver for Intrinsically Motivated Agents

by **Ahmed AKAKZIA**

Under the co-supervision of **Olivier SIGAUD** and **Mohamed CHETOUANI**

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Sorbonne University
Faculty of Science and Engineering
Major in Computer Science Telecommunication and Electronics

Date of Submission: 30/09/2022

Composition of the jury:

Pr. Martin V. BUTZ	Full Professor	University of Tübingen	Reviewer
Dr. Peter Ford DOMINEY	Research Director	CNRS	Reviewer
Dr. Pierre-Yves OUDEYER	Research Director	INRIA	Examiner
Pr. Ginevra CASTELLANO	Full Professor	Uppsala University	Examiner
Pr. Stéphane DONCIEUX	Full Professor	Sorbonne University	Examiner
Pr. Mohamed CHETOUANI	Full Professor	Sorbonne University	Supervisor
Pr. Olivier SIGAUD	Full Professor	Sorbonne University	Supervisor

To my family,
for making home feel right around the corner.
There wasn't any manual in the world
that could have prepared you for dealing with my complaints.
And yet, with love and support,
you've never stopped doing it.

Acknowledgements

The last three years marked the beginning of my journey as a young scientist — a journey through which I could never have made it alone. I would like to thank the following persons without whom I would not have completed this research.

First and foremost, my supervisors Olivier Sigaud and Mohamed Chetouani, who believed in me, gave me freedom to explore while being my safeguards and guided me with their insights and knowledge. They were always available to discuss ideas and give pointers. Thanks for reviewing all the pages I wrote since my internship up to these ones.

The Sorbonne Center for Artificial Intelligence (SCAI), for granting me the opportunity to conduct a PhD. Thanks for all the events you organized, which gave me the chance to meet other PhD candidates from many labs. You guys are inspiring, keep it up!

The AMAC team, this incredible and tasty cocktail of researchers working at the interface between Artificial Intelligence and Neuroscience. I especially thank Stéphane Doncieux, Benoît Girard, Olivier Serris, Alexandre Chenu, Elisa Massi, Johann Huber, Elias Hanna, Nicolas Fontbonne, Jeremy Fersula, Jeanne Barthélemy, Yoones Mirhosseini, Giuseppe Paolo, Maud de Tollenaere, Matthieu Sarazin, Astrid Merckling, Thomas Pierrot, Nicolas Perrin-Gilbert and Pierre Fournier for the exciting scientific discussions. As always, 310's gang rules!

The ISIR lab, one of the best environments for a PhD. I would like to thank all the members, and I am especially grateful to Hugo Caselles-Dupré and Ali Hammoud, with whom I had the chance to exchange ideas about subjects related to our field of research.

All my co-authors, who were very crucial to the work I have published: Cédric Colas, Pierre-Yves Oudeyer, Firas Jarboui, Olivier Serris, Hugo Caselles-Dupré, Mohamed Chetouani and Olivier Sigaud. More details about these collaborations at the end of the introduction.

Awatef Barra and Sylvie Piumi, very helpful and always responsive whenever it comes to administrative stuff.

I am grateful to my friends, who supported and never stopped believing in me during these last three years. When everything seemed dark and blurry, especially during the lockdown, they were the lighthouse that helped me keep moving forward. Thank you guys!

My biggest thanks go to my family: Mom and Dad, I wish I could make you proud as you gave me the greatest gift of all; My brother and sister, my guardian angels, I will always look up to you; My brother and sister in law, for your inspiring love and generosity; My three little nieces and nephew, you're the cherry on the cake, the glue that hold us all. Thank you 3000!

I thank every person that knows me. Crossing paths with you certainly made me the person I am today.

> *"Go back? No good at all! Go sideways? Impossible! Go forward? Only thing to do! On we go." Bilbo Baggins, The Hobbit.*

Teaching Predicate-based Autotelic Agents

Learning Goal Representations with a Social Caregiver for Intrinsically Motivated Agents

Abstract

As part of the quest for designing embodied machines that autonomously explore their environments, discover new behaviors and acquire open-ended repertoire of skills, artificial intelligence has been taking long looks at the inspiring fields of developmental psychology and cognitive sciences which investigate the remarkable continuous and unbounded learning of humans. This gave birth to the field of developmental robotics which aims at designing autonomous artificial agents capable of self-organizing their own learning trajectories based on their intrinsic motivations. It bakes the developmental framework of *intrinsically motivated goal exploration processes* (IMGEP) into *reinforcement learning* (RL). This combination has been recently introduced as *autotelic reinforcement learning*, where *autotelic agents* are intrinsically motivated to self-represent, self-organize and autonomously learn about their own goals. Naturally, such agents need to be endowed with good exploration capabilities as they need to first *physically encounter* a certain goal in order to *take ownership of* and *learn about* it. Unfortunately, discovering interesting behavior is usually tricky, especially in *hard exploration setups* where the rewarding signals are parsimonious, deceptive or adversarial. In such scenarios, the agents *physical situatedness* — in the Piagetian sense of the term — seems insufficient. Luckily, research in developmental psychology and education sciences have been praising the remarkable role of *socio-cultural* signals in the development of human children. This *social situatedness* — in the Vygotskyan sense of the term — enhances the toddlers exploration capabilities, creativity and development. However, deep RL considers social interactions as dictating instructions to the agents, depriving them from their autonomy. This research introduces *teachable autotelic agents*, a novel family of autonomous machines that can learn both alone and from external social signals. We formalize such a family as a *hybrid goal exploration process* (HGEPs), where autotelic agents are endowed with an internalization mechanism to rehearse social signals and with a goal source selector to actively query for social guidance.

The present manuscript is organized is two parts. In the first part, we focus on the design of teachable autotelic agents and attempt to leverage the most important properties that would later serve the social interaction. Namely, we introduce *predicate-based autotelic agents*, a novel family of autotelic agents that represent their goals using spatial binary predicates. These insights were based on the *Mandlerian view* on the prelinguistic concept acquisition suggesting that toddlers are endowed with some innate mechanisms enabling them to translate *spatio-temporal* *information* into an *iconic static form.* We show that the underlying semantic representation plays a pivotal role between raw sensory inputs and language inputs, enabling the decoupling of sensorimotor learning and language grounding. We also investigate the design of such agents' policies and state-action value functions, and argue that combining *Graph Neural Networks* (GNNs) with relational predicates provides a light computational scheme to transfer efficiently between skills. In the second part, we formalize social interactions as a goal exploration process. We introduce *Help Me Explore* (HME), a novel social interaction protocol where an expert social partner progressively guides the learning agent beyond its *zone of proximal development* (ZPD). The agent actively selects to query its social partner whenever it estimates that it is not progressing enough alone. It eventually internalizes the social signals, becomes less dependent on its social partner and maximizes its control over its goal space.

Keywords: deep reinforcement learning, intrinsic motivations, goal-conditioned behavior, autotelic agents, symbolic behavior, relational inductive bias, graph neural networks, transfer learning, open-ended learning, skill acquisition.

Enseigner des Agents Autotéliques basés sur des Prédicats

Apprentissage des Représentation de buts avec un Partenaire Social pour les Agents Intrinséquement Motivés

Résumé

Dans la quête de concevoir des machines incarnées qui explorent leurs environnements en autonomie, découvrent des nouveaux comportement et apprennent des répertoires non-bornés de compétences, l'intelligence artificielle s'est longuement inspirée des domaines de psychologie du développement et des sciences cognitives qui étudient la capacité remarquable des humains à apprendre tout au long de leur vie. Ceci a donné naissance au domaine de la robotique du développement qui a pour but de concevoir des agents artificiels autonomes capables d'auto-organiser leurs trajectoires d'apprentissage en se basant sur leurs motivations intrinsèques. Ce domaine combine les *processus d'exploration de but intrinséquement motivé* (IMGEPs) et *l'apprentissage par renforcement* (RL). Cette combinaison est connue sous le nom d'*apprentissage par renforcement autotélique*, où des *agents autotéliques* sont intrinséquement motivés pour représenter, organiser et apprendre leurs propres buts. Nautrellement, ces agents doivent démontrer de bonnes capacités d'exploration puisqu'ils ont besoin de *découvrir physiquement* les buts pour pouvoir les apprendre. Malheureusement, découvrir des comportements intéressants peut être compliqué, surtout dans les *environnements d'exploration difficile* où les signaux de récompenses sont parcimonieux, déceptifs ou contradictoires. Dans ces scénarios, la *situation physique* des agents semble insuffisante. Heureusement, la recherche en psychologie du développement et les sciences de l'éducation soulignent le rôle important des signaux socio-culturels dans le développement des enfants humains. Cette *situation sociale* améliore les capacités d'exploration des enfants, leur créativité et leur développement. Cependant, l'apprentissage par renforcement profond considère l'apprentissage social comme une imposition d'instructions aux agents, ce qui les prive de leur autonomie. Dans ce document, nous introduisons les *agents autotéliques enseignables*, une nouvelle famille de machines autonomes qui peuvent apprendre à la fois toutes seules et à travers des signaux sociaux externes. Nous formalisons cette famille en tant que *processus d'exploration de but hybride* (HGEPs), où les agents autotéliques sont augmentés d'un mécanisme d'internalisation leur permettant de rejouer les signaux sociaux et d'un selecteur de source de buts pour demander activement de l'aide sociale.

Ce document est organisé en deux parties. Dans la première partie, nous nous concentrons sur la conception d'agents autotéliques enseignables et nous essayons d'implémenter des propriétés qui faciliteraient l'interaction sociale. Notamment, nous introduisons les *agents autotéliques basés sur les prédicats*, une nouvelle famille d'agents autotéliques qui représentent leurs buts en utilisant des prédicats binaires spatiaux. Nous montrons que l'espace de représentation sémantique sous-jacent joue le rôle de pivot entre la représentation sensorimotrice et le langage, permettant un découplage entre l'apprentissage sensorimoteur et l'ancrage du langage. Nous étudions également la conception des politique et des fonctions valeurs état-action et nous soutenons que la combinaison des réseaux de neurones graphiques (GNNs) et des buts en prédicats relationnels permet l'utilisation de schémas computationnels légers qui transfèrent bien entre les tâches. Dans la deuxième partie, nous formalisons les interactions sociales en tant que processus d'exploration de buts. Nous introduisons *Help Me Explore* (HME), un nouveau protocole d'interaction sociale où un partenaire social expert guide progressivement l'agent au-delà de sa *zone de développement proximale* (ZPD). L'agent choisit activement de lancer des requêtes à son partenaire social dès qu'il estime qu'il ne progresse plus sur les buts qu'il connait déjà. Il finit éventuellement par internaliser ces signaux sociaux, devient moins dépendant envers son partenaire social et arrive à maximiser son contrôle de son espace de buts.

Keywords: apprentissage par renforcement profond, motivations intrinséques, agents autotéliques, comportement symbolique, réseaux de neurones graphiques, apprentissage non-borné.

Contents

List of Figures

List of Tables

Introduction

Humans are the most remarkable learners we know to exist. From a very young age, toddlers begin to freely interact with their surroundings, crawling around the living room, playing with the curtains of the backyard french door or throwing the ball under the sofa. They never stop acquiring skills as they grow older, piling up colored blocks to construct towers, drawing sketches and playing hide-and-seek with each other. They seem to be inherently motivated to grow a large and diversified repertoire of skills. Actually, psychologists define the learning and development process in human children as *open-ended*: children engage with *non-stationary* and *continuously changing* environments through *learner-centered activities* to acquire new capabilities of growing complexities [Piaget, 1977, Hannafin et al., 1994, Thelen and Smith, 1996, Smith and Gasser, 2005].

Recently, the fields of robotics and artificial intelligence (AI) have been striving to design embodied autonomous machines that interact with complex environments and continuously learn new skills the way human infants do. Such agents are by definition *physically situated*: they understand the world and respond to its signals based on physical interactions. Reinforcement Learning (RL) has been introduced as a conceptual and mathematical framework to train such agents [Sutton and Barto, 2018]. Motivated by Thorndike's principle of *instrumental conditioning* — actions followed by reinforcing signals such as food or praise are more likely to be reproduced [Thorndike, 1898, 1911] — RL represents a backbone for autonomous skill acquisition in artificial agents. However, standard RL usually requires defining a single reward signal for a single task. Interestingly, goal-conditioned RL (GC-RL) was introduced as an RL-based paradigm for learning multiple skills [Schaul et al., 2015]. Artificial agents trained within GC-RL produce actions that are conditioned on some predefined context or *goal*. By training on multiple goals, such agents are able to grow their repertoire of skills and diversify their behavior. However, the set of goals on which these agents train are usually domain-specific and hand-engineered *a priori*. This does not account for the learner-centered activities in human learners which rely on *intrinsic motivations* to define their purposes and intents.

Intrinsically Motivated AI

Exploring, making sense of the surroundings, discovering and learning new skills seems to come spontaneously in children. The role of play in infants goes beyond giving pleasure, but marks one of the first contexts within which they use their needs and incentives to act [Vygotsky, 1978b]. Actually, behavior in children, and in humans in general, seems to be driven by *intrinsic motivations* that push them towards experiencing novelty and fulfilling their curiosity to understand and control their surroundings [Gopnik et al., 1999, Oudeyer and Kaplan, 2007, Oudeyer et al., 2016, Rule et al., 2020].

Recently, the field of robotics has been taking inspiration from these insights to model embodied artificial agents that use intrinsic motivations to explore and learn about their environment [Baldassarre and Mirolli, 2013, Cangelosi and Schlesinger, 2018]. The main challenge has been to either augment or completely replace the standard external reinforcing signals in RL with some internal motivators that the agents construct by themselves during their lifetime [Schmidhuber, 2010, Kaplan and Hafner, 2006]. Existing approaches in intrisincally motivated AI can be categorized in two families. On the one hand, there are methods that aim at *minimizing the agents uncertainty* over their surroundings. The underlying agents use proxies such as novelty, surprise and disagreement to update their current knowledge and expectations [Bellemare et al., 2016, Burda et al., 2018, Pathak et al., 2019, Raileanu and Rocktäschel, 2020]. On the other hand, there are approaches where artificial agents aim at *maximizing their control* of their environments. These methods use estimations of learning progress and competence to self-organize their skills to facilitate their acquisition. In the next paragraph, we briefly introduce to combination of intrinsic motivations and goal-based learning in AI.

Autotelic AI

In the quest for the autonomous acquisition of large repertoires of skills, baking these kinds of intrinsic motivations into GC-RL agents seems promising. This combination has been recently introduced as *autotelic* RL. The term autotelic first originated from the theory of *flow* introduced by Csikszenmihaly [Mihaly, 2000]. It refers to activities whose motivating purposes (*telos*) come from the embodied agents themselves (*auto*). Recent approaches have been attempting to incorporate this *autotelic principle* within AI [Steels, 2004, Colas et al., 2022b]. The underlying agents are *self-motivated*, *self-organized* and *self-developing*.

In the context of GC-RL, autotelic agents have been defined as intrinsically motivated agents that spontaneously represent, set, pursue and learn about their own goals [Colas et al., 2022a]. Such agents need to autonomously explore their environments, discover the skills they can potentially achieve and learn an optimal goal-conditioned policy that masters them all. In these agents start with no clue about the goals they can physically reach, they need to efficiently manage the *exploration-exploitation dilemma*. In fact, as they would be continuously acquire new goals, they ought to know when to focus on learning the already discovered ones and when to start exploring again. Consequently, methods tackling this problem usually train an *internal goal-generator module* that maximizes the coverage of the goal space, prioritizing the exploration of sparsely achieved regions over others.

Notably, some environments are hard to explore in the first place. In such scenarios, random uninformed actions are not likely to yield interesting states for these agents to learn from. This issue translates the effect of modelling only *physical situatedness* traits — interacting with objects, discovering and learning only from these interactions.

Socially-Situated AI

Back to our main source of inspiration, human children are also *socially situated*. Besides from being autonomous open-ended learners, developmental psychology and education sciences have shown that infants also benefit from external *socio-cultural signals* that guide their exploration, drive their motivations and affects their development [Wood et al., 1976, Vygotsky, 1978b, Bruner, 1990, 1991, Lindblom and Ziemke, 2003, Tomasello, 2005, Weisberg et al., 2016, Yu et al., 2018]. Shortly after birth, infants can imitate facial expressions such as happiness and surprise [Field et al., 1983, Meltzoff and Moore, 1983, Meltzoff, 1988]. Besides, findings from developmental science argue that human toddlers register the equivalence between their own actions and the actions of others even before language acquisition [Meltzoff, 2007]. This *social responsiveness* continues to develop as toddlers grow. They start performing role plays together where they imitate their caregivers or some character they watched on television, they react to their caregivers gaze, pointing and reprimanding facial expressions. They also ask for help from their caregivers whenever they attempt to accomplish a relatively complex task such as riding a bike, drawing a lotus flower or constructing a double tower. They might later internalize these external signals and manage to fulfill these task autonomously.

Combining social learning and intrinsic motivations in artificial agents has been investigated as a promising way to help agents explore and continuously acquire new skills without having to implement heavy exploration mechanisms [Schaal et al., 2003, Thomaz et al., 2006, Thomaz and Breazeal, 2008b, Peters and Schaal, 2008, Kober and Peters, 2011, Stulp and Schaal, 2011, Nguyen and Oudeyer, 2014a, Najar et al., 2016, Fournier et al., 2017, Najar et al., 2020, Caselles-Dupré et al., 2022]. All these works argue that the social guidance provided by humans can drive the learner to new regions of its state space, where its own random exploration alone is usually insufficient. The agents' own intrinsic motivations could then take the wheel by further exploring these new areas. On the one hand, such a combination makes use of the *broader aspect* of intrinsic motivations, that is *exploring as many goals as possible*. On the other hand, it only profits from the *specialized aspect* of social learning, that is *following a specific instruction*. These ideas were studied not only in the context of learning one single skill [Schaal et al., 2003, Peters and Schaal, 2008, Kober and Peters, 2011, Stulp and Schaal, 2011], but also in achieving a variety of goals [Thomaz et al., 2006, Thomaz and Breazeal, 2008b, Nguyen and Oudeyer, 2014a].

Since human intelligence is inherently symbolic, endowing artificial agents with some symbolic systems might actually facilitate the intervention of human caregivers within the training process of embodied machines. In the next paragraphs, we frame the connectionist vs. symbolic dualism within AI.

Connectionist and Symbolic AI

A long-standing debate marking the fields of psychology and biology is the *nature versus nurture* debate [Haldane, 1946, Moore, 2003, Carlson et al., 2005, Gruber, 2013, Moore, 2015, Normile, 2016]. It attempts to determine the factors responsible for skill acquisition and development in humans: genetics (nature) or environment (nurture). On the one hand, nature means acquiring knowledge, representing it and reasoning about it are grounded in some programmed cognitive functions within humans. On the other hand, nurture means that knowledge representations are acquired from continuous interactions with the world [Gruber, 2013].

Recently, the traces of such a debate have been translated into the field of artificial intelligence, where *connectionist* AI has been confronted to *symbolic* AI. The connectionist view suggests that designing artificial agents that generalize to many domains should involve as little engineering as possible — that is, few to none programmed and hand-engineered functions. It is motivated by the fact that most hard-coded features and modules would most likely be *contextual* — they would only be valid in some cases but never broadly. Modern deep learning approaches adopt a connectionist view, following an end-to-end scheme to design models which use minimal a priori representational and computational assumptions to avoid hand-engineering. These methods capitalize on the abundance of cheap data and computational resources, which resulted in end-to-end breakthroughs in the fields of image classification [Redmon et al., 2016, Ren et al., 2015] and video processing [Zhang et al., 2016] to speech recognition [Hinton et al., 2012] and neural machine translation [Luong et al., 2015, Wu et al., 2017]. This was highlighted by Sutton in his formulation of *the bitter lesson*:

> *"The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin."*
> *Sutton [2019]*

However, transferring beyond training conditions and learning from little experience remains a key challenge for these approaches. More specifically, the field of RL, which requires artificial agents to perform online interaction with their environment, should not require huge amounts of training steps to alleviate both economic and ecological burdens.

By contrast, the symbolic view argues that human intelligence is inherently symbolic, and thus implements symbolic systems within artificial agents would make them intelligent [Newell and Simon, 2007]. It is mainly based on the *physicaly symbol system hypothesis* introduced by Newell and Simon:

> *"A physical symbol system has the necessary and sufficient means for general intelligent action."*
> *Newel and Simon [1976]*

Methods within the symbolic AI view build models that are predefined, and endow their systems with such models. This lead to several successes in problems such as theorem proving [Newell and Simon, 1956], puzzle resolution [Newell et al., 1959] and instruction-following [Winograd, 1972]. More recent works argued for the implementation of symbolic behaviour in embodied agents [Santoro et al., 2021]. They suggest that symbols might not be fixed, but should rather represent a background with which agents would interact. These interactions would eventually update these symbols, making them more suitable for the tasks they are confronted to.

Interestingly, psychologist Donald Hebb frames a rather interesting rhetorical question on the predominance of nature or nurture: "Which contributes more to the area of a rectangle, its length or its width?". This suggests that connectionist and symbolic AI could actually be complementary rather than competing views. Recently, many research directions argued on the importance of combining connectionism and symbolism, claiming that using them jointly yields wholes which are greater than the sums of their parts. Among these works, *neuro-symbolic* approaches have been shown to be promising in a variety of domains [Andreas et al., 2016, León et al., 2020]. The present research aligns with these approaches.

Teachable Autotelic AI

The present research represents one of the first steps towards endowing autotelic RL agents with the capacity to also learn from external social signals provided by expert social partners. We introduce a new family called *teachable autotelic agents* (TAAs). These agents are intrinsically motivated to learn about their own goals, but they can also be taught from external caregivers. This enables such agents to be social and autonomous at the same time: their intrinsic motivations are influenced by their caregiver's interventions, but they can still interact with their environment and learn new things even if they were left alone. This suggests that the exploration process in TAAs depends on both intrinsic motivations and external guidance signals that agents could eventually internalize and take ownership of. We focus on this issue from a *goal exploration process* point of view. We attempt to answer these scientific questions:

- How can TAAs represent their goals to facilitate the external guidance by expert caregivers ?
- How can we design TAAs' internal models' architecture to enable efficient transfer and generalization between different skills ?
- How can we formalize the process of learning multiple goals under the assistance of social caregivers ?

- How can we construct efficient social interaction protocols enabling caregivers to help TAAs scaffold their skills and learn as much goals as possible ?

Contributions

The central purpose of the present research is to make progress towards the autonomous acquisition of skills under social guidance. We build upon the related AI approaches we outlined in the previous paragraph and introduce novel methods, frameworks and conceptualizations that form the family of socially-guided and intrinsically motivated goal-conditioned agents, which we call *Teachable Autotelic Agents*.

The first contribution in the present research is a novel family of optimization criteria which we call the *delayed geometric discounting criteria*. This family generalizes the standard geometric discounts to policies with longer mixing times. Underlying policies give more weight to rewards that are farther in the future, thus yielding agents capable of sacrificing short term returns for better rewards in the long run. We show that this family solves several hard exploration problems which include sparse, deceptive and adversarial rewarding signals. This is presented in Chapter 1.

The second contribution outlined in the present document is a *teachability checklist*, a set of properties characterizing the teaching process in humans. We argue that these properties should be leveraged in artificial agents to enable them to be efficiently taught by external caregivers. We also investigate current works in AI and point out the main teachability properties they lack. This is presented in Chapter 2.

The third contribution outlined in the present document is a novel family of autotelic reinforcement learning agents that represent their goals using sets of binary predicates based on spatial relationship.

The fourth contribution is a language goal generator which build upon the first contribution and facilitates language grounding in the embodied agents' sensorimotor behavior. These two contributions are baked within a novel architecture which we call *Language-Goal-Behavior* (LGB). In Chapter 3, we show that this architecture enables autotelic agents to increase their behavioral diversity and follow natural language-based instructions.

The fifth contribution consists in an investigation of the design of goal spaces and policy architectures in autotelic agents. More specifically, we consider *Graph Neural Networks* (GNNs) as technical tools, and argue that it can be combined with the structured predicate-based goal spaces to yield better transfer and generalization capabilities. This is discussed in Chapter 4

The sixth contribution of the present research is a novel family of goal exploration processes that handles multiple sources of goals. More specifically, we focus on goals generated by *external programs* and goals selected by the agents *autonomously*, and introduce *Hybrid Goal Exploration Processes* (HGEPs). We show that external goals enable agents to discover new regions of their goal spaces. This is outlined in Chapter 5.

Finally, our last contribution consists in a social interaction protocol which we call *Help Me Explore* (HME). HME involves an expert social partner (SP) and autotelic agents that are able to *internalize social signals* and *actively select to query* their SP for help. They rely on their internal models to estimate their learning progress, and ask for guidance whenever they estimate that they are not progressing enough. The SP suggests goals based on the agents' current exploration limits. Their goal is to take the learning agents beyond their *zone of proximal development* (ZPD), which represents the space between what agents can do alone and what they can do with the help of expert caregivers. The artificial agents eventually grow their repertoire of skills, progressively shrinking their ZPD until they become fully independent of their SP.

All our studies are based on *state-of-the-art* techniques from deep reinforcement learning, including optimization algorithms, exploration and relabeling techniques.

Document Structure

This manuscript is organized in two parts:

1. First, we introduce Teachable Autotelic Reinforcement Learning Agents, a family of goal-conditioned embodied machines that use both their own intrinsic motivations and external social signals to explore their environment, discover new skills and learn about them. We take inspiration from works in developmental psychology and education sciences to present a *teachability checklist* — a set of properties characterizing teaching in humans. We argue that autotelic agents which represent their goals as sets of binary spatial predicates and use *Graph Neural Networks* to model their policies exhibit a subset of these properties.

2. Second, we investigate mechanisms for Teaching Autotelic Reinforcement Learning Agents. We introduce a novel goal exploration framework which we call *Hybrid Goal Exploration Processes* (HGEPs). HGEPs formalize the goal discovery, goals exploration and goal learning in autotelic agents that also receive external signals in the form of external goals. We also introduce *Help Me Explore* (HME), a social interaction protocol involving an expert caregiver and autotelic agents that can *internalize external signals* and *actively select queries*.

We integrate this research into the current related works, discuss its limits as well as future perspectives.

Collaborations

I believe that one of the most inspiring and fruitful assets of conducting a PhD is the opportunity to establish different collaborations and collectively dig into a particular project. The present research is the outcome of a quite hard teamwork and interesting discussions with other researchers. All along these last three years, I had the chance not only to work with my supervisors Olivier Sigaud and Mohamed Chetouani, who graciously helped me refine my understanding of the different challenges involving our research subject and guided me with their useful insights and experience, but also with other colleagues from inside and outside of our lab. Some of these collaborations gave birth to preprints, conference and workshop papers. Hereby, I enlist the ones whose resulting papers are used within the present manuscript:

- I have worked with Cédric Colas and Pierre-Yves Oudeyer (INRIA) on the study in Chapter 3. Their expertise on intrinsically motivated agents helped me better understand many conceptual and technical features. I continued working with Cédric Colas on the study in Chapter 6.

- I have worked with Firas Jarboui (ENS) on defining a novel family of criteria that generalize the standard geometrically discounted RL in the study of Chapter 1. His insights and expertise in the domain helped me a lot refine the idea that the main problem with standard RL approaches is the focus on immediate rewards, which rapidly becomes problematic in hard exploration setups.

- I have worked with Olivier Serris (ISIR) and Hugo Caselles-Duprés (ISIR) on designing a social interaction protocol in Chapter 6, where expert social partners can help autotelic agents explore complex environments. I have learned a lot from the discussions we had on the subject, and their insights were very crucial to my own development.

I had many other fruitful discussions which did not result in publications but were extremely crucial to both my personal and technical development within this brief journey. For more details, see the Acknowledgments.

Publications

The present research includes several publications in conferences and workshops and other pre-prints:

- Delayed Geometric Discounts: An alternative criterion for Reinforcement Learning (pre-print), Chapter 1 [Jarboui and Akakzia, 2021]

- Towards Teachable Autonomous agents (pre-print), Chapter 2 [Sigaud et al., 2021]

- Grounding Language to Autonomously-Acquired Skills via Goal Generation (ICLR 2021), Chapter 3 [Akakzia et al., 2021]

- Language-Conditioned Goal Generation: a new approach to language grounding for RL (ICML 2021, LaReL workshop), Chapter 3 [Colas et al., 2020a]
- Learning Object-Centered Autotelic Behaviors with Graph Neural Networks (CoLLAs 2022), Chapter 4 [Akakzia and Sigaud, 2022].
- Help Me Explore: Minimal Social Interventions for Graph-Based Autotelic Agents (IMOL 2022), Chapter 6 [Akakzia et al., 2022].

Software

We open-source all the software corresponding to each of the studies conducted in the present research. All the repositories can be found on github under the *akakzia* account:

- The Eta-optimality Software: /eta_optimality
- The Fetch Manipulate Software: /gym-object-manipulation
- The LGB-DECSTR Software: /decstr
- The RL-Graph Software: /rlgraph.
- The GANGSTR Software: /gangstr
- The HME Software: /help_me_explore

Part I

Teachable Autotelic Reinforcement Learning Agents

Recently, *Autotelic Reinforcement Learning Agents* were introduced as a family of embodied *goal-conditioned agents* that are *intrinsically motivated* to represent, set and pursue their own goals. In the first part of this manuscript, we introduce a novel family of goal-conditioned reinforcement learning agents which we call *Teachable Autotelic Reinforcement Learning Agents*. These agents are both *autonomous* and *socially situated*: they are capable of learning alone, but can still benefit from social signals provided by external human tutors, thus mimicking the process of *guided-play* within human toddlers. The main objective of this part is to design artificial agents capable of autonomously acquiring open-ended repertoires of skills while being sensitive to external social signals that guide their exploration. We incrementally develop this idea in three chapters:

- Chapter 1 bridges the gap between the *Reinforcement Learning* (RL) paradigm and *Open-Ended Learning*. It highlights the fundamental concepts and computational frameworks that tackle key challenges in continuously learning skills of increasing complexity. We first discuss the limitations of standard RL methods in training agents to overcome hard exploration problems and simultaneously learn multiple skills. Second, we introduce a novel family of RL algorithms that allow artificial agents to discard short term rewards in favor of better long term ones, thus overcoming hard exploration problems where rewards are sparse, deceptive or adversarial. Third, we present a sub-family of RL methods called Goal-Conditioned Reinforcement Learning (GC-RL). These methods yield artificial agents that learn contextual policies conditioned on *goals*, thus allowing them to master a multitude of skills simultaneously. Finally, we focus on GC-RL methods and introduce *goal exploration processes* as a family of algorithms that aim at maximizing the coverage of goal spaces.

- Chapter 2 introduces *teachable autotelic agents* from a developmental perspective. Its central objective is to provide a road map towards designing artificial agents that can learn goal-directed behaviors in autonomy and with a human tutor. First, we present key works investigating the ”natural teaching of a child” [Turing, 1948] to extract main properties found in human learners, human teachers and in the tutoring process as a whole [Wood et al., 1976, Vygotsky, 1978b, Bruner, 1985]. Then, we present an overview on works in the field of Artificial Intelligence that attempt to leverage some of these properties, and highlight their limitations.

- Chapter 3 introduces our first steps towards designing artificial teachable agents. *Predicate-based* learning is the central concept developed in this chapter. We start from a developmental perspective on the prelinguistic concept formation in human toddlers. More specifically, we adopt *the Mandlerian view*, which proposes a theory of *Perceptual Meaning Analysis* that builds on primitive attention to temporal information and translates it into static spatial concepts [Mandler, 2012]. Then, we formalize this idea and introduce *predicate-based semantic configurations* as tools to represent goal spaces based on spatial relational primitives. We argue that endowing artificial agents with such representations accounts for the Mandlerian View. Finally, we introduce the *Language-Goal-Behavior* (LGB) architecture which decouples skill learning and language grounding. The LGB architecture uses semantic configurations as a pivot between autotelic skill learning and social language grounding.

- Chapter 4 studies the transfer and generalization capabilities of goal-based artificial agents. It argues that the behavioral diversity of autotelic agents is *grounded* in their goal space representation and their policy design. First, we start from a developmental point of view on the relationship between *perception* and *conception* in humans. First, we argue that humans are endowed with intrinsic mechanisms that enable them to perceive their world in a structured fashion and to interact with it by bootstrapping primitive sub-routines. This probably explains why humans exhibit impressive combinatorial generalization: they are able to handle new situations by efficiently combining previously learned skills Then, we show how *Graph Neural Networks* (GNNs) represent promising tools for endowing artificial agents with combinatorial generalization capabilities. Finally, we investigate the learning and transfer capabilities of autotelic agents in multi-object manipulation scenarios using different goal space representations and policy architecture. Namely, we study 4 graph-based architectures: *full graph networks* [Battaglia et al., 2018], *interaction networks* [Battaglia et al., 2016], *relation networks* [Santoro et al., 2017] and *deep sets* [Zaheer et al., 2017].

Chapter 1

Reinforcement Learning Meets Open-Ended Learning

Open-ended learning is a key characteristic of learning and development in human children [Piaget, 1977, Hannafin et al., 1994, Thelen and Smith, 1996, Smith and Gasser, 2005]. It consists in interacting within an *open-ended environment* promoting cognitive and embodied engagement through learner-centered activities [Hannafin et al., 1994]. Such environments are distinctively *non-stationary*, where new objects might appear and sudden concept drift might occur due to unknown dynamics. Children evolving in open-ended environments usually have few grounding rules and a lot of freedom to explore within these rules. They build their own purposes and intents while playing, for instance deciding what structure they want to build from colored blocks, which instrument they want to play and what kindergarten song they want to sing. This requires a sophisticated level of cognitive functioning and the ability to acquire repertoires of behaviours of *increasing difficulty* [Hannafin et al., 1994, Land, 2000, Baldassarre et al., 2014]. The continuous acquisition of skills is usually referred to as *lifelong learning* [Thrun, 1998, Aspin and Chapman, 2000, Laal, 2011]. On the one hand, it is a *continuous* learning process where children adapt their behavior without forgetting what they have already acquired. On the other hand, it is an *incremental* learning process, where children progressively add building bricks to their previously acquired behaviors to build more complex ones.

In the quest for leveraging similar properties within Artificial Intelligence (AI), the field of Reinforcement Learning (RL) represents a promising backbone for open-ended learning in artificial agents. Originally inspired from empirical results in Psychology [Thorndike, 1898, 1911] and Neuroscience [Olds and Milner, 1954, Montague et al., 1996], RL offers a computational framework for automating decision making in embodied artificial agents [Sutton et al., 1998, Sutton and Barto, 2018]. Nevertheless, most AI research acknowledges the insufficiency of RL in tackling the open-ended learning problem, which introduces two main challenges that standard RL methods fail to circumvent.

On the one hand, acquiring open-ended repertoires of skills requires artificial agents to address *hard exploration* and *deceptive* problems. While the former is characterized by the rarity of the rewarding signal which obstructs exploration in standard setups, the latter corresponds to situations where agents need to sacrifice early low rewards in order to receive higher ones that come later in their life span. Standard RL approaches are usually based on maximizing *cumulative discounted reward criteria*. They usually use geometric discounts, attributing higher coefficients to earlier rewarding signals. This induces a build-in bias against policies with longer mixing times. Hence, the underlying learning agents are likely to fall in local minima corresponding to deceptive rewards (in deceptive problems), and struggle from sample inefficiency when back-propagating the reinforcing signals to initial states (in hard exploration problems). Some approaches use *average reward criteria*, but the underlying methods are intractable and yield computational instabilities in long-horizon scenarios.

On the other hand, the decision making module — known as the policy in RL agents — needs to be flexible and sensitive to different purposes and goals. In fact, open-ended learning requires mastering a multitude of tasks by definition. This suggests that artificial agents need to be somehow aware of the task they are attempting to fulfill, and take actions accordingly. Whenever another task is at hand, they need to be able to adaptively *switch their strategy*. Standard RL approaches are not well suited to design artificial agents that master multiple tasks (unless they define and train a different policy for each task and consider the agent as an ensemble of separate policies). In fact, an RL problem is usually mapped to a single Markov Decision Process (MDP), which handles a single reward function that defines a single task or goal.

In this section, we introduce computational frameworks that use RL as a backbone to tackle the two challenges discussed above. In Section 1.1, we formalize RL as a paradigm for automating learning and decision making in artificial agents. In Section 1.2, we bridge the gap between discounted and average cumulative rewards by introducing *delayed geometric discounting criteria*. In Sections 1.3 and 1.4, we present the *Goal-Conditioned Reinforcement Learning* paradigm. First, Section 1.3 introduces a formal definition of *goals* and of the *multi-goal* RL problem, a typology of the existing goal representations and an overview of the existing methods. Second, Section 1.4 presents goal-based artificial learning agents as *Goal Exploration Processes*. In Section 1.5, we focus on *autotelic RL*, a sub-family of goal exploration processes where agents use intrinsic motivations to represent, select and pursue their own goals.

1.1 The Reinforcement Learning Paradigm

Reinforcement Learning (RL) is a computational approach to understanding and automating goal-directed learning and decision making. Compared to other computational approaches, RL considers embodied and situated agents: agents that directly interact with their environment through autonomous actions without requiring supervision or any model of the environment. In this section, we first motivate the choice of this particular paradigm based on empirical results from psychology and neuroscience and on early thoughts in artificial intelligence (Section 1.1.1). Then, we present a formal definition of RL problems (Section 1.1.2). Finally, we exhibit several categories of RL methods which are the most relevant to our work (Section 1.1.3).

1.1.1 Motivations

One of the first connections that infants establish with their environment is undoubtedly sensorimotor. In fact, within minutes of their birth, infants already start waving their arms and looking about. As they grow up, infants further exercise this connection to learn how their environment works and what to do in order to achieve their goals. Therefore, learning from interaction is the backbone of most theories of learning and intelligence.

The motivation behind the modern field of RL is the idea of *trial-and-error* which originated from the psychology of animal learning. One of the first psychologists to develop a comprehensive formalization of the idea of trial-and-error as a principle of learning was *Edward Thorndike*. He was the first to introduce the term of *instrumental conditioning* to describe experiments in which reinforcement is contingent upon behavior [Thorndike, 1898]. This type of conditioning is opposed the *classical conditioning*, where reinforcing stimulus is independent of the animal's behavior [Pavlov, 1904, Pavlov and Gantt, 1928].

To establish the effect of instrumental conditioning, Thorndike proposed experiments where cats were placed in puzzle boxes with different escape mechanisms. In all experiments, food was placed outside the box so that it was visible to the cats, which had to perform a sequence of three separate actions to break out (pulling strings, pushing bars, depressing platforms) [Thorndike, 1898], see Figure 1.1 for an illustration. Thorndike observed that the time to escape over multiple experiences decreased as the cat encountered successive trials. Hence, he described the cats' behavior as follows:

> *"The cat that is clawing all over the box in her impulsive struggle will probably claw the string or loop or button so as to open the door. And gradually all the other non-successful impulses will be stamped out and the particular impulse leading to the successful act will be stamped in by the resulting pleasure, until, after many trials, the cat will, when put in the box, immediately claw the button or loop in a definite way."Thorndike [1898]*

This behavior describes the effect of reinforcing stimulus on the animals' actions, and thus was called by Thorndike *the law of effect*. Since then, this law has been regarded as a founding principle of most influential learning theories [Hull, 1943, 1952] and experimental methods [Skinner, 1938].

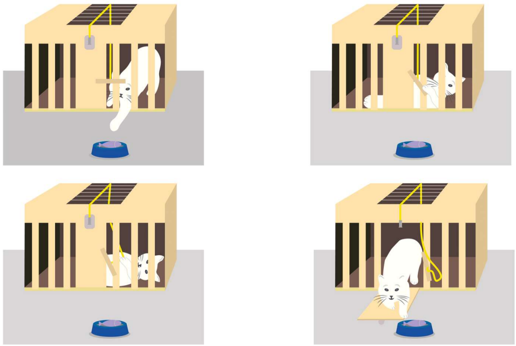


Figure 1.1: Illustration of Thorndike instrumental conditioning experiments taken from Pinterest

Beyond the field of animal psychology, implementing the idea of trial-and-error to train artificial learning agents was promising. In fact, among the earliest thoughts about the possibility of artificial intelligence was the idea of *pleasure-pain system* described by Alan Turing, and which is identical in principal to the idea of the Law of Effect:

> *"When a configuration is reached for which the action is undetermined, a random choice for the missing data is made and the appropriate entry is made in the description, tentatively, and is applied. When a pain stimulus occurs all tentative entries are cancelled, and when a pleasure stimulus occurs they are all made permanent."Turing [1948]*

Recently, the combination of trial-and-error and learning gave birth to RL as a mathematical and computational framework which trains embodied agents to maximize future rewards in an environment. RL leverages the idea of instrumental conditioning: agents perform sequences of actions in an environment, they observe the consequences of these actions in the form of rewards, they learn from these rewards, thus following the Law of Effect.

1.1.2 Formalization

Reinforcement learning problems are often formalized as *Markov Decision Processes* (MDPs). They involve two main aspects. First, an *evaluative aspect* indicating how good the action taken was, but not whether it was the best action possible. These evaluations correspond exactly to the reinforcing signals, typically modeled by a *reward function*. Second, an *associative aspect* enabling the selection of different actions for different situations. These associations describe the strategy of the agents, more formally defined as a *policy function*. Note that the reward function depends on the current situation and the action taken, while the policy function only depends on the current situation. MDPs represent a mathematically idealized formalization of sequential decision making. Since actions influence not only immediate situations but also subsequent ones, MDPs usually require a trade off between immediate and delayed rewards.

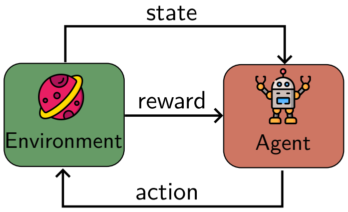


Figure 1.2: Illustration of the agent-environment interaction cycle in reinforcement learning

MDPs as Episodic Interactions. The learning from interaction framed by MDPs involves an *agent* and an *environment* interacting together during a specific amount of time. The interaction is usually *episodic*: time is discretized to a sequence of *time steps*, $t = 0, 1, 2, ..., T$, and there is exactly one interaction by time step. T denotes the maximum number of time steps possibly conducted, and which can be finite or infinite. Notably, many of the ideas of MDPs can be extended to the continuous-time case [Bertsekas and Tsitsiklis, 1996, Doya, 1996]. However, we focus here on the discrete setting. Each interaction between the agent and the environment at each time step involves the following cycle. First, the agent observes its current state and takes an action with its policy function. Second, the environment returns two values: 1) an evaluation of the action taken by the agent; 2) the new state to which the taken action leads. The former is given by the reward function introduced above. The latter is determined by the *dynamics function* of the underlying MDP. See Figure 1.2 for an illustration.

MDPs and the Markov Property. First, we denote by $\mathcal{S}$ the state space. The initial state at time step $t = 0$ is sampled from an *initial state distribution* $s_0 \sim \rho_0(\mathcal{S}) \subset \mathcal{S}$. Second, we denote by $\mathcal{A}$ the action space and $\mathcal{R} : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the reward function. Finally, we denote $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ the state-transition probabilities which define the dynamics function. An MDP $\mathcal{M}$ is exhaustively defined by the tuple of the quantities defined above: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0, \mathcal{R}\}$. By definition, the MDP makes the *Markov assumption*: the transition dynamics to the next state depend only on information about the current state and action, rather than the whole trajectory. More formally, the state-transition probabilities verify the following Markov property:

$$p(s_{t+1} \mid s_0, a_0, s_1, a_1, ..., s_t, a_t) \;=\; p(s_{t+1} \mid s_t, a_t) \tag{1.1}$$

Goals, Cumulative Rewards and Discounted Returns. In its classical format, the reward signal passes from the environment to the agent and determines the *purpose of the agent*. In its simplest case, the purpose of the agent, at time step t, is to maximize the cumulative reward it receives in the long run

$$G_t \;=\; R_{t+1} + R_{t+2} + ... + R_T \;=\; \sum_{i=t+1}^{T} R_i, \tag{1.2}$$

where R_i denotes the reward obtained at time step i and T a finite time horizon. Note that G_t is also called the *return* at time step t. Interestingly, formalizing the idea of goals from the reward signal is one distinctive feature of RL. Consider the example where we want an artificial agent to pick up an object and place it at a certain position (a goal). We might want to define the reward function as zero everywhere, and +1 when the object is at its desired position. In this case, maximizing the cumulative reward is perfectly aligned with achieving the goal. In other words, the reward signal should give the agent information about *what* must be achieved (goal), and not about *how* to achieve (means). However, defining a goal using a single reward function is insufficient if we want agents to learn diversified behaviors — which is the case in open-ended learning. The idea of goals in RL is further developed and formalized in Section 1.3.

In many cases, the interactions between the agent and the environment do not break naturally into identifiable episodes but rather keep going without limit until the agent succeeds or breaks. In this situation, the time horizon verifies $T = \infty$, and the agent-environment interaction withint the MDP is not episodic anymore. As a consequence, the cumulative reward in Equation 1.2 becomes divergent. To circumvent this convergence issue, an additional concept of *discounting* is introduced. The main idea is to value the rewards that are closer in time more than the farther ones. In practice, this is done by introducing a discounting parameter γ. To ensure the convergence of the infinite series, this parameter is usually taken within the interval $[0, 1)$. In this situations, agents maximize the following discounted return

$$G_t \;=\; R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \;=\; \sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \leq \frac{M}{1-\gamma}, \tag{1.3}$$

where M is an upper bounded for the reward values.

1.1.3 Taxonomy of RL Approaches

Throughout the development of RL, many themes have developed and evolved, resulting in the grouping of various RL algorithms into different categories. In this section, we focus specifically on the themes that directly impact our work.

Action-Value Methods and Policy Gradient Methods. Most of RL problems require building complex models and training them using an optimization method.

There are mainly two families of optimization methods for RL: *value-based* and *policy gradient* algorithms. In the former, the idea is to perform as many actions as possible, assess a value to each possible situation and follow the strategy that yields the best results. In the latter, the idea is to maintain a model of policy and tweak it progressively through gradient steps towards actions that produce the best result.

On the one hand, value-based algorithms rely on the definition of *value* and/or *action-value* functions. Both are directly indexed by a policy. The value function of a policy π for a state s is usually denoted $V_\pi(s)$. It estimates the expected return from s when following π. The action-value function of a policy π for a state s and an action a is usually denoted $Q_\pi(s,a)$. It estimates the expected return from state s, taking action a and then following the policy π. Consequently, both the value and the action-value function quantify the fitness of being in a state (the value function) or being in a state and performing an action (the action-value function). A recursive decomposition of these function yields the following *Bellman equations* [Sutton and Barto, 2018]:

$$V_\pi(s) = \mathbb{E}_{\substack{a\sim\pi(s)\\ s'\sim\mathcal{T}(s,a)}}\Big[R(s,a) + \gamma V_\pi(s')\Big] \tag{1.4}$$

$$Q_\pi(s,a) = \mathbb{E}_{\substack{s'\sim\mathcal{T}(s,a)\\ a'\sim\pi(s')}}\Big[R(s,a) + \gamma Q_\pi(s',a')\Big] \tag{1.5}$$

Note that the action-value function and the value function verify the following relation: $V_\pi(s) = \mathbb{E}_{a\sim\pi(s)}Q_\pi(s,a)$. The objective of a value-based algorithm is to determine the optimal value function and/or the optimal action-value function such that

$$V^*(s) = \max_\pi V^*_\pi(s),$$

$$Q^*(s,a) = \max_\pi Q^*_\pi(s,a).$$

The optimal policy can easily be derived from the optimal action-value function, by simply ensuring to take actions that always maximize the action-value function:

$$\pi^*(s) = \arg\max_a Q^*(s,a). \tag{1.6}$$

On the other hand, while value-based methods require spanning the entire action space to evaluate the action-value function for a given state and take the action that maximizes it (see Equation 1.6), policy gradient methods directly model a policy that produces actions for a given state. This makes policy-gradient methods more useful in continuous setups, where spanning the entire action spaces for every state is usually intractable. We denote by $\theta \in \mathbb{R}^d$ the parameter vector of a policy π. The idea behind policy gradient methods is to consider some scalar performance measure $J(\theta)$ with respect to the policy parameter. These methods aim at maximizing this performance metric by updating the parameters at every time step t using the gradient of the performance measure with reference to the parameters before update. Formally:

$$\theta_{t+1} \;=\; \theta_t + \alpha \; \nabla J(\theta_t),$$

where α and $\nabla J(\theta_t)$ correspond respectively to a learning rate and the gradient of the performance measure. In the episodic case, the performance measure is usually defined with reference to the true value function for the current policy starting from an initial state. The *policy gradient theorem* gives a straightforward formulation of the gradient of such a performance measure [Sutton and Barto, 2018].

A particular sub-family of policy gradient methods is called the *actor-critic* family, which simultaneously learns approximations to both the policy and the value function.

On-Policy and Off-Policy. The exploration-exploitation trade-off in reinforcement learning [Sutton and Barto, 2018] poses a dilemma in the learning control methods: even though they seek to learn action values conditional on subsequent optimal behavior, they need to behave non-optimally in order to explore all actions and find the optimal ones. Hence, the question here is as follows: how can the learning control methods actually learn about the optimal policy while behaving according to another policy? Two learning paradigms are confronted: *on-policy* learning and *off-policy* learning.

On the one hand, on-policy methods attempt to evaluate or improve the policy that is used to make decisions as it learns action values not for the optimal policy, but for a near-optimal one that still explores. These methods are generally soft, meaning that $\pi(a|s) \;>\; 0 \; \forall \; s \in \mathcal{S}$ and $\forall \; a \in \mathcal{A}$, but gradually shifted closer to a deterministic optimal policy.

On the other hand, the off-policy approach consists in using two separate policies, one that is learned about and that converges to the optimal policy — *a target policy* — and one that is more exploratory and is used to generate behavior — *a behavior policy*. The data used to describe the behavior is therefore “off” the target policy. To estimate the expected values under the target policy using samples of trajectories generated with the behavioral policy, one can rely on the *importance sampling* technique [Schulman et al., 2015, 2017, Sutton et al., 1998]. Its main idea is to weigh the returns according to the relative probability of their trajectories occurring under the target and behavior policies.

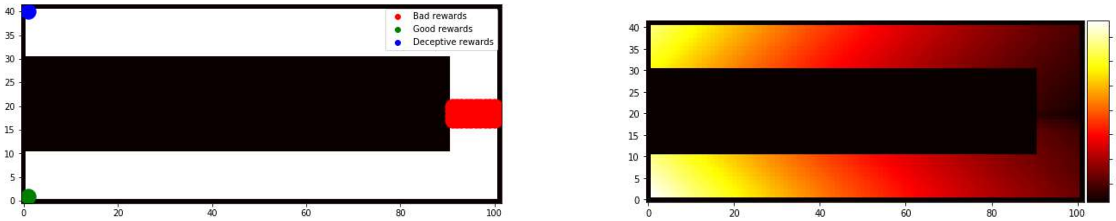
Model-Free and Model-Based. Depending on whether or not the algorithm learns a model of the environment, there exist two classes: *model-based* and *model-free* algorithms. On the one hand, model-based methods make use of knowledge about the environment to improve learning. Agents evolving in such scenarios can for example learn an estimation of the state-transition probabilities, of which they can make use to perform planning, imagining estimated outcomes of their actions even though they have not performed them yet. Another way of incorporating knowledge about the environment is to incorporate some environment-based inductive bias such as constraining the algorithm to produce only valid action or providing the maximum length of the episode. In practice, such algorithms are used especially in games where rules are perfectly defined and where understanding the effects of actions and planning becomes crucial. On the other hand, model-free algorithms can in principle be applied to any problem. They do not directly learn environmental rules, but rather absorb them within their policies through interactions.

All the work conducted within this PhD research uses a model-free, off-policy and policy gradient-based algorithm: Soft Actor-Critic [Haarnoja et al., 2018]. Note that our research is orthogonal to the choice of the RL algorithm.

1.2 Delayed Geometric Discounts for Reinforcement Learning

In the infinite horizon setting, and without further assumptions on the underlying Markov Decision Process (MDP), available RL algorithms learn optimal policies only in the sense of the discounted cumulative rewards presented in Equation 1.3 [Puterman, 2014]. While geometric discounting is well suited to model a termination probability or an exponentially decaying interest in the future, it is not flexible enough to model alternative weighting of the returns. Consider for example settings where the agent is willing to sacrifice short term rewards in favor of the long term outcome. Clearly, for such situations, a discounted optimality criterion is limited and does not describe the actual objective function.

This is particularly true in tasks where agents are willing to sacrifice short term rewards to get good final ones. In such scenarios, agents should be able to tolerate visiting states that provide them with negative rewarding signals if these states are stepping stones to later high rewards. In other words, even if the sum of rewards obtained within a trajectory is not the optimal one, agents would continue performing that trajectory as long as it takes them to the best rewards in the environment. Consider for example the U-maze environment in Figure 1.3a, where the reinforcement signal provides a high reward (+1) when reaching the green dot in the bottom arm, a deceptive reward (+0.9) when reaching the blue dot in the upper arm, and a negative reward (−1) for crossing the red corridor. If the agent is only interested in the long term returns, then the optimal control should always lead to the green dot. However, depending on the initial state, optimal policies in the sense of the discounted RL problem are likely to prefer the deceptive reward due to the exponentially decaying interest in the future (Figure 1.3b). This behavior can obstruct the agent exploration in hard exploration environment, as they would stop exploring the lower part of the maze whenever they start at the upper part, since the region surrounding the adversarial reward is highly repulsive.



(a)

(b)

Figure 1.3: (a) Hard exploration problem example, (b) optimal value function of standard geometrically discounted RL with a discount factor $\gamma = 0.99$.

Naturally, higher discount factors are associated with optimal policies that also optimize the average returns of Equation 1.2, which can solve in principle the described hard exploration problem [Blackwell, 1962]. However, in practice, such discount values can be arbitrarily close to 1 which entails severe computational instabilities. In addition, and particularly in continuous settings or when tasks span over extremely long episodes, discount-based RL approaches are sample-inefficient and are slow at propagating interesting feedback to early states.

In this section, we generalize the geometric discount to derive a variety of alternative time weighting distributions and we investigate the underlying implications of solving the associated RL problem both theoretically and practically. Our contributions in this section are twofold. First, we introduce a novel family that generalizes the geometrically discounted criteria, which we call the *delayed discounted criteria*. Second, we derive tractable solutions for optimal control for both stationary policies using these novel criteria. Finally, we evaluate our methods on hard exploration mazes in both discrete and continuous settings, and on continuous long-episodic control tasks where we show that:

1. Our agents can solve the hard exploration problem in a proof of concept setup.
2. Our methods improve sample-efficiency on continuous robotics tasks compared to Soft-Actor-Critic.

For additional theoretical derivations, proofs and experiments, we refer the reader to Appendix A.

Figure 1.4 showcases how non-geometrically discounted criteria impacts the profile of optimal value function in the U-maze example illustrated in Figure 1.3a.

1.2.1 Reinforcement Learning with Non-Geometric Discounts

In this section, we present our methods. First, we introduce a novel family of parameterized discounting functions that alleviate the attention on immediate rewards. We

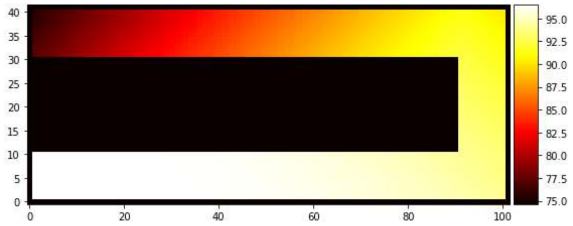


Figure 1.4: Optimal value function of delayed discounted RL criteria in the U-Maze example.

derive the underlying optimization criterion and introduce a novel algorithm that generalizes the Soft Actor-Critic to non-geometric discounts.

Consider an infinite horizon MDP denoted by $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0, \mathcal{R}, \gamma\}$, where the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are either finite or compact subsets of respectively $\mathbb{R}^d$ and $\mathbb{R}^{d'}$, $(d, d') \in \mathbb{N}^2$, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a state transition kernel[1], $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a continuous reward function, $\rho_0 \in \Delta(\mathcal{S})$ an initial state distribution and $\gamma \in [0, 1)$ is the discount factor. A policy π is a mapping indicating, at each time step $t \in \mathbb{N}$, the action a_t to be chosen at the current state s_t. The goal of geometrically discounted reinforcement learning algorithms is to optimize the discounted returns:

$$\mathcal{L}(\pi, \mathcal{R}) := \mathbb{E}_{\pi,\rho_0}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t\Big] \quad ; \quad \mathcal{R}_t := \mathcal{R}(s_t, a_t) \tag{1.7}$$

where $\mathbb{E}_{\pi,\rho_0}$ denotes the expectation over trajectories generated in $\mathcal{M}$ using the policy π and initialized according to ρ_0.

Beyond the Geometric Discount: A Delayed Discounted Criterion

Let's first generalize the optimization criterion from Equation 1.7 to model any discounting function Φ. More formally, we rewrite the discounted return as $G_t = \sum_{t=0}^{\infty} \Phi(t) \times \mathcal{R}_t$. Note that taking $\Phi(t) = \gamma^t$ leads to retrieving exactly the geometrically discounted criterion from Equation 1.7.

We propose to investigate a particular family of parametric discounting functions defined by a sequence of different discount factors $\gamma_0, \gamma_1, ..., \gamma_D$, where $D \in \mathbb{N}$ and the discount factors $\gamma_d \in [0, 1)$ for any integer $d \in [0, D]$. We call D the *delay parameter*. We consider the following parametric discounting function:

$$\Phi_D(t) := \sum_{\substack{\{a_d \in \mathbb{N}\}_{d=0}^{D} \\ \sum_d a_d = t}} \prod_{d=0}^{D} \gamma_d^{a_d}, \tag{1.8}$$

[1] $\Delta(\mathcal{S})$ denotes the set of probability measures over $\mathcal{S}$

and the underlying non-geometrically discounted criterion:

$$\mathcal{L}_D(\pi, r) := \mathbb{E}_{\pi,\rho_0}\Big[\sum_{t=0}^{\infty} \Phi_D(t)\mathcal{R}_t\Big]. \tag{1.9}$$

This class of optimality criteria is a generalization of the classical geometric discount. In fact, we highlight that for $D = 0$, $\Phi_0(t) = \gamma_0^t$ which implies that $\mathcal{L}_0(\pi, \mathcal{R}) = \mathcal{L}(\pi, \mathcal{R})$ for any policy π and any reward function $\mathcal{R}$. In Figure 1.5, we report the normalized distribution of the weights $\Phi_D(t)$ (i.e. $y(t) = \frac{\Phi_D(t)}{\sum_i \Phi_D(i)}$) over time as we vary the delay parameter $D \in \{0, \dots, 9\}$. Notice that for higher values of the delay parameter D, the mode of the probability distribution is shifted towards future time steps. In fact, the delay parameter controls the region of time steps on which to focus more, by putting more weights on that region rather the others.

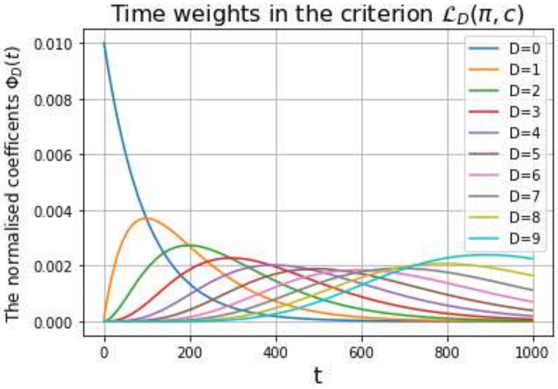


Figure 1.5: The normalized coefficients $\Phi_D(t)$ over time for different values of D

Intuitively, the proposed criterion in Equation 1.9 describes the goal of an agent that discards short term gains in favor of long term discounted returns. In the remainder of this section, we only focus on learning optimal stationary solutions for the delayed discounted criterion. Appendix A is dedicated to further investigations, where we introduce a more general problem formulation using linear combinations of delayed losses to account for the attention over multiple regions of time steps.

Optimal Stationary Policies

In this section, we propose to learn a stationary solution using an algorithmic scheme akin to policy-iteration. As in the classical setting, the goal is to learn an optimal policy π^* that maximizes the state-action value function Q_D^π defined as follows:

$$Q_D^\pi(s, a) := \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \Phi_d(t)\mathcal{R}(s_t, a_t)|s_0, a_0 = s, a\Big]. \tag{1.10}$$

In the geometrically discounted setting, the value function is the fixed point of the Bellman optimality operator. Luckily, this property is also valid for Q^{π}_{D}:

Proposition 1 *For any discount parameters $(\gamma_d)_{d=0}^{D}$, the value functions Q^{π}_{D} is the unique fixed point of the following γ_D-contraction:*

$$[T^{D}_{\pi}(q)](s,a) = \underbrace{\mathbb{E}_{\substack{s'\sim\mathcal{T}(s,a)\\ a'\sim\pi(s')}}\Big[\mathcal{R}(s,a) + \sum_{d=0}^{D-1}\gamma_d Q^{\pi}_{d}(s',a')\Big]}_{:=\,\mathcal{R}^{\pi}_{D}(s,a)} + \gamma_D \mathbb{E}_{\substack{s'\sim\mathcal{T}(s,a)\\ a'\sim\pi(s')}}\Big[q(s',a')\Big]. \tag{1.11}$$

Proposition 1 suggests that the state-action value function Q^{π}_{D} with respect to the delayed criterion $\mathcal{L}_D$ respects the following equation:

$$Q^{\pi}_{D}(s,a) = \mathcal{R}^{\pi}_{D}(s,a) + \gamma_D \mathbb{E}_{\substack{s'\sim\mathcal{T}(s,a)\\ a'\sim\pi(s')}}\Big[Q^{\pi}_{D}(s',a')\Big]. \tag{1.12}$$

In other words, Q^{π}_{D} can be seen as the state-action value function using an augmented reward signal $\mathcal{R}^{\pi}_{D}(s,a)$ parameterized by the delay parameter D and dependent on the policy π with reference to the γ_D-discounted returns. Intuitively, the instantaneous worth of an action (the term $\mathcal{R}^{\pi}_{D}$) is the sum of the environments' myopic returns (the term $\mathcal{R}(s,a)$) and the long term evaluations (with lower delay parameters $(Q^{\pi}_{d})_{d<D}$).

This has the beneficial side-effect of enhancing sample efficiency as it helps the agent to rapidly back-propagate long-term feedback to early states. It reduces the time needed to distinguish good from bad behaviors, particularly in continuous settings where function approximations are typically used to learn policies. This is discussed in detail in Section 1.2.3.

Similarly to standard value-based RL algorithms, we choose to model the state-action value function and the policy with neural network approximators. On the one hand, we parameterize Q^{π}_{D} with θ_D. Since the computation of the state-action value function Q^{π}_{D} requires the evaluation of Q^{π}_{d}, where $d \in [0, D-1]$ (see Proposition 1), we parameterize each of these functions by θ_d, where $d \in [0, D-1]$. Consequently, by contrast with standard value-based algorithms, our methods use $D+1$ approximators, one for each value of the delay. On the other hand, we parameterize the policy π with ϕ.

As for standard value based methods, we consider a set of trajectories $\mathcal{D} = \{s, a, s'\}$ stored in an experience replay buffer. We update the Q-values by optimizing a performance measure $J^{Q}_{D}(\theta)$:

$$J^{Q}_{D}(\theta) = \mathbb{E}_{\substack{s,a,s'\sim\mathcal{D}\\ a'\sim\pi_{\phi}(s')}}\Big[\frac{1}{2}\big(Q_{\theta} - (\mathcal{R}(s,a) + \sum_{d=0}^{D}\gamma_d Q_{\bar{\theta}_d}(s',a'))^2\big)\Big], \tag{1.13}$$

where $\bar{\theta}_d$ denotes the parameters of the target state-action value functions. As for the policy update step, inspired from the Soft-Actor-Critic (SAC) algorithm [Haarnoja et al., 2018], we propose to optimize an entropy regularized soft Q-value using the following loss where α is the learning rate parameter:

$$J_D^{\pi}(\phi) = -\mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\phi}\Big[Q_{\theta_D}(s,a) - \alpha\log(\pi_\phi(a|s))\Big]. \tag{1.14}$$

We use Equations 1.13 and 1.14 to construct Algorithm 1, which we call the *Generalized Soft Actor-Critic* algorithm (G-SAC). It generalizes the SAC algorithm that approximates optimal stationary policies in the sense of $\mathcal{L}_D$. In practice, this can be further improved using the double Q-network trick and the automatic tuning of the regularization parameter α. This is discussed in Appendix A. Besides, since using additional Q-networks for each delay level increases the number of parameters — which increases the update time by iteration — we share the lower layers between all the critics and use a different head for each one. Unfortunately, unlike the geometrically discounted setting, the policy improvement theorem is no longer guaranteed in the sense of $\mathcal{L}_D$. This means that depending on the initialization parameters, Algorithm 1 can either converge to the optimal stationary control or get stuck in a loop of sub-optimal policies. This is discussed in detail in Section 1.2.3.

Algorithm 1 Generalized Soft Actor Critic

```
Input: initial parameters $(\theta_d)_{d=0}^{D}$, $\phi$, learning rates $(\lambda_d)_{d=0}^{D}$, $\lambda_\pi$, polyak parameter $\tau$
initialize target networks $\bar{\theta}_d \leftarrow \theta_d$ and replay buffer $\mathcal{D} \leftarrow \emptyset$
for each iteration do
    for each environment step do
        $a_t \sim \pi_\phi(s_t),\ s_{t+1} \sim \mathcal{P}(s_t, a_t),\ \mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, s_{t+1}, \mathcal{R}_t)\}$
    for $d \in [0, D]$ do
        for each $Q_d$ gradient step do
            update parameter $\theta_d \leftarrow \theta_d - \lambda_d \hat{\nabla}_{\theta_d} J_d^{Q}(\theta_d)$
            update target $\bar{\theta}_d \leftarrow \tau\bar{\theta}_d + (1-\tau)\theta_d$
    for each policy update do
        update policy $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_D^{\pi}(\phi)$
Return: $\pi_\phi, (Q_{\bar{\theta}_d})_{d=0}^{D}$
```

1.2.2 Related Work

Bridging the Gap Between Discounted and Average Rewards Criteria

It is well known that defining optimality with respect to the cumulative discounted reward criterion induces a built-in bias against policies with longer mixing times. In fact, due to the exponential decay of future returns, the contribution of the behaviour from the T^{th} observation up to infinity is scaled down by a factor of the order of γ^T. In the literature, the standard approach to avoid this downfall is to define optimality with respect to the average reward criterion $\bar{\mathcal{L}}$ defined as :

$$\bar{\mathcal{L}}(\pi, r) := \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_{\pi,p_0}\Big[\sum_{t=0}^{T} r_t\Big]. \tag{1.15}$$

This setting as well as dynamic programming algorithms for finding the optimal average return policies have been long studied in the literature [Howard, 1960, Veinott, 1966, Blackwell, 1962, Puterman, 2014]. Several value based approaches [Schwartz, 1993, Abounadi et al., 2001, Wei et al., 2020] as well as policy based ones [Kakade, 2001, Baxter and Bartlett, 2001] have been investigated to solve this problem. These approaches are limited in the sense that they require particular MDP structures to enjoy theoretical guarantees.

Another line of research is based on the existence of a critical discount factor $\gamma_{crit} < 1$ such that for any discount $\gamma \in (\gamma_{crit}, 1)$ the optimal policy in the sense of the γ-discounted criterion also optimizes the average returns [Blackwell, 1962]. Unfortunately, this critical value can be arbitrarily close to 1 which induces computational instabilities in practice. For this reason, previous works attempted to mitigate this issue by increasing the discount factor during training [Prokhorov and Wunsch, 1997], learning higher-discount solution via learning a sequence of lower-discount value functions [Romoff et al., 2019] or tweaking the reinforcement signal to equivalently learn the optimal policy using lower discounts [Tessler and Mannor, 2020].

Exploration Strategies

Another line of research attempted to tackle the hard exploration problem by further driving the exploration of agents towards interesting states. Inspired by intrinsic motivation in psychology [Oudeyer and Kaplan, 2008], some approaches train policies with rewards composed of extrinsic and intrinsic terms. Namely, count-based exploration methods keep track of the agents' past experience and aim at guiding them towards rarely visited states rather than common ones [Bellemare et al., 2016, Colas et al., 2019]. Alternatively, prediction-based exploration defines the intrinsic rewards with respect to the agents' familiarity with their environment by estimating the accuracy of a model predicting the dynamics of the environment [Stadie et al., 2015, Pathak et al., 2019]. Other approaches maintain a memory of interesting states [Ecoffet et al., 2019, 2021], trajectories [Guo et al., 2020] or goals [Guo and Brunskill, 2019]. Ecoffet et al. [2019, 2021] first return to interesting states using either a deterministic simulator or a goal-conditioned policy and start exploration from there. [Guo et al., 2020] train a trajectory-based policy to rather prefer trajectories that end with rare states. Guo and Brunskill [2019] revisit goals that have higher uncertainty. Finally, based on the options framework [Sutton et al., 1999b], options-based exploration aims at learning policies with termination conditions—or macro-actions. Leveraging these abstract actions helps driving the agents' exploration towards behaviours of interest [Gregor et al., 2016, Achiam et al., 2017a].

1.2.3 Experiments

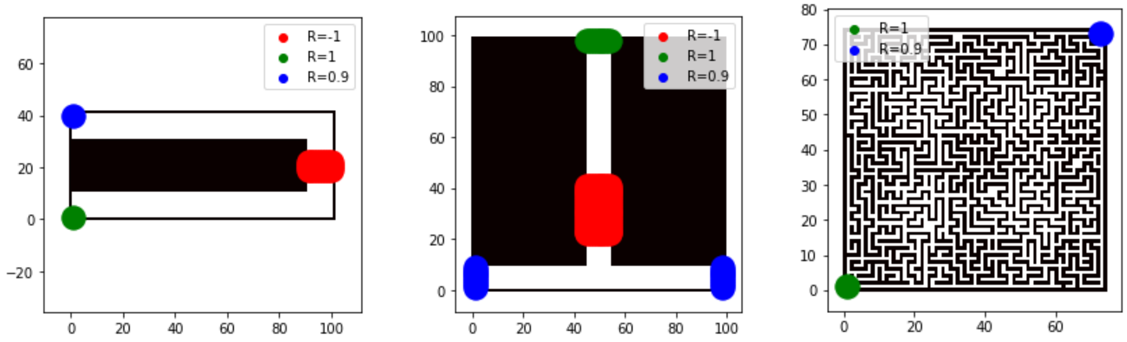
In this section, we present the experimental setup used to evaluate our methods. Our goal is to investigate the impact of the delayed discounting criterion. To do so, we focus on the following questions:

- How does optimizing the delayed discounting criterion impacts the performance metrics of the agents in hard exploration problems where artificial agents need to handle both deceptive and adversarial rewarding signals?
- How does the proposed G-SAC algorithm impact the performance metrics of artificial agents in classic continuous control problems with dense rewarding signals?

To answer these questions, we first describe the environments we considered in this study. Then, we introduce the evaluation metrics. Finally, we present the underlying results.

Environments

To answer the scientific questions highlighted in the introduction of this section, we consider two types of environments: *hard exploration environments* with deceptive, adversarial and sparse rewarding signals and *classic control environments* with dense rewarding signals.



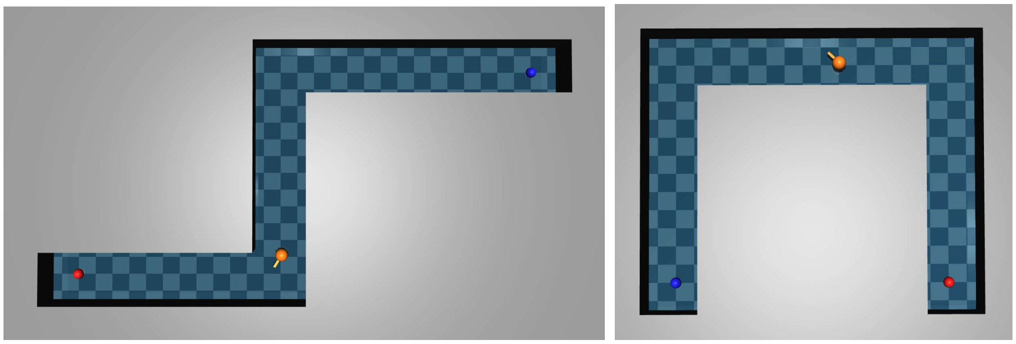
(a) U-maze

(b) T-maze

(c) Random maze

Figure 1.6: Hard exploration environments

Hard Exploration Discrete Mazes with Deceptive and Adversarial Rewards. This family of environments consists of three discrete mazes illustrated in Figure 1.6. In all these problems, the state space is discretized to correspond to different cells of the mazes and the action space corresponds to moving in four directions (right, left, up, down). In the U-Maze and the T-Maze environments (Figures 1.6a and 1.6b), we use three different types of rewards: a *deceptive reward* of 0.9 represented by the blue zone; an *adversarial reward* of -1 represented by the red zone; a *target reward* of +1 represented by the green zone. In the Random Maze environment (Figure 1.6c), we discard the adversarial reward for simplicity and only focus on the deceptive and target rewards. In all these environments, we want the artificial agents to sacrifice short term rewards for higher long term ones.



(a) SMaze-v0

(b) UMaze-v0

Figure 1.7: Continuous Maze Environments

Hard Exploration Continuous Mazes with Deceptive Rewards. This family of environments consists of two continuous mazes illustrated in Figure 1.7. In both environments, the agent is a point whose action space is 2-dimensional, corresponding to movements along the x and y axis. The state space contains features corresponding to the positions and the velocities of the artificial agent. In both environments (SMaze-v0 in Figure 1.7a and Umaze-v0 in Figure 1.7b), we consider two types of rewards: a *deceptive reward* of +0.8 (blue dot) and a *target reward* of +1 (red dot).

Simulated Continuous Control Benchmarks. We aim at evaluating the performance of artificial agents using the delayed discounting criterion on several continuous control domains. To this end, we consider 5 different environments within the MUJOCO physics engine [Todorov et al., 2012]: Walker2d, Hopper, HalfCheetah, Ant and Humanoid. In all these environments, the rewarding signal is dense and consists of a sum of control term (penalizing falling agents) and a running term (rewarding higher speed to ensure going longer distances).

Evaluation Metrics

Hard Exploration Discrete Mazes. We investigate the ability of G-SAC to approximate the solution using the discounting function phi_D. To this end, we consider different values for the delay parameter $D \in [0, 1, 3, 4, 7, 8]$. For each level d of delay, we use a discounting factor $\gamma_d = 0.99 - \frac{i}{1000}$. For each value of the delay, we consider two different agents: Φ_D-Random, where weights of the policies and value function are randomly initialized; Φ_D-Optimal, where weights are initialized as optimal solutions of the geometrically discounted criterion (for $D = 0$). We evaluate our agents by investigating their average rewards after training. Values are averaged over 25 runs using trajectories of length 4000, where agents are uniformly reset in any state of the mazes.

Hard Exploration Continuous Mazes. We assess the capabilities of G-SAC agents in distinguishing the target rewards from the deceptive ones. To this end, we investigate if these agents are able to reach the target reward even though they were initialized close to the deceptive ones. We evaluate the average rewards obtained after training G-SAC agents with a delay parameter fixed at $D = 1$. We discretize the continuous mazes into different cells, initialize our agents in each cell and compute the associated performance metrics. We confront our agents to standard ones trained with SAC (with geometrically discounted criteria). Experiments were conducted over 5 seeds

Simulated Continuous Benchmarks. For each of the considered continuous benchmarks, we fix the delay parameter $D = 1$. This suggests that two discount factors are used: γ_1 and γ_2. We are interested in the average rewards obtained during training. We compare G-SAC agents to standard SAC agents.

Results

In the next paragraphs, we present the results and key insights for the different environments considered in our experimental setup.

Table 1.1: Average rewards in the discrete maze environments for different values of delay D. See Appendix A for additional results.

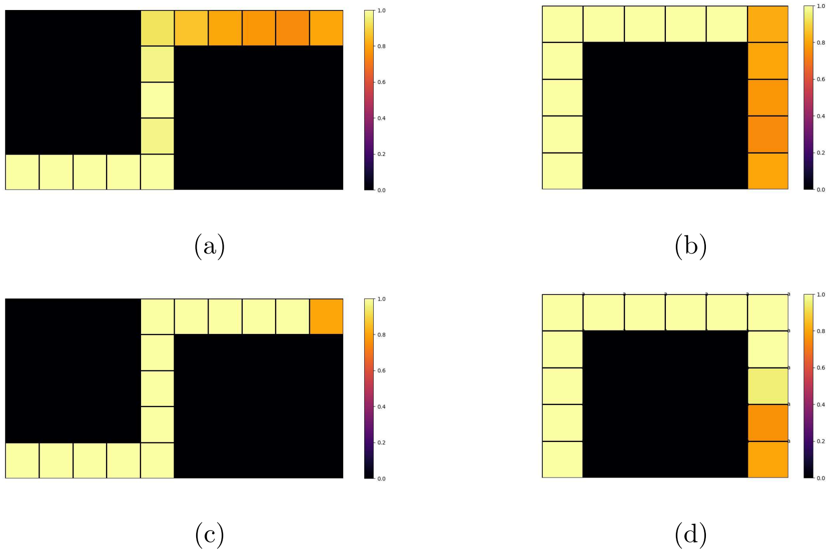
Delay Value (D)	0	1	3	4	7	8
Discrete U-Maze						
Φ_D-Random	0.92 ± 0.02	0.93 ± 0.03	**0.96 ± 0.01**	0.93 ± 0.04	0.6 ± 0.12	0.05 ± 0.3
Φ_D-Optimal	0.92 ± 0.02	0.93 ± 0.01	0.96 ± 0.02	**0.98 ± 0.01**	0.87 ± 0.08	0.4 ± 0.2
Discrete T-Maze						
Φ_D-Random	0.92 ± 0.02	0.93 ± 0.02	**0.99 ± 0.01**	**0.98 ± 0.01**	0.7 ± 0.09	0.22 ± 0.4
Φ_D-Optimal	0.92 ± 0.02	0.93 ± 0.02	**0.99± 0.01**	**0.98 ± 0.01**	0.02 ± 0.5	0.05 ± 0.4
Discrete Random-Maze						
Φ_D-Random	0.93 ± 0.01	0.93 ± 0.02	0.89 ± 0.04	0.35 ± 0.17	0.45 ± 0.2	0.05 ± 0.3
Φ_D-Optimal	0.93 ± 0.01	0.93 ± 0.02	**0.98 ± 0.01**	**0.98 ± 0.01**	0.9 ± 0.04	0.01 ± 0.2

Hard Exploration Discrete Mazes with Deceptive and Adversarial Rewards. Table 1.1 depicts the average rewards on the different discrete maze environments for different values of the delay parameter D.

As discussed earlier, unlike the geometrically discounted setting, the policy update is not guaranteed to improve performance. For this reason, depending on the initialisation, the algorithm can either converge to the optimal stationary policy, or get stuck in a sequence of sub-optimal policies.

A common observation is that for a depth D of 7 or higher, the algorithm is unstable and we couldn't learn a good stationary policy in a reliable way. However, the learned stationary policies with even a relatively shallow depth parameter yielded reliable policies. Notice how the baseline ($D = 0$, i.e. the geometrically discounted case) always under-performs when compared to the learned policies for a depth parameter of 3 and 4.

We also observe that the algorithm is sensitive to the used initialization. Using the optimal policy in the sense of the geometrically discounted objective (Φ_D-Optimal agents) helps stabilize the learning procedure in most cases: this is particularly true in the random maze environment where Φ_D-Random yields bad performance metrics even with a low depth parameter. This aligns with what was discussed in earlier sections about the lack of guarantee of improvement through policy updates and possible convergence to sub-optimal policies



(a)

(b)

(c)

(d)

Figure 1.8: Grid plot of the average rewards per cell initialization for SAC within (a) the *SMaze-v0* environment, (b) the *UMaze-v0* environment; G-SAC within (c) the *SMaze-v0* environment and (d) the *UMaze-v0* environment

Hard Exploration Continuous Mazes with Deceptive Rewards. The grid plot on Figure 1.8 highlights the average rewards obtained by the agents when initialized in different cells. Depending on the cell in which they were initialized, both agents choose to opt either for the deceptive or the true reward. However, the G-SAC agent can choose the target reward even if it was initialized close to the deceptive one (see Figures 1.8c and 1.8d). Meanwhile, the SAC agent is more myopic, as the number of cells that leads it to the deceptive rewards are more than the ones encountered in G-SAC (see Figures 1.8a and 1.8b).

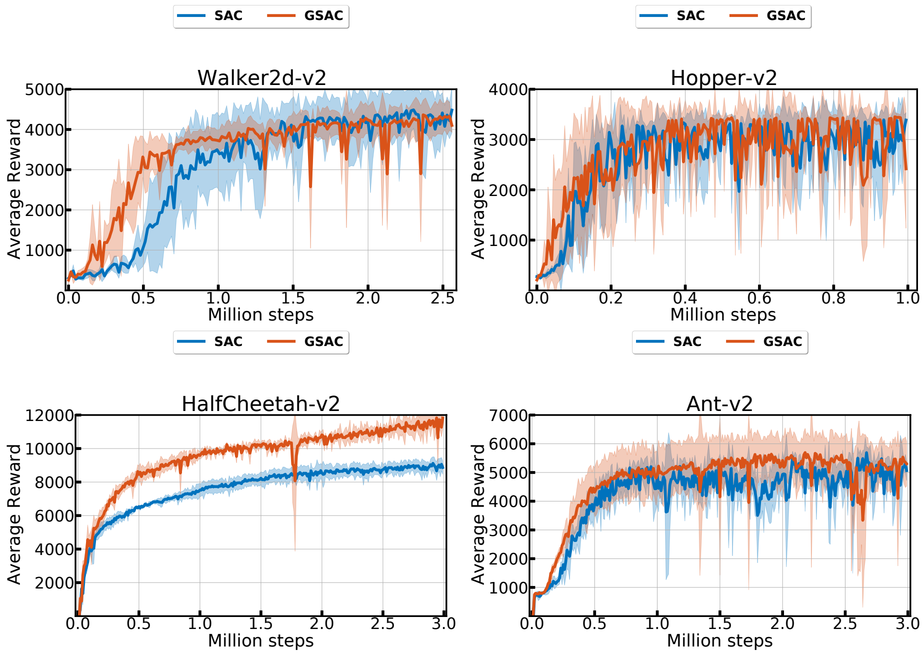


Figure 1.9: Training curves on continuous control benchmarks. Generalized Soft-Actor-Critic (GSAC) shows better sample efficiency across all tasks

Simulated Continuous Control Benchmarks In Figure 1.9, we report the average rewards obtained by the different agents over time in continuous control domains. Note that in all environments, the G-SAC agents are faster in collecting positive rewards than the SAC agents. The main reason behind this is that the critic tends to discern good from bad actions in the G-SAC agents *faster* than the SAC agents. In fact, on the one hand, the SAC agents spend more time uncertain about the quality of their actions for a given state, and thus need more time and more experience to make the estimations of their critic a more accurate estimate of the true long-term value of their actions. On the other hand, in the G-SAC agents, the fact that there is a separate critic for each level of delay (Q_{θ_0} and Q_{θ_1}) implies that estimations of action values get bootstrapped. Thus, G-SAC agents are able to distinguish faster the good from the bad actions, and as a result are more sample efficient.

1.2.4 Conclusion

Designing intelligent autonomous agents requires — among other things — the ability to discard short term returns in favor of long term outcomes. Unfortunately, existing formulations of the reinforcement learning problem stand on the premise of either discounted or average returns: both providing a monotonic weighting of the rewards over time.

In this chapter, we have proposed a family of delayed discounted objective functions that captures a wide range of non-monotonic time-preference models. We analyzed the new formulation to construct the Bellman optimality criterion of stationary solution. The derived algorithms successfully solved tabular hard exploration problems and out-performed the sample efficiency of SAC in various continuous control problems.

The ability to sacrifice short term rewards for long term higher ones is a key property in open-ended learning. In fact, continuously growing a repertoire of skills requires overcoming the local minima problem, where artificial agents learn suboptimal policy that can maximize the optimized criterion but still does not align with the task at hand. However, reinforcement learning alone is not sufficient for learning policies that can simultaneously accomplish multiple tasks. In the next section, we introduce the framework of goal-conditioned reinforcement learning, which constructs a background for learning contextual policies. In the remaining of the document, we use standard RL methods as a backbone for learning multiple goals.

1.3 Goal-Conditioned Reinforcement Learning

In Section 1.1, we defined Reinforcement Learning as a framework where embodied agents interact in an environment by taking actions and receiving rewarding signals. In this standard setup, we defined the purpose of such embodied agents — or their goal — with reference to the rewards they receive: the goal of reinforcement learners is to maximize the cumulative reward. In the case of sparse reward signals, where agents only receive a positive feedback when they fulfill a particular task, maximizing the cumulative reward should perfectly align with accomplishing the task. In other words, the rewarding signal, engineered beforehand, pushes the agent towards a state where the task is achieved.

This standard setup is not suitable for training artificial agents to achieve a set of different goals, since there is usually a unique mapping between a goal and a reward signal. A straightforward way to circumvent this issue is to define *goal experts* modules. This implies that an embodied agent would have a set of policies equal to the number of potentially learnable goals. Whenever the agent attempts to reach a particular goal, it selects actions according to the policy that corresponds to this goal. These methods defined the first attempts to solve multi-goal problems [Kaelbling, 1993, Baranes and Oudeyer, 2013a], some of which used modular representations of the state space [Forestier and Oudeyer, 2016a]. Unfortunately, all these methods present two main drawbacks. First, they all require knowing the number of goals beforehand in order to define the number of policies to be trained. Second, they do not leverage generalization and transfer between goals, since all the policies are by definition independent from one another.

Recently, with the promising results leveraged by neural networks as universal function approximators, a new framework where a single policy could learn to achieve multiple goals has been developed. This defines the sub-family of *Goal-Conditioned Reinforcement Learning* (GC-RL), which originated from results on universal value function approximators [Schaul et al., 2015]. The main principle is simply to condition the agent's policy not only on observations or states, but also on embeddings of the goals to be achieved. Instead of having one policy for each goal, these methods have a single *contextual* policy, where the context defines the goal [Andrychowicz et al., 2017, Colas et al., 2019, Akakzia et al., 2020b].

Learning to master multiple goals simultaneously is not the only scenario in which GC-RL can be used. In fact, training a goal-conditioned policy can facilitate the acquisition of complex skills by enabling agents to decompose the long-term goals into intermediate sub-goals that are usually easier to reach. This particular scenario is interesting in the context of open-ended learning, where agents need to continuously learn an unbounded set of skills. By learning to decompose hard goals, or to bootstrap easier goals, such agents can progressively grow their repertoires of skills. Nevertheless, the fashion by which goals are represented is crucial in order to accomplish this endeavour.

In this section, we present basic challenges and practical solutions towards training goal-conditioned agents. First, we start with an overview of the notion of goals in psychology and introduce a subsequent universal definition used in the field of Artificial Intelligence. Second, we formalize the problem of learning multiple goals as a reinforcement learning problem where standard MDPs are augmented with a goal space, and where reward functions correspond to goal achievement functions, which are by definition dependent on the running goal. Finally, we investigate some of the methods that attempted to tackle the multi-goal problem and introduce a specific typology based on the phase on which these methods focus.

1.3.1 Goals in Reinforcement Learning

As mentioned in the introduction of this section and in Section 1.1, the idea of giving a *purpose* to artificial agents is strongly related to the definition of the *rewarding signal*: fulfilling its purpose should align with maximizing the rewarding signal. In standard RL, this hypothesis is known as *the reward hypothesis*:

> *"[...] all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward)." [Sutton and Barto, 2018]*

Seen under the scope of standard RL, the reward hypothesis seems to be closely related to what psychologists call *habitual behavior*: a pattern triggered by appropriate stimuli and then performed more-or-less automatically [Sutton and Barto, 2018]. In fact, the reward hypothesis suggests that goals should not be known distinctively when interacting with the environment. It is only the situation of the agent at a certain time step that perfectly determines its behavior. In other words, whenever the agent is in this situation, it would most likely take the same actions in an *habitual fashion*. Psychologists qualify habits as behaviors that are controlled by antecedent stimuli (the current state). By contrast to habitual behavior, *goal-directed behavior* is purposeful and is controlled by its consequences (the next state, a goal achievement function) [Dickenson et al., 1980, 1985].

We define a goal as a couple $g = (z_g, R_g)$, where z_g denotes the goal embedding and R_g corresponds to a fitting function parameterized by g that assesses the achievement or not of the underlying goal [Forestier and Oudeyer, 2016a]. This fitting function is also known as the *goal achievement function* [Colas et al., 2022b]. It aims at measuring the agent's progress towards a particular goal. The set of goal-achievement functions associated with a set of goals can be represented either as a single goal-conditioned parameterized function (example, a neural network conditioned by the goal), or a set of predefined goal-conditioned reward functions $(R_{\mathcal{G}}(.|z_g) = R_g(.)$, where $\mathcal{G}$ defines the goal space.

The goal embeddings can be seen as the manner through which artificial agents represent and reason about their goals. Depending on how these representations are fashioned, agents are more likely to develop different behaviors. In fact, some goal representations leverage a topology that is more structured than others, facilitating their mapping to states and actions, promoting the transfer between different goals and potentially enabling a more efficient scaffolding. Many approaches from the literature considered different goal representations. Some works considered goals as choices between multiple objectives, where the embeddings are usually one-hot encodings of the objective to be considered [Oh et al., 2017, Mankowitz et al., 2018, Codevilla et al., 2018]. Other approaches used specific targets of states to define the goals [Andrychowicz et al., 2017, Nair et al., 2018, Plappert et al., 2018, Colas et al., 2019, Fournier et al., 2019, Lanier et al., 2019, Li et al., 2019]. Another line of research considered language instructions to define goals. In this scenario, sentences in natural language define a specific set of constraints that need to be verified in order for the goal to be accomplished [Bahdanau et al., 2019, Colas et al., 2020b, Lynch and Sermanet, 2020]. Finally, some recent works modeled goals as abstract binary problems. This approach enables the discretization of the state representations, thus defining a finite goal space. This discretization can be based on some semantic features based on relational predicates [Tellex et al., 2011, Alomari et al., 2017, Bapst et al., 2019, Akakzia et al., 2020a]. For a more detailed typology of goal representations, we refer the reader to Colas et al. [2022b].

1.3.2 Formalizing Multi-Goal Reinforcement Learning Problems

In this section, we formalize multi-goal reinforcement learning problems. While standard RL uses a single MDP (see Section 1.1) and requires the agent to finish one specific task defined by the reward function, GC-RL focuses on a more general and more complex scenario where agents can fulfill multiple tasks simultaneously. To tackle such a challenge, we introduce a goal space $\mathcal{G} = Z_{\mathcal{G}} \times R_{\mathcal{G}}$, where $Z_{\mathcal{G}}$ denotes the space of goal embeddings and $R_{\mathcal{G}}$ is the space of the corresponding reward functions. We also introduce a tractable mapping function $\phi : \mathcal{S} \rightarrow Z_{\mathcal{G}}$ that maps the state to a specific goal embedding. The term *goal* should be differentiated from the term *task*, which refers to a particular MDP instance. Next, we need to differentiate the notions of *desired goal* and *achieved goal*.

- **Achieved Goal:** An achieved goal defines the outcome of the actions conducted by the agent during a rollout episode. More specifically, it is the output of the mapping function applied at time step t on the current state of the agent: $\phi(s_t)$. We denote by $p^a_{\mathcal{G}}$ the distribution of achieved goals. Note that these goals are exactly the goals *discovered* by the agent in play.
- **Desired Goal:** A desired goal defines the task that the agent attempts to solve. It can be either provided externally (by a simulator or an external instructing program) or generated intrinsically. We denote by $p^d_{\mathcal{G}}$ the distribution of desired goals. This distribution is predefined when the agent receives goals from its external world, and corresponds to the distribution of achieved goals if the agent is intrinsically motivated.

Based on these definitions, we extend RL problems to handle multiple goals by defining an augmented MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0, \mathcal{G}, p^d_{\mathcal{G}}, \phi\}$. Consequently, the objective of GC-RL is to learn a goal-conditioned policy $\pi : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow [0, 1]$ that maximizes the expectation of the cumulative reward over the distribution of desired goals:

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{\substack{g \sim p^d_{\mathcal{G}},\ s_0 \sim \rho_0 \\ a_t \sim \pi(. \mid s_t, g) \\ s_{t+1} \sim \mathcal{T}(s_t, a_t)}} \Big[\sum_t \gamma^t R_{\mathcal{G}}(\phi(s_{t+1}) \mid z_g) \Big]. \tag{1.16}$$

1.3.3 Typology of GCRL Algorithms

In standard GC-RL, the agent goes through 3 main phases. First, it *generates a goal to pursue* and performs a rollout in the environment. As mentioned in the previous section, this goal is sampled from the distribution of desired goals $p^d_{\mathcal{G}}$. Second, it *replays trajectories* which are usually stored in a replay buffer. Third, it *optimizes its policy* using the relabeled historical data. Based on these phases, we consider a categorization of GC-RL algorithms. This categorization is illustrated in Figure 1.10. We refer the reader to Liu et al. [2022] for a more detailed survey.

Goal Generation. Algorithms that investigate the goal generation process within goal-conditioned agents mainly rely on the idea of biasing the distribution of sampled goals in a way that either improves learning or fosters exploration. On the one hand, a line of research focuses on designing artificial goal-conditioned agents that automatically generate goals of *intermediate difficulty*. These goals are neither too hard nor too easy, they lie in the frontier of what these agents can autonomously accomplish. These methods rely on trained generative models [Florensa et al., 2018], prioritizing goals that show the highest disagreement based on an ensemble of trained Q-functions [Zhang et al., 2020], or use estimation of the learning progress to push the goal generation towards goals where the learning progress is maximal [Colas et al., 2019, Akakzia et al., 2020b]. On the other hand, other approaches focus on *goal diversity*. In fact, the sparsity of the reward signal makes some goals harder to reach than others. These methods focus on sampling these goals more often, which is shown to lead to better exploration capabilities. Finally, another line of research benefits from external signals to *learn from experts*. A sub-family of these methods uses expert demonstrations to extract checkpoints within the given trajectories and train goal-conditioned policies to chain goals [Paul et al., 2019, Chenu et al., 2022]. Another sub-family relies on socially-suggested by an external social partner whose aim is to guide the agent exploration [Akakzia et al., 2022].

Relabeling. As opposed to goal generation methods which bias the distribution of sampled goals, relabeling methods replace the history data in the replay buffer before training the agent. We consider two main lines of research that use this technique. On the one hand, *hindsight relabeling* uses the achieved goals in a given trajectory to replace the desired goals [Andrychowicz et al., 2017]. This method alleviates the sparse reward problem by focusing on learning from failures, assuming that the learned behavior would generalize to the distribution of unseen goals. This technique is widely used as the backbone of many related works [Fang et al., 2019, Lanier et al., 2019, Pitis et al., 2020, Eysenbach et al., 2020, Akakzia and Sigaud, 2022]. On the other hand, *foresight relabeling* does not focus on the historical data from the experience buffer, but rather performs planning based on current situations. This technique is put in practice by learning dynamics models of the environment and generating virtual trajectories for relabeling.

Optimization. Methods that focus on the optimization procedure within GC-RL problems train a goal-conditioned policy on a set of goals. These methods mainly use the *universal value function approximation* (UVFA) [Schaul et al., 2015], which generalizes the standard value function to a goal-conditioned value function. A line of research within this procedure attempts to alleviate the sparse rewards problem by reshaping the reinforcing signal [Wu et al., 2018, Trott et al., 2019, Hartikainen et al., 2019, Durugkar et al., 2021, McCarthy and Redmond, 2021]. These methods go beyond the distance-based reshaping, which is shown to lead to additional local minima [Liu et al., 2022]. Another sub-line of research uses the idea of *self-imitation learning*, which treats any performed trajectory as a successful trial for reaching its final achieved state. The methods learn the goal-conditioned policy by maximizing the likelihood of actions for a reached goal within a number of steps [Ghosh et al., 2019]. Another method is to filter actions from the performed demonstrations based on their corresponding Q-values and use a supervised auxiliary loss on the RL objective to improve the learning efficiency.

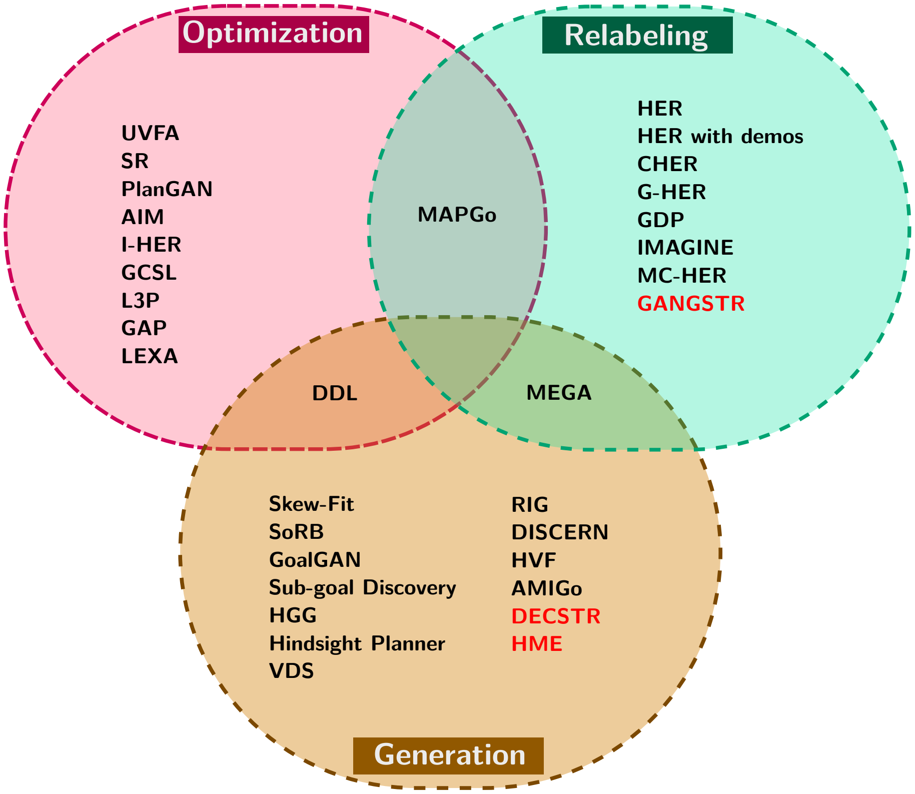


Figure 1.10: Typology of Goal-Conditioned RL Algorithms adapted from Liu et al. [2022]. Our contributions are indicated in red. See Appendix F for the references of the labels.

1.4 Goal Exploration Processes

In multi-goal setups, the objective of goal-conditioned artificial agents is to simultaneously learn as many goals as possible. In other words, the training of such agents should in principle yield optimal goal-conditioned policies that maximize the coverage of the goal space. As presented in Section 1.3, this coverage is specifically defined with reference to the distribution of desired goals. Hence, agents should be able to efficiently explore their behavioral goal space in order to match the widest possible distribution of desired goals. Goal Exploration Processes (GEPs) are a family of frameworks for exploring multiple goals. For any environment — which can be defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$ and a transition distribution $\mathcal{T}$ that determines the next state given a current state and an action — a GEP essentially aims at maximizing its behavioral diversity by exploring the maximum number of goals. We consider goals here as pairs composed of a fitness function and a goal embedding, where the latter is the result of projecting the state space on a predefined or learned goal space $\mathcal{G}$ using a surjective function: each goal is mapped to at least one state.

GEPs were first defined in the context of intrinsically motivated population based agents [Forestier et al., 2017]. In this section, we present GEPs as a general framework regardless of the underlying motivations (which can either be external or internal). First, we start from the policy search view on GEPs to derive a policy gradient perspective for goal-conditioned RL agents. Then, depending on the source of motivations, we present the sub-families: Externally Motivated and Internally Motivated GEPs.

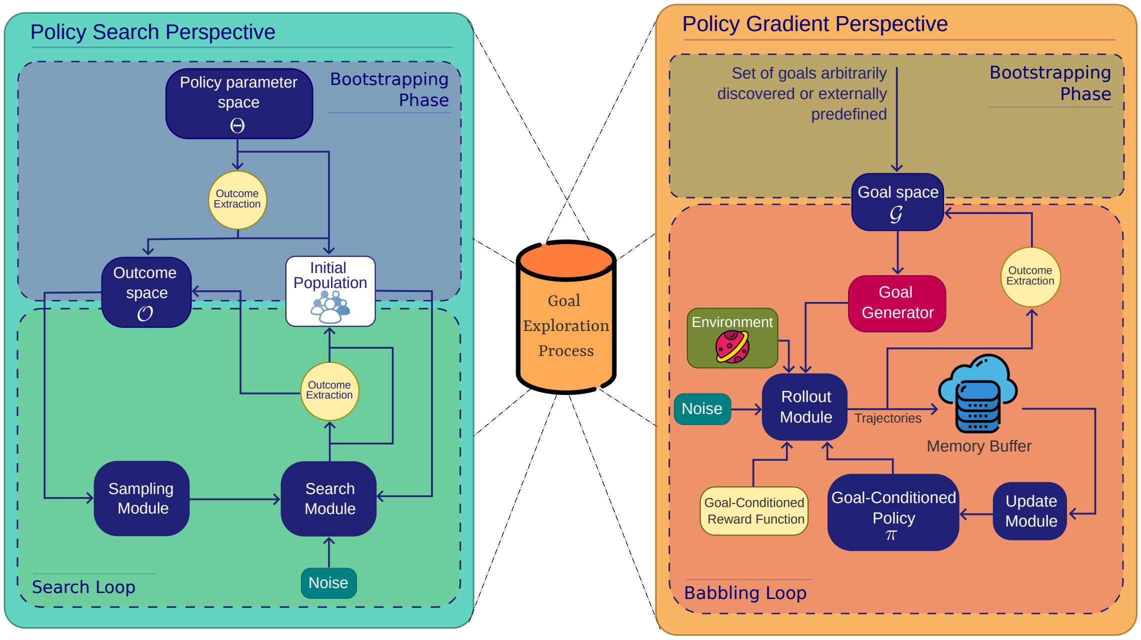


Figure 1.11: Illustration of the two stages leveraged by the Goal Exploration Processes (GEPs), as seen from the policy search perspective (left) and the goal-conditioned RL perspective (right).

1.4.1 GEPs: Policy Search Perspective

From the policy search point of view, GEPs explore multiple goals starting from an initial population of policy parameters. The process leverages two phases: a first phase called the *bootstrapping phase*, which is conducted once, and a second phase called the *search loop*, which is repeated until convergence. Both phases require an *outcome extractor*, which is a predefined deterministic function that takes as input the policy parameters and outputs the outcome of applying that particular policy in the environment.

Concerning the bootstrapping phase, N sets of policy parameters are randomly sampled from Θ. Each one of the sampled policies is fed the outcome extractor to observe the corresponding outcome, which lays in an outcome space $\mathcal{O}$. The pair formed by each policy and the corresponding outcome is stored in a buffer defining the population where the search phase will be conducted.

Concerning the search loop, the following cycle is repeated until convergence. First, a set of outcomes is sampled from the outcome space $\mathcal{O}$. Second, this sampled outcomes are fed to the search module which looks in the available population for the closest policy parameters that achieve the sampled outcomes (simply using the K-nearest neighbors algorithm for instance). Third, a noise is applied to the policy parameters picked from the previous step. This promotes behavioral diversity and enables the potential discovery of new outcomes. In fact, the noisy policy parameters are fed to the outcome extractor, yielding an outcome for each entry. Finally, the obtained outcomes are appended to the initial outcome space $\mathcal{O}$, while the pairs of policy parameters and the corresponding outcomes are added to the initial population.

1.4.2 GEPs: Policy Gradient Perspective

While the objective of GEPs from the policy search perspective is to maximize the size of the explored population of $< policy, outcome >$ pairs, the policy gradient view presents it differently. In this perspective, the output of the process can be a single policy and a set of goals that the policy can achieve. In the policy gradient perspective, the policy is conditioned on the goals. The process leverages two phases: first a *bootstrapping phase* to initialize the goal space, then a *babbling loop* to learn and discover new goals.

During the bootstrapping phase, the goal space $\mathcal{G}$ is filled with either a set of arbitrarily discovered or externally predefined goals, depending on the nature of motivations considered within the process. More details are given in Sections 1.4.3 and 1.4.4.

During the babbling loop, the following cycle is repeated until convergence. First, a goal generator is used to sample goals from the goal space $\mathcal{G}$. Second, a rollout module takes as input the sampled goals, the environment, a goal-conditioned reward function, a goal conditioned policy and noise to produce trajectories. This rollout module can be viewed as running an episode within a simulator using an arbitrary policy with predefined noise. Third, the obtained trajectories are stored in a memory buffer, which feeds an update module responsible for adjusting the goal-conditioned policy so that it maximizes the reward. Finally, the new trajectories are used to extract novel goals discovered during play. These goals are added to the initial goal space.

In the remainder of the document, we adopt the policy gradient perspective. Depending on the origins of goals obtained in the bootstrapping phase, we consider two sub-families of GEPs: externally and internally motivated.

1.4.3 Externally Motivated Goal Exploration Processes

Externally Motivated Goal Exploration Processes (EMGEPs) is a sub-family of GEPs where goals are predefined externally. Recall that a goal is a pair of a goal achievement function and a goal embedding. During the bootstrapping phase, an external program defines the goals that will be babbled and the corresponding goal achievement functions. If goals are discrete, then all goals are given. If goals are continuous, then both the support and the goal generator are given. See Figure 1.12 for an illustration.

If the goal generation process is embedded within the simulator and not the agent, then the corresponding GEP is considered as an EMGEP. Standard works that tackle the multi-goal reinforcement problem usually define a goal generation function within the environment [Schaul et al., 2015, Andrychowicz et al., 2017, Lanier et al., 2019, Li et al., 2019]. If goals are given by an external program, such as an external artificial or human agent, the corresponding GEP is also considered as an EMGEP. In particular, instruction following agents are the most straightforward EMGEPs, where agents are fully dependent on external goals in the form of natural language instructions [Hermann et al., 2017, Bahdanau et al., 2018, Chan et al., 2019, Cideron et al., 2019, Jiang et al., 2019, Fu et al., 2019].

1.4.4 Intrinsically Motivated Goal Exploration Processes

Intrinsically Motivated Goal Exploration Processes (IMGEPs) is a sub-family of GEPs where goals are exclusively discovered by the exploring agents itself. In other words, there is no external signal to provide goal embeddings nor goal achievement functions. Initially, during the bootstrapping phase, IMGEP agents have no clue whatsoever on the goal space. They use an arbitrary policy performing random actions in the environment and unlocks easy goals that are close in distribution term to the distributions of initial states. Once a sufficient set of goals is discovered, the babbling phase kicks off. As opposed to the first phase, the babbling phase uses a goal-conditioned policy. The exploration-exploitation dilemma is stronger in IMGEPs: the exploration should be efficient enough to avoid getting stuck in a particular distribution of discovered goals, but should be smooth enough to avoid catastrophic forgetting or getting the policy stuck in a local minimum.

For IMGEPs, the goal generation process is inherent to the agent. It is the agent itself that discovers the goals that it learns about (that is, it discovers both goal embeddings and goal achievement functions). Note that IMGEPs can discover a goal space whose support is defined externally (example: 3D positions, relational predicates, see Section 1.16 for more details on goal representations) [Nair et al., 2018, Colas et al., 2019, 2020b, Akakzia et al., 2021, Akakzia and Sigaud, 2022], or a goal space that is previously learned in an unsupervised fashion, using information theory techniques for example [Warde-Farley et al., 2018], see Figure 1.12 for an illustration.

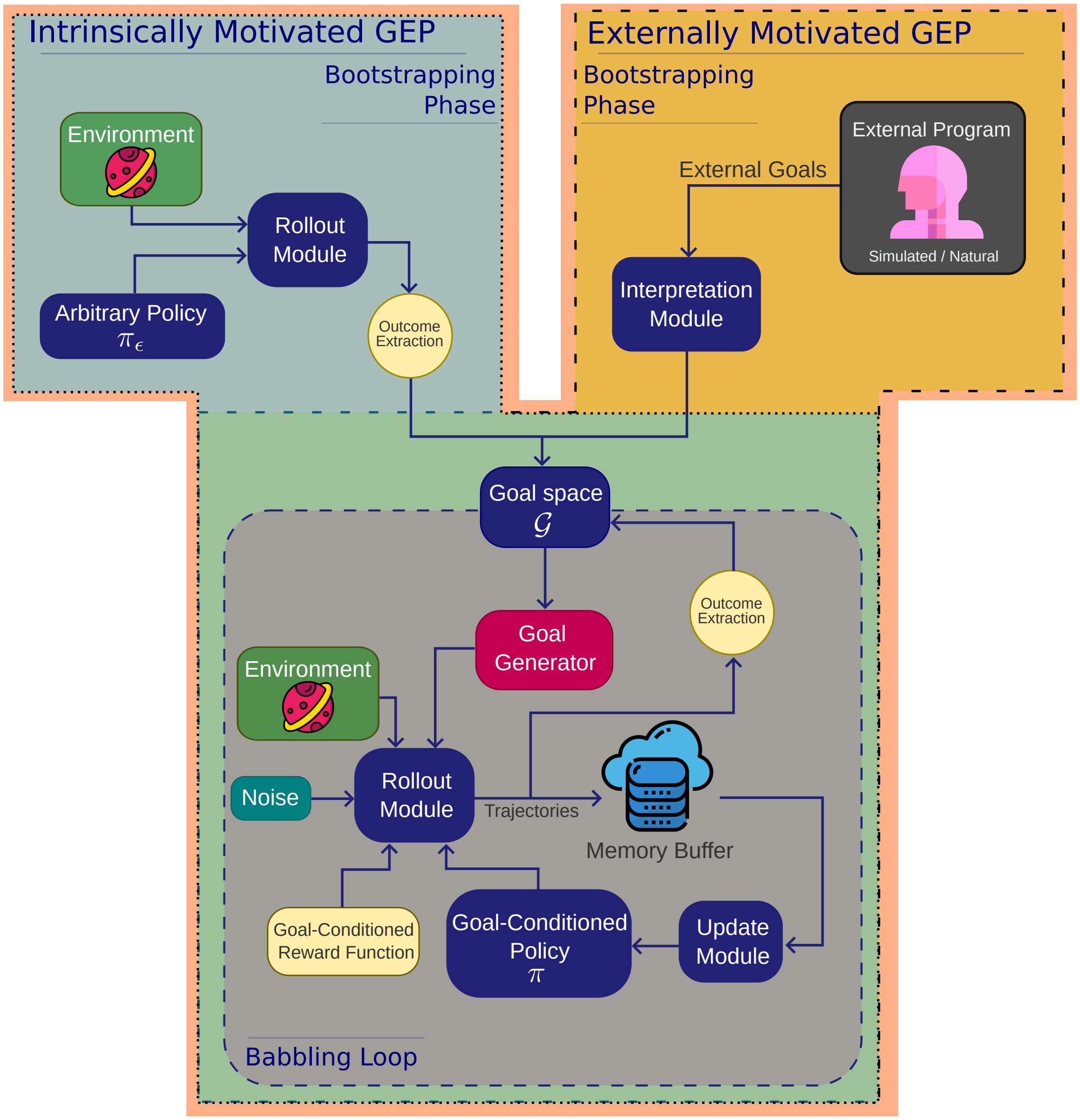


Figure 1.12: Illustration of the two sub-families of Goal Exploration Processes (GEPs): (left) IMGEPs (right) EMGEPs. Each type has its own bootstrapping phase but both share the same babbling loop.

1.5 Autotelic Reinforcement Learning

The term *autotelic* was first introduced by the humanistic psychologist Mihaly Csikszenmihaly as part of his theory of *flow*. The latter corresponds to a mental state within which embodied agents are deeply involved in some complex activity without external rewarding signals [Mihaly, 2000]. His observations was based on studying painters, rock climbers and other persons who show full enjoyment in the process of their activity without direct compensation. He refers at these activites as "autotelic", which implies that the motivating purposes (*telos*) come from the embodied agents themselves (*auto*).

In Artificial Intelligence, the term is used to define artificial agents that are self-motivated, self-organized and self-developing [Steels, 2004, Colas et al., 2022a]. More formally, autotelic agents are *intrinsically motivated* to represent, generate, pursue and learn about their *own goals* [Colas et al., 2022a]. In the context of goal exploration processes, these agents are IMGEPs endowed with an *internal goal generator*: the goals that are explored and learned about depend only on the agents themselves.

In this section, we present an overview on recent autotelic reinforcement learning — autotelic agents trained with RL algorithms. We distinguish three categories, depending on whether the goal space and the set of reachable goals is known in advance. First, we present the case where autotelic agents do not know the goal space representation, but need to learn it themselves in an unsupervised fashion (Section 1.5.1). Second, we present the case where autotelic agents know the goal space representation beforehand, but have no clue on which goals they can physically reach (Section 1.5.2). Finally, we present the case where autotelic agents know both the goal space representation and the set of reachable goals, but need to self-organize their learning in order to master these goals (Section 1.5.3)

1.5.1 Autotelic Learning of Goal Representations

When the structure of the goal space is not known in advance, artificial agents need to autonomously learn good representations by themselves. They usually rely on *information theory* methods which leverage quantities such as entropy measures and mutual information [Eysenbach et al., 2018, Pong et al., 2019]. The main idea is to efficiently explore their state space and extract interesting features that enable them to discover new skills, which they attempt to master afterwards. They use generative models such as variational auto-encoders [Kingma et al., 2019] to embed high-dimensional states into compact latent codes [Laversanne-Finot et al., 2018, Nair et al., 2018, 2020]. The underlying latent space forms the goal space, and generating a latent vector from these generative models corresponds to generating a goal from the goal space. While these approaches are task-agnostic, they usually do not leverage a sufficiently high level of abstraction. In fact, since states are usually continuous, distinguishing two different high level features corresponding to two close states is challenging (e.g. distinguishing when two blocks are close to each other without further information). Besides, the learned goal representation is usually tied to the training-set distribution, and thus cannot generate well to new situations.

1.5.2 Autotelic Discovery of Goals

When artificial agents know the structure of the goal space but have no clue about the goals that can be physically reached within this space, they need to efficiently explore and discover skills by themselves [Ecoffet et al., 2019, Pitis et al., 2020, Colas et al., 2020b, Akakzia et al., 2020b, Akakzia and Sigaud, 2022]. Such scenarios become more challenging if randomly generated goals are likely to be physically unfeasible [Akakzia et al., 2020b, Akakzia and Sigaud, 2022]. In this case, the only goals that the agents can learn about are the ones that they have discovered through random exploration. Consequently, such agents need to have efficient exploration mechanisms that overcome bottlenecks and explore sparsely visited regions of their goal space. They might also need additional features such as the ability to imagine new goals based on previous ones [Colas et al., 2020b], or to start exploring from specific states that maximize the discovery of new goals [Ecoffet et al., 2019, Pitis et al., 2020, Akakzia et al., 2020b].

1.5.3 Autotelic Mastery of Goals

In some scenarios, artificial agents can know the structure of their goal space as well as the set of goals they can physically achieve. In other words, any goal they sample using their goal generator can potentially be reached and mastered. The main challenge for these agents is not to discover new goals, but rather to *autonomously organize their training goals* in order to master as many skills as possible. This is actually challenging, especially in environments where goals are of different complexities [Lopes and Oudeyer, 2012, Bellemare et al., 2016, Burda et al., 2018, Colas et al., 2019, Lanier et al., 2019, Li et al., 2019, Akakzia et al., 2020b]. Such agents usually use *Automatic Curriculum Learning* (ACL) methods, which rely on proxies such as learning progress or novelty to generate efficient learning curricula [Lopes and Oudeyer, 2012, Bellemare et al., 2016, Burda et al., 2018, Colas et al., 2019, Akakzia et al., 2020b]. Besides, other works train generative adversarial networks to produce goals of intermediate difficulty [Florensa et al., 2017], or use methods such as asymmetric self-play to train an adversarial goal generation policy with RL which samples interesting goals for the training agent [Sukhbaatar et al., 2017].

Chapter Summary

This chapter outlined the challenges faced by standard reinforcement learning (RL) methods in the context of the autonomous acquisition of open-ended repertoires of skills. More specifically, we focused on two challenges:

- Handling *hard exploration problems* including sparse, deceptive and adversarial rewarding signals. In such scenarios, standard methods usually struggle in propagating useful information to all the states, especially in long horizon setups. They fail at leveraging the property of sacrificing short-term outcomes for good long-term rewards due to the exponential decay of future returns. This usually affects the agents exploration and increases their propensity to fall in local minima.

- Learning a single policy that leverages *multiple skills*. RL methods are usually good for solving a single task defined by a specific *Markov Decision Process* (MDP). They produce a single policy that is expert on that specific task. However, open-ended learning requires a continual learning of new skills.

First, we presented RL as a computational and algorithmic framework used to train agents on one task predefined by its external rewarding function (Section 1.1). Then, we argued that the RL formulation, which usually attempts to maximize the discounted sum of rewards, struggles with policies with longer mixing times due to the exponential decay of future returns. This affects the learning agents' exploration, especially in hard exploration problems where they might need to sacrifice short rewards in order to get good ones on the long-horizon (Section 1.2). We introduced a novel family of criteria, which we call the *delayed geometric discounting criteria*. This family generalizes the standard geometric discounts, providing more attention to rewards that are farther in the agents' life span. After that, we presented *goal-conditioned reinforcement learning* (GC-RL), a framework for training a single policy to learn multiple goals (Section 1.3). We argued that this framework is better suited for open-ended acquisition of skills. We presented *goal exploration processes* (GEPs), a family of algorithms that leverage exploration in GC-RL (Section 1.4). More specifically, we distinguished two sub-families of GEPs depending on where the goals come from: *externally motivated* GEP*s* (EMGEPs) where goals are generated by external programs and *intrinsically motivated* GEP*s* (IMGEPs) where agents generate their goals themselves. Finally, we presented *autotelic reinforcement learning agents*, a family of IMGEPs that are goal-conditioned and intrinsically motivated to represent, set and pursue their own goals. The remainder of the part investigates the design of autotelic agents that can also benefit from external teaching signals.

Chapter 2

Teachable Autotelic Agents: Developmental Perspective

Children learn through play. Depending on how independent they are from their caregivers — autonomy in setting purposes, rules and taking actions — there exist two extreme playing contexts: *free play*, where children have no rules nor boundaries and rely only on their own motivations and objectives, and *structured play*, where a third party assigns a specific purpose, instructs rules to follow and provides motivations and learning objectives. What lies in between defines a large spectrum of learning processes known as *guided play*, a combination of both extremes. Education sciences have shown that this type of assisted play helps promote the learners capabilities [Weisberg et al., 2016, Yu et al., 2018].

A similar distinction exists in Artificial Intelligence (AI), where these two extremes respectively map to *autotelic agents* — agents that are intrinsically motivated to set, pursue and learn about their own goals using their own rewarding signals — and *interactive reinforcement learning agents* — agents that exclusively rely on direct teaching signals from external sources. As in children, in between should stand a family of artificial agents that can learn from both internal and external teaching signals in order to leverage prior knowledge from social partners. We call this family *Teachable Autotelic Agents* (TAA). TAA should in principle benefit from the higher efficiency of guided play. They are not only *embodied*, but also *social* agents. Many developmental studies argue these are properties of human learning, and we believe it is a prerequisite in AI in the quest for human-level intelligent agents.

This chapter presents a road-map towards the design of such agents. Building on developmental psychology and education sciences, we identify key features of child-tutor interactions in guided play. We then present a checklist of features that future TAAs will need. This allows us to pinpoint various limitations of current AI agents. We then highlight key open research challenges to design autonomous agents that can be taught by ordinary people via natural pedagogy.

2.1 Motivations

From the etymology of the word, being *autonomous* means deciding by oneself (*autos*) of its own rules (*nomos*). More generally, an agent can be said to be autonomous if it sets its own purpose and determines its own sensorimotor behavior. Autonomy matters for AI. Indeed, at first glance, intelligence seems to require some autonomy: if we always had to tell an agent what to do at each step of a sequential decision or control process, we would not consider such an agent as intelligent. Let us temporarily consider a radical definition and call "truly autonomous" an agent which would *only* decide what to do on its own, without any constraint, e.g. consideration for our needs and expectations, without ethics. This agent would be useless, perhaps even dangerous. At first glance, *true autonomy* and *usefulness* seem to be contradictory requirements: if an autonomous agent decides what to do only on its own, how can it be useful at all?

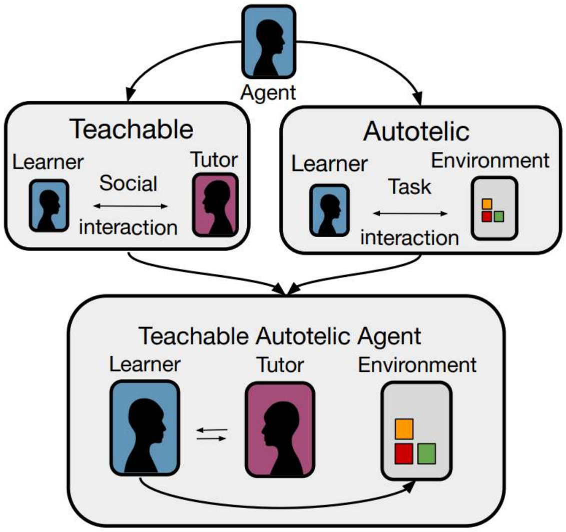


Figure 2.1: Towards Teachable Autotelic Agents. We argue that teachability and autotelic learning are complementary components to reach human-level AI. To be fully teachable, agents must also be capable of inferential social learning.

This apparently rhetorical question can be turned into a much more practical one: if a truly autonomous agent decides on its own, how can we influence it to make it useful anyways? Charkraboti et al. proposed that autonomous agents should understand and adapt to human behavior much like humans adapt to the behavior of other humans [Chakraborti et al., 2017]. How to obtain such an adaptation? Part of the answer can be found in the conclusion of the seminal paper of Alan Turing about Artificial Intelligence:

> *"It can also be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand*

and speak English. That process could follow the normal teaching of a child."
Turing [1950]

Even if they are not always autonomous in the common sense — they may need caregivers to fulfill their basic living requirements — children are definitely autonomous in the sense that we cannot fully control their behaviour. Nevertheless, we can succeed in influencing their behavior through many ways, including "normal teaching."

But again, we cannot teach an agent if it is "radically autonomous" in the sense outlined above. So, resolving the contradiction between *true autonomy* and *usefulness* requires a weaker notion of autonomy. In the following sections, we propose to equate *autonomy* with the concept of *autotelicity*: the ability to set one's own goals and to learn to achieve them using one's own learning signals [Steels, 2004, Colas et al., 2020c]. Autotelic agents are equipped with forms of intrinsic motivations enabling them to represent, generate and pursue their own goals. Though autotelic agents pursue their own goals, their behavior can still be influenced by external signals. Such agents could be made appropriate and useful through teaching if we can influence their goal representations or their goal sampling strategy. Thus this chapter proposes to reach useful autonomous agents by developing *teachable autotelic agents* — agents that choose their own goals but whose choice still benefits from teaching via natural social interactions, see Figure 2.1.

According to education sciences, children can learn in three ways:

1. **Direct Instruction**, also known as *structured play*: a tutor explicitly sets the learning goals of children step by step.

2. **Unassisted Discovery**, also known as *free play*: children are left on their own to discover new things.

3. **Assisted Discovery**, also known as *guided play*: the tutor intervenes on the discovery process of children to make it more fruitful [Weisberg et al., 2016, Yu et al., 2018].

In the remaining of this chapter, we take the stance that, when transposed to AI research, two of these developmental learning processes have their counterparts while one of them does not. First, children learning under the strict guidance of a caregiver (structured play) corresponds to *interactive reinforcement learning* artificial agents [Lin et al., 2020]. Second, learning alone by setting their own purposes and motivations (free play) corresponds to *autotelic reinforcement learning* artificial agents [Colas et al., 2022a, 2020c]. Finally, learning alone with parcimonious evaluative or instructive interventions from external tutors (guided play) has no specific counterpart in current AI research, although it is believed to be the most efficient [Yu et al., 2018]. A major step towards endowing artificial agents with the ability to be simultaneously autotelic and teachable consists in combining the properties of interactive and autotelic reinforcement learning agents. This chapter describes preliminary efforts in this direction. However, the chapter will show that this integration remains insufficient. One must also keep in mind that children are also inferential social learners and that social inference mechanisms play a key role in the extraordinary learning capabilities of children [Gweon, 2021].

In practice, AI agents will need such capabilities. When immersed in human societies, they will need to acquire the various socio-cultural skills required in these ecosystems — skills specific to regions or social groups. The only way to do so is by learning them through practical interactions with social partners. For these agents, being teachable, autonomous and capable of social inferences means being equipped with the core capabilities to acquire such skills.

Beyond this, a deeper, more fundamental reason for endowing AI agents with such capabilities stems from the endeavour of building a human-level AI [Lake et al., 2017]. It might be the case, as put forward by the *social situatedness* vision of researchers like Vygotsky [Vygotsky, 1978b], Bruner [Bruner, 1990, 1991, 2009], Tomasello [Tomasello, 2005] and others [Dautenhahn, 1995, Zlatev, 2001] that, in addition to the capability to pursue their own goals, social interactions with caregivers, tutors and mates are themselves necessary conditions for the emergence of sophisticated forms of intelligence in agents, see Lindblom and Ziemke [2003] for a review. In particular, these social and cultural interactions may play a crucial role in the acquisition of shared cognitive representations making sense for agents and their human partners. From that perspective, to obtain useful autonomous agents in a strong sense, we should focus on questions centered on the role of social interactions in the acquisition of representational capabilities compatible with those of social partners [Vygotsky, 1978a, Tomasello, 2005].

Our roadmap towards the design of such agents is organized as follows. In Section 2.2, we build on the developmental psychology and education sciences literature to extract a *teachability checklist* emerging from the natural teaching of children. More precisely, we organize these properties in three categories according to whether they involve the learning child, the tutor or the whole tutoring process. We argue that these properties are needed to teach autotelic agents. In Section 2.3, we scrutinize current research in Interactive Reinforcement Learning, Autotelic Reinforcement Learning and Inferential Social Learning under the light of these identified properties. The main outcome of this chapter is Table 2.1, where we recap all the surveyed approaches and state whether they demonstrate the properties expected from teachable autonomous agents. In Chapter 8, we build on the table and highlight key research directions towards the design or autonomous agents that can be taught by ordinary people via natural pedagogy.

Table 2.1: The teachability check-list. The table states whether the algorithms in the columns support the properties in the rows. **x** (red): no, ○ (light green): preliminary, ● (green): yes, N/A (grey): not applicable. The column on autotelic agents considers isolated agents, thus none of the properties related to social interactions can be applied.

Agents Properties	IRL agents	Autotelic Agents	Inferential Social Agents	IMAGINE	DECSTR	HME + GANGSTR
Learner Properties (Autotelism)						
Autotelic learning	N/A	●	○	●	●	●
Open-ended learning	**x**	●	**x**	●	●	●
Few shot learning	**x**	●	**x**	●	●	●
Hierarchical learning	**x**	○	**x**	**x**	**x**	●
Learner Properties (Social awareness)						
Sensitivity to social signals	●	N/A	●	●	●	●
Proficient language learning	○	N/A	**x**	○	○	**x**
Recognition of pedagogical signals	**x**	N/A	●	**x**	**x**	**x**
Observational learning	**x**	N/A	●	**x**	**x**	**x**
Learner Properties (Social inference)						
Modelling the tutor	**x**	N/A	●	**x**	**x**	●
Internalization	**x**	N/A	**x**	**x**	**x**	●
Pragmatic learning	**x**	N/A	●	**x**	**x**	**x**
Tutor Properties						
Motivation Regulation	**x**	N/A	●	**x**	**x**	●
ZPD management	**x**	N/A	●	○	○	●
Pedagogical demonstrations	**x**	N/A	●	**x**	**x**	**x**
Modelling the learner	**x**	N/A	●	**x**	**x**	●
Tutoring Process Properties						
Social-based tutoring strategy	●	N/A	●	○	**x**	○
Task-based tutoring strategy	●	N/A	●	**x**	**x**	**x**
Social communication-based transparency	●	N/A	●	**x**	**x**	**x**
Task-based transparency	●	N/A	●	●	**x**	●

2.2 Teaching in Humans

This section leverages observations from developmental psychology to characterize how children learn and how tutors contribute to their learning process. First, we extract the properties of learning children (Section 2.2.1), then those of the tutors (Section 2.2.2) and finally, those of tutoring interactions themselves (Section 2.2.3).

2.2.1 Properties of Children Learners

In the list of properties below, the first four are related to the task learning capabilities of children and the rest to their social learning capabilities.

Children are autotelic learners. Either through free or assisted play, children engage in sensorimotor interactions with their environment and discover new things which they might later try to reproduce. During free-play, children rely on their intrinsic motivations to spontaneously explore their surroundings and unlock new reachable goals [Berlyne, 1966, Gopnik et al., 1999, Chu and Schulz, 2020]. They automatically take ownership of these discovered goals, try to build their own understanding of them and attempt to pursue them. This process is very important for subsequent learning under the guidance of a tutor. For instance, the experiments in Wood et al. [1976] start with a short period of free-play with the experimental setup. This time is necessary for them to build an understanding of their surroundings before they can be influenced by social signals.

Children are open-ended learners. An extraordinary property of natural learning in children is that it is open-ended: the child can solve new problems of increasing difficulty up to becoming an adult and keeps learning during their whole life. Given the potentially infinite set of goals that they may pursue, children have to select some goal at all times. For that, they may attribute to potential goals a value of interest that evolves with time, resulting in efficiently organizing their own *developmental learning trajectory* [Piaget, 1977, Thelen and Smith, 1996, Smith and Gasser, 2005]. In other words, they self-define a learning curriculum that makes them very sample efficient: they avoid spending too much time on goals that are either too easy or too difficult, focusing on goals that present the right level of complexity at the right time.

Children are few shot learners. Children can transfer what they have learned from solving one task to a novel one. This flexibility allows them to benefit from some prior knowledge when facing a new problem that looks like the one they already know. Consequently, children are few shot learners: they can leverage what they learned in previous tasks to master new tasks in a few trials. Thus, these mechanisms allow children to discover highly complex skills such as biped locomotion, block stacking or tool use, which would have been extremely difficult to learn if they had directly addressed these goals before mastering simpler skills.

Children are hierarchical learners. It is often the case that our tasks in everyday life have a hierarchical structure. For instance, the block assembly task of Wood et al. [1976] involves several repetitions of the same basic block manipulation movements. More generally, the idea that children are hierarchical learners is pervasive in the developmental psychology literature [Eppe et al., 2022]. The elementary skills mastered by children are often stepping stones for discovering how to learn other skills of increasing complexity. As Bruner writes:

> "*The acquisition of skill in the human child can be fruitfully conceived as a hierarchical program in which component skills are combined into 'higher skills' by appropriate orchestration to meet new, more complex task requirements.*" Bruner [1973]

Children are social learners. Even shortly after birth, newborn infants can imitate complex facial expressions such as happiness, sadness and surprise [Field et al., 1983, Meltzoff and Moore, 1983, Meltzoff, 1988]. Infants can detect caregiver's eyes and prefer to look at pictures of direct gaze over averted gaze [Farroni et al., 2002]. The developmental psychology literature reported several pieces of evidence of children's social responsiveness during interaction with caregivers [Bruner, 1973, Vygotsky, 1978b, Tomasello, 2005]. This literature demonstrates children's social sensitivity, in particular when actions are directed to them such as for infant-directed speech [Saint-Georges et al., 2013].

Children are proficient language learners. Only humans master complex compositional and recursive language. Children's puzzling ability to learn it so rapidly seems to be essential to their development. Indeed, children born deaf with no access to recursive sign language and Romanian children socially abandoned in Ceausescu's orphanages showed decreased abilities for abstract compositional thinking and mental simulation [Vyshedskiy, 2019]. The mastery of compositional language seems to further support other cognitive functions such as creativity [Chomsky, 1957, Hoffmann, 2018] or analogical reasoning [Gentner and Hoyos, 2017].

Children recognize pedagogical signals. Children are able to recognize the pedagogical stance of a tutor from the general context of the initiated interaction and from communicative signals of the action itself [Csibra and Gergely, 2009]. Children are sensitive to ostensive cues such as gaze, pointing or gesture modulation ("*motionese*" [Nagai and Rohlfing, 2007]), which are generated by the caregivers to signal their pedagogical intent. This specific strategy of caregivers and the children's sensitivity to it is called natural pedagogy [Csibra and Gergely, 2009]. Children are able to infer the communicative intention and correctly interpret the instrumental intention to learn from the pedagogical caregiver.

Children are observational learners. In natural interactions, children observe others acting in the environment even when no pedagogical intention is present, and still extract a lot of information from these observations, a process referred to as observational learning [Varni et al., 1979, Meltzoff, 1999].

Children construct a model of the tutor. During assisted-play, it is crucial that children adjust their understanding of the task at hand using the available social signals. As Wood et al. [1976] put it, "*children understand goals before being able to produce them.*"

Understanding the goal from the behavior of a tutor mostly relies on inferring the tutor's expectations and reasoning to figure out how to meet them. Thus, this process clearly calls upon a mental model of the tutor's expectation. Such a mental model derives from a more general model called a Theory of Mind (ToM), as illustrated in Figure 2.2 and put forward in recent developmental psychology papers [Vélez and Gweon, 2021, Gweon, 2021]. Having a ToM means being capable of reasoning about other people's mental states [Jara-Ettinger, 2019]. The most common mental states these theories refer to are beliefs, desires and intentions.

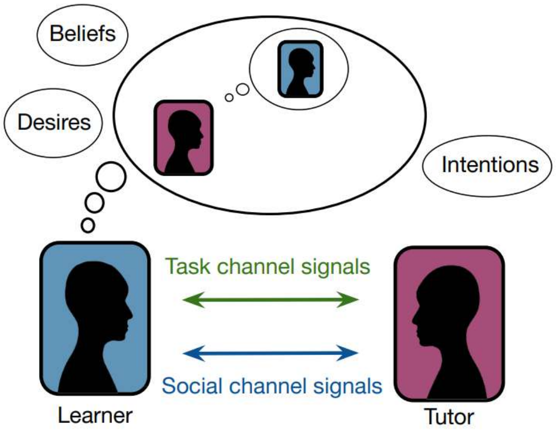


Figure 2.2: Learners build and maintain a model of the tutor's beliefs about their own knowledge, capabilities, intentions and desires. Tutors also have a model of learners, and maintain it to provide adapted teaching feedback.

Children internalize social signals. In Vygotsky's theory of development, higher-level cognitive capacities first appear as interpersonal processes before children internalize them and turn them into *intra-personal psychological tools* [Vygotsky, 1978b]. Parents first narrate the child's activities, orient their attention, keep them motivated, or decompose tasks for them. As children grow, they progressively internalize this social narration into private speech (outer speech for oneself) and, eventually, inner speech. Just like we use traditional tools to augment our control on the physical world, psychological tools augment our control on our own thoughts and behaviors. In a wealth of studies, private speech was proved instrumental to children's ability to reason and solve tasks. They use it for planning [Vygotsky, 1978b, Sokolov, 1972], and even more so when the task gets harder [Berk, 1994b]. Children seem to internalize models of their tutors and self-generate tutor-like guidance and judgements on their own behavior.

Children are pragmatic learners. Several striking experiments have shown that infants use probabilistic inference guided by an intuitive understanding of how other people think, plan and act [Gweon, 2021]. When presented with teaching signals, they consider how the information is generated, by whom, for whom, and why [Gweon et al., 2010]. Children also leverage a causal understanding of how and why those behaviors came to be, that is, a generative model of other minds. Children consider the tutor's mental states (i.e., goals, beliefs, desires) but also their utilities (i.e., costs and rewards). Given this evidence, they are able to efficiently infer the intent that is communicated to them, and thus rapidly understand the task at hand [Gweon, 2021]. All these properties make them pragmatic learners and play a key role in their learning efficiency.

2.2.2 Properties of Tutors

Up to now, we have focused on the properties of children as efficient learners. We now investigate the properties of tutors. Our aim is not to design an artificial tutoring agent, but rather to extract from this perspective the properties that help a teachable agent better respond to natural tutoring signals, coming either from a human or another artificial agent.

Tutors regulate children's motivation. Wood et al. [1976] outline that most tutoring interactions are targeting motivation regulation in children. They intend to keep them engaged in pursuing the instructed goal rather than their own goals. In turn, children must have developed their own interests beforehand, independently from social pressure. To illustrate this, Sobel and Sommerville [2010] oppose a *discovery condition*, where free-play precedes social interaction, to a *confirmation condition*, where it is the opposite: the tutor shows what to do and then leaves children on their own. They show that freely acting on a toy beforehand allows children to construct their own motivations, which then can be further regulated if needed in guided play. Wood et al. [1976] also account for the discovery condition when children are left to freely engage and get familiarized with the task before the tutoring process kicks off. As the latter starts, the authors identify three processes that regulate the children's motivations. Through *recruitment*, the tutor should find a way so that the child engages into the targeted goal rather than its other own goals.

Through *direction maintenance*, the tutor ensures that the child keeps committing to that specific goal, providing incentives to make further progress towards the goal. Finally, *frustration control* is meant to prevent children from giving up on the instructed goal.

Tutors maintain children in their zone of proximal development. In the construction domain, a scaffolding is a temporary physical structure used to support a working crew. The term has been borrowed by education researchers to refer to assistance provided by a tutor to support learning [Wood et al., 1976, Vygotsky, 1978b, Winn, 1994, Benson, 1997]. The two key aspects of scaffolding are 1) helping the learner with yet unmanageable skills while 2) allowing them to do as much as possible without help. First, in order to grasp initially hard tasks, the tutor can, for example, break them into more manageable sub-parts or engage in a *thinking aloud* process, providing guidelines on how the task should be performed [Rosenshine et al., 1992]. Second, by leaving them unassisted as much as possible, the tutor helps learners take responsibility over the task. This makes scaffolding a *temporary* process, eventually enabling the learner to work independently.

Embedded within the scaffolding process is Vygostsky's concept of the *zone of proximal development* (ZPD) [Vygotsky, 1978b]. The ZPD is defined as the area between what the learner can accomplish on its own and what can be accomplished with the help of a tutor. Notably, the ZPD is always shifting as the child learns. As a result, tutor interventions must constantly be individualized to address this change and ensure ZPD management until children eventually internalize the information and get exclusively self-regulated.

Tutors are modelling the tutee. To efficiently regulate the motivational system of children and provide appropriate instructions bringing them towards their ZPD, the tutor needs to monitor a model of the knowledge, hypotheses and performance of children as well as a model of the task itself: "*The effective tutor must have at least two theoretical models to which he must attend. One is a theory of the task or problem and how it may be completed. The other is a theory of the performance characteristics of his tutee.*" [Wood et al., 1976]. Modelling the learner helps the tutor interpret what children are trying to do, so as to efficiently teach them. The authors go further and consider that this interpretation process consists in generating hypotheses about the behavior of children, something supposedly intuitive for humans.

Tutors are pedagogical. Wood et al. [1976] provide several cues showing that natural tutoring interactions do not rely much on demonstrations to be followed blindly. They observe that, among the 30 children of their study, "*there was not a single instance of what might be called blind matching behaviour.*" Blind matching behavior is what would be observed if children were replaying the tutor's trajectory without understanding the goal. Second, the authors mention that the only acts that children imitate are those they can already perform fairly well. That is, imitating is not a way to learn how to perform the tutor's actions, it is more a way to move to the next problem solving step. Beyond showing what to do next, demonstrations also play a role in learning new skills. If they are not used for blind imitation, how do they help? "*Demonstrating or 'modelling' solutions to a task, [...] involves an 'idealization' of the act to be performed and it may involve completion or even explication of a solution already partially executed by the tutee himself*" [Wood et al., 1976]. In fact, the tutor is 'imitating' in idealized form an attempted solution tried (or assumed to have been tried) by children in the expectation that they would then 'imitate' it back in a more appropriate form. That is, the tutor builds on a model of the learner's knowledge to communicate on the problem solving steps that are still inadequate. In other terms, the tutor shows rather than it does, which characterizes pedagogical teaching.

2.2.3 Properties of the Tutoring Process

Finally, we extract properties of the tutoring process as interactions between tutors and their tutees, building on the social learning perspective of Bandura and McClelland [1977]. Tutoring can be conceptualized as a mutual exchange process using two main communication channels, the *social channel* and the *task channel*, were both partners can play alternatively the role of the emitter or the recipient, see Figure 2.2.

Tutoring combines the social and task channels. To be successful, tutoring should exploit and combine both the social and the task channels. On one hand, the social channel involves instructions, feedback or gaze (Figure 2.3). It allows learners to not only adjust their own understanding of the task through non-motor signals, but also constructively learn to understand the tutor's signals. On the other hand, the task channel involves watching motor interactions with the environment performed with the intent to teach or not (Figure 2.4). Besides, children learn to adjust their beliefs from goal-related signals, thus fostering their proactive communication abilities.

Partners are emitters and recipients. Both participants usually alternate the emitter and recipient roles, resulting in a mutual exchange (Figure 2.2). For example, when learners execute a task in order to obtain feedback from the tutor, they are first the provider and then the recipient. In more details, the tutor may emit through the social channel when providing instructions or non-verbal feedback, and through the task channel when performing or showing the task. We call the former social-based tutoring strategy and the latter task-based tutoring strategy. From the other side, the learner emits through the social channel when providing feedback on their understanding or asking questions. We call this social communication-based transparency. The learner emits through the task channel when imitating the tutor

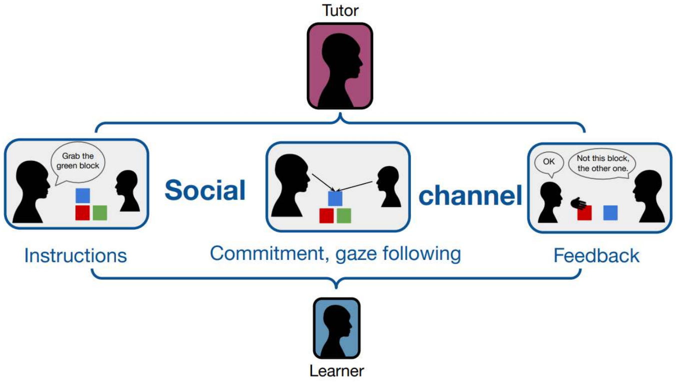


Figure 2.3: Examples of social channel signals exchanged between a tutor and a learner. The signals can be verbal, such as instructions or feedback, or non-verbal, such as gaze following. These interaction signals are used to maintain the learner engaged into the learning activity.

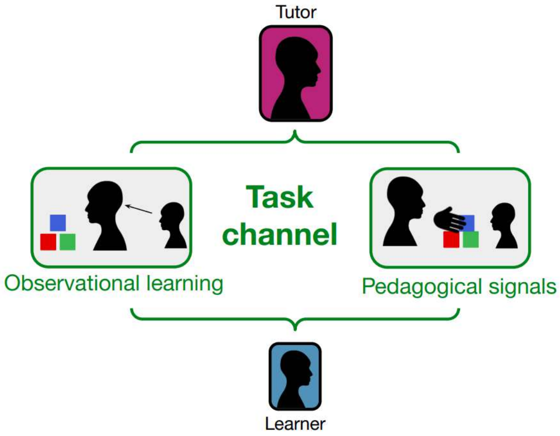


Figure 2.4: Examples of task channel signals exchanged between a tutor and a learner. Here, these signals consist of the task-related behavior of the tutor, either as demonstrations or as performance without a pedagogical intention.

or performing the task to display their current capabilities, we call it task-based transparency. Note that when both the tutor and the learner are artificial agents, since the learner has to perform the task, the presence of task-based transparency corresponds to the fact that the tutor reacts to the behavior of the learner. Combining both channels results in various "frames" of exchanges where both participants can use both channels and both roles. These frames of exchanges being goal-oriented, they are called "*pragmatic frames*" in Vollmer et al. [2016].

2.3 Interactive, Autotelic and Inferential Social Agents

After this overview of natural tutoring processes, we turn to the AI research dedicated to the design of artificial learners. We investigate three broad classes corresponding to the first columns of Table 2.1. We highlight how much they account for the properties we have listed in Section 2.2.

2.3.1 Reinforcement learners

Reinforcement Learning (RL) is a process by which an agent learns to solve sequential decision problems from a reward signal [Sutton and Barto, 2018]. The learning agent is initially ignorant of the consequences of its actions and must explore its environment to discover them. By maximizing future rewards, it progressively learns to favor more rewarding actions while avoiding costly ones. A formal definition of the RL paradigm is depicted in Section 1.1.

RL first appeared as a computational model of learning by trial-and-error in rodents [Thorndike, 1911], but is also useful to explain conditioning phenomena in monkeys [Schultz et al., 1997] and human decision making [Daw and Doya, 2006]. Thus, RL seems to be a natural framework for modelling learning to solve problems in children. However, this framework suffers from several limitations.

First, the behavior of RL agents is fully determined by the reward. In the standard framework, this reward is externally provided by a human designer with some specific goal in mind. If well defined, maximizing the cumulative reward leads to achieving the underlying goal. In the absence of this external reward, most RL agents would learn nothing. As outlined in Section 2.2.1, this is to be contrasted with children who can set their own goals and learn in full autonomy.

Second, RL agents optimize a predetermined reward function. Their behavior should converge to a corresponding optimum and stop changing, which contrasts with the open-ended learning capabilities of children. At first glance, the richer framework of multitask RL [Caruana, 1997] and its derivatives such as meta-RL [Finn, 2018] seem to do better but, as long as the set of tasks is bounded, the obtained behavior should also converge to a steady optimum. To overcome these limitations, open-ended learning approaches suggest that RL agents receive a potentially infinite sequence of unknown tasks [Doncieux et al., 2018]. However, this framework does not explicitly answer one of the most important questions: where should all these tasks and reward functions come from? Practically, all existing approaches revert to the bounded learning case: first define a (bounded) space of reward functions, then sample from it. Indeed, it is hard to imagine and define a space of reward functions that would encompass all activities humans are able to pursue. Even in that case, how should we sample appropriate tasks for an agent along its lifetime? In contrast, humans seem to generate their own goals, learn from their own reward signals and organize their own learning trajectories in a virtually infinite space of tasks.

Third, RL is notoriously slow and sample inefficient [Botvinick et al., 2019]. The groundbreaking successes that have made the field popular these last years have all been obtained with weeks of heavily parallel computations that would correspond to centuries of human experience. This is to be contrasted with the few shot learning capabilities of animals [Gruber et al., 2019] and particularly humans [Csibra and Gergely, 2009].

Finally, the standard RL framework accounts for an agent learning in isolation. As a consequence, standard RL agents lack all the social properties linked to tutoring interactions that we outlined in Sections 2.2.2 and 2.2.3.

Now that we have established the limitations of the standard RL framework, let us describe three AI research lines tackling them: interactive reinforcement learning, autotelic learning and inferential social learning.

2.3.2 Interactive Reinforcement Learners

Agents immersed in the real-world cannot be programmed beforehand to meet all their users' expectations. Interactive learning research tackles that fundamental problem by enabling non-expert users to communicate or teach their preferences and expectations in a natural way, as they would do with children [Vollmer et al., 2016]. Thus, the field of interactive learning investigates and models the way a human tutor guides the learning process of an agent by providing teaching signals [Breazeal and Thomaz, 2008]. More specifically, Interactive RL research focuses on the case where the agent is an RL agent.

From the perspective adopted in this chapter, Interactive RL can be seen as solving several of the issues outlined above. First of all, by definition, Interactive RL agents are social learners. Though not all Interactive RL agents are endowed with all these capabilities, one can find works where Interactive RL agents benefit from social-based tutoring strategy [Knox and Stone, 2010, Grizou et al., 2014, Najar et al., 2016] or task-based tutoring strategy [Abbeel and Ng, 2004, Argall et al., 2009] and works in which the Interactive RL agent displays some task-based transparency [Petit et al., 2012, Wallkotter et al., 2021] and social communication-based transparency [Boucher et al., 2012, J. and Chetouani, 2019, Wallkotter et al., 2021].

The Interactive RL approach relies too much on the tutor to drive the learning process and fails to account for the autonomy of children. As for their open-ended learning properties, one may consider that a human user may specify a sequence of increasingly more difficult tasks, for instance by specifying preferences about outcomes [Christiano et al., 2017]. But in the absence of autonomous learning, it is too much load for the tutor to always monitor the learning progress of an agent and to provide new tasks along a potentially infinite learning trajectory. Thus, Interactive RL accounts better map to the *direct instruction* approach to education than to assisted discovery.

Along the same line, though teaching signals can substantially accelerate learning, Interactive RL research does not generally consider multitask nor hierarchical contexts. Thus the corresponding agents do not display few shot learning nor hierarchical learning capabilities.

In addition, Interactive RL research does not satisfactorily account for the social interaction properties of the tutoring processes outlined in Section 2.2.3.

First, a well-known weakness of Interactive RL research is that the way naive human users tend to teach an agent is far from meeting the expectations of the RL framework [Thomaz and Breazeal, 2008a,b].

Second, the way Interactive RL research accounts for task-based tutoring strategy is called *Learning from Demonstration* (LfD) [Argall et al., 2009]. In this approach, an expert first performs highly rewarded trajectories in the environment where the task is defined. Then, data from these trajectories are collected and fed into the replay buffer of the learning agent, a sort of episodic memory. From this data, the agent learns an efficient policy with RL as if it was its own memories [Hester et al., 2018, Večerík et al., 2017]. Rather than using RL, an alternative approach called *Behavioral Cloning* (BC) consists in *cloning* the imitated policy by applying standard —another kind of machine learning process— from the same data to directly obtain a policy which behaves like the imitated one [Torabi et al., 2018]. These methods assume the experience of the expert to be directly transferred into the memory of the learning agent, which cannot happen yet in real life given that both participants have different viewpoints. Besides, they assume that the tutor and the learner share the same state space, action repertoire and dynamics of interaction with the environment. This is unlikely in real life situations as made obvious by works on the so called *correspondence problem* [Nehaniv and Dautenhahn, 2002]. Finally, they assume that the agent can perfectly observe the states and actions of the demonstrator and imitate these actions, in sharp contrast with what we outlined in Section 2.2.2. It is more likely that natural learners recognize the goals of the partner and try on their own to reach these goals. This is known as *goal emulation* [Tomasello, 1998, Ugur et al., 2011] and can be accounted for in the RL framework through inverse RL processes [Abbeel and Ng, 2004]. Among other things, this approach can help solving the correspondence problem.

Beyond this, in Interactive RL, teaching signals lack a communicative intent. As outlined in Section 2.2, infants and more generally humans are sensitive to pedagogical teaching, where the teaching signals are specifically emitted to optimize learning efficiency. This is overlooked in the Interactive RL literature, where a tutor will provide the same demonstrations to all learners without taking their current knowledge in consideration, thus failing to account for the ZPD management, pedagogical teaching and modelling the learner properties. Reciprocally, these methods expect all learners to learn equally well from a given set of demonstrations, thus Interactive RL agents cannot be seen as pragmatic learners.

Finally, natural teaching methods build on linguistic description of behaviors, instructions, explanations and both verbal and nonverbal feedback. As for the linguistic signals, Interactive RL agents rely on a limited set of predefined tokens, which cannot be confused with a form of language proficiency. For educating agents with the richer signals used in natural teaching, these agents must be equipped with richer capabilities. We claim that having the capability to represent and pursue goals, to autonomously imagine and select goals, to infer the goals of others and to interact with them about these goals are some of the required capabilities, as these goals can play a pivotal role at the interface between user expectations and autonomous behavior learning.

2.3.3 Autotelic Reinforcement Learners

The extraordinary transition from the mental life of human infants to the sophisticated intelligence of adults is mostly modelled in the domain of developmental robotics and AI [Weng et al., 2001, Zlatev, 2001, Lungarella et al., 2003]. A central line of research in this domain is interested in the design of autotelic agents [Steels, 2004, Colas et al., 2022a]. These embodied agents interact with their environment at the sensorimotor level and are provided with the ability to represent, set their own goals and reward themselves when they achieve them [Oudeyer et al., 2007, Forestier et al., 2017, Colas et al., 2020c]. By definition, they are autotelic learners.

Fundamentally, these agents are problem solvers. Implementing their learning capabilities using RL is natural, since the RL framework provides the model of choice to account for problem solving capabilities [Sutton et al., 1998]. Most of these agents are equipped with one or several goal spaces and rely on goal-conditioned RL [Colas et al., 2020c] and automatic curriculum learning [Portelas et al., 2020] to learn to achieve those goals along an open-ended developmental trajectory. This endows autotelic agents with the capability to decide which goals to target and learn about as a function of their current capabilities [Florensa et al., 2018, Fournier et al., 2019, Colas et al., 2019, Racaniere et al., 2019, Stooke et al., 2021]. See Section 1.3 for a formal definition. Thus, by contrast to Interactive RL agents, autotelic agents are open-ended and generally few shot learners. Note however that very few works focus on the importance of hierarchical learning in such agents [Eppe et al., 2022, Etcheverry et al., 2020].

By contrast, there is a growing tendency to combine autotelic architectures with language learning capabilities. Many papers combining goal-conditioned RL and language understanding have recently flourished under the banner of *instruction following agents* [Luketina et al., 2019], but the corresponding agents are not truly autotelic in the sense that they do not set their own goals. The emergence of autotelic agents endowed with language learning capabilities is covered in Chapter 8.

Thus, autotelic reinforcement learners are endowed with a lot of the properties that are missing to their interactive counterparts. But symmetrically, they generally model isolated agents, thus they miss all the interactive and social learning properties listed in Section 2.2.

2.3.4 Inferential Social Learners

While Interactive RL research provides efficient ways for AI agents to learn from demonstrations, feedback or instructions, they lack methods for implementing the richest social interactions between a tutor and a tutee presented in Section 2.2. A key to model these richer interactions consists in simultaneously addressing tutoring and learning processes.

A new family of agents that we call *inferential social learners* have started to implement these more elaborated types of interactions where the tutor and the learner build a model of each other and infer the other's beliefs, desires and intentions using these models. These models are inspired by the inferential social learning framework of Gweon [2021], which considers that learners recover the meaning of underlying others' actions by inverting intuitive causal models of the way others think, plan and act. Such inference mechanisms are crucial for social learning, notably to improve the efficiency of the tutoring process.

By drawing inspiration from language-based communication, inferential social learning goes beyond literal interpretation of actions and considers pragmatic inference [Grice, 1975]. The current approach of inferential social learning mechanisms exploits probabilistic inference over structured representations of the world, the other and even the representations of the other. A recent work describes the rational speech act (RSA) for pragmatic reasoning in language understanding [Goodman and Frank, 2016]. Its approach results in probabilistic models of pragmatic speakers and listeners able to capture the meanings of complex phenomena of linguistic interaction. Taking a more developmental perspective, some other works try to account for the way parents employ "motherese" [Gleitman et al., 1984, Saint-Georges et al., 2013] to talk to their children [Lim and Okuno, 2014], so as to endow social agents with the roots of language proficiency.

Transposing these linguistic considerations to the domain of interactions with objects, some works study communicative demonstrations [Ho et al., 2016]. Such demonstrations are not just directed towards the manipulated objects, but also accompanied with non-verbal cues such as eye gaze and or exaggerations of the demonstrations in the space–time dimensions used to convey the pedagogical intent. Again, similarly to the language case, some studies have focused on "motionese," the nonverbal equivalent of motherese [Nagai and Rohlfing, 2007, 2009]. The corresponding agents can partly be seen as autotelic, as they have some desires and intentions, but as the corresponding works focus on the interaction itself, generally they do not come with sophisticated agents capable of open-ended learning, few shot learning or hierarchical learning.

The mechanism behind these studies always requires to have models of the other's beliefs, desires and intentions and to reason about them to choose the most appropriate way to communicate, for instance selectively choose when to provide feedback and corrections [Ho et al., 2017].

In the Human-Robot Interaction community, similar works about agents not just intending to perform the ordinary action but also to convey something about it are using the concept of *legibility* [Dragan et al., 2013, Lichtenthäler and Kirsch, 2016]. The key idea is to design transparency through motion models by which a robot communicates its intent to a human observer [Wallkotter et al., 2021]. The main assumption is that humans will be able to infer the robot's intention from its motion by inverting a generative model. Thus, such works account for both forms of transparency.

In Ho et al. [2016], the authors proposed a pedagogical model based on Bayesian Inference to generate communicative demonstrations. They argue that this model should help the teacher select examples to communicate a concept to the learner. They further performed experiments involving real human instructors and showed that the results were aligned with their proposed model. A more recent work proposed a demonstration-based ZPD teaching strategy where demonstrations are not perfect, but are adapted to the current capabilities of the agent [Seita et al., 2019].

Driven by the Interactive RL framework where agents learn from an external reward function, some works also study how an agent can infer information about a reward function from observed premises in the tutoring context [Reddy et al., 2020, Bobu et al., 2020, Jeon et al., 2020]. Moving to autotelic agents, similar processes could be transposed to infer a goal rather than a reward function. A few recent works start addressing this issue by captioning the goal of a demonstration through natural language and a goal generator [Zhou and Small, 2020, Nguyen et al., 2021]. We expect follow-up of such works to contribute to answering the key question of the nature of the information conveyed by the tutor through the task channel [MacGlashan and Littman, 2015, Ho et al., 2016, 2017].

As we noted above, social inferences require that both learners and teachers reason about the beliefs and intent of each other. Thus, the corresponding agents need to be endowed with a Theory of Mind (ToM). There has been recent attempts to account for the acquisition of a ToM through inverse RL [Jara-Ettinger, 2019] and in the domain of multi-agent RL [Nguyen et al., 2020], but these works generally suffer from the same limitations as Interactive RL approaches: they do not consider explicit goals nor social inference processes. In robotics, efficient human-robot collaboration seems to require ToM models [Chakraborti et al., 2017]. A recent study investigated the combination of endowing both the learner and the social partner with models of each other in the context of learning multiple goals [Caselles-Dupré et al., 2022]. It qualifies the learning agents to be *pragmatic* whenever it attempts to infer the goal of the social partner from a given demonstrator. Besides, it qualifies the social partners to be *pedagogical* whenever they modify their demonstrations to disambiguate goal inference. This study shows that this combination results in faster learning and reduced goal ambiguity compared to standard methods that use demonstrations. To summarize the whole section, Interactive RL agents usually fall short in terms of autonomy, lacking all the properties of autotelic agents and the inferential capabilities of inferential social learners. Reciprocally, standard autotelic agents are not teachable at all. This is only when combining all approaches that future agents will display both capabilities to learn on their own and to be pedagogically taught. We now turn towards preliminary attempts in this direction.

2.4 Conclusion

Many efforts have been made to endow artificial agents with the capacity to learn from humans, in a natural and unconstrained manner. However, for now, we are still far from achieving "normal teaching of a child," in reference to Turing's view. In this chapter, by investigating the way children are taught, we claimed that autotelic agents were a better starting point for such a research than standard RL agents. The resulting agents would pursue their own goals, but should be endowed with an additional capability to be taught so that they choose their goals in accordance with the expectation of their users. We have then described some of the ongoing and immediate future work along this line of research, and revealed some of the issues which must be overcome to get closer to the way children are taught.

In the immediate future, the existing teachable autotelic agents could be combined and integrated with inferential social learning capabilities and more natural language learning capabilities. Some effort is also necessary so that these agents can actually be taught by human users, rather than by other software agents as is still the most common practice. Once this is done, given the fast progress currently observed in the design of autotelic learning agents, we expect to soon see good enough teachable autonomous agents to use them for quantitative analyses in developmental psychology studies and for a better design of education programs. We also believe that this starting point is a key move towards better insertion of AI agents in the society, with improved capabilities to communicate with and to adapt to their human users, which is one of the central concerns of AI research.

Chapter Summary

This chapter introduced the concept of *teachable autotelic agents* (TAAs). It is a subfamily of autotelic agents that can also benefit from external teaching signals that drive their exploration and help them grow their repertoires of skills. First, we conducted an overview on findings in developmental psychology and education sciences which argued for the benefits of *guided-play* in children (Section 2.1). This allowed us to extract a *teachability checklist*, including properties found in the learning children, tutors and in the learning process itself (Section 2.2). Then, we argued that this list should be leveraged by artificial agents so that they can learn autonomously and from external teaching signals. Finally, we showed that these properties are not exhaustively leveraged by current methods in artificial intelligence, including reinforcement learning (RL), interactive RL, autotelic RL and inferential social learning (Section 2.3).

Chapter 3

Predicate-based Goal-Conditioned Agents

In the previous chapter, we introduced the idea of *Teachable Autotelic Agents* from a purely developmental perspective based on key insights from the tutoring process in humans. Namely, we highlighted key properties that should exist within the tutoring process of artificial agents: a two-way communication schema between the learning agent and the human tutor. As discussed in Chapter 2, this communication triggers both the *task channel* and the *social channel*. In this chapter, we focus on the latter.

In order for the learning agent to grasp the task assigned by its teacher — which can take the form of a specific goal that needs to be achieved — it should be able in principle to *understand it*. Interestingly, humans grasp their surrounding world in terms of structured concepts. They usually communicate through language, but mastering natural language should not in principle be a prerequisite for the learning agent. In fact, pre-verbal toddlers engage in guided play with their caregivers without even mastering language.

This chapter represents a first step towards *artificial teachable prelinguistic agents*. We start from theories in developmental psychology on prelinguistic concept formation in humans. More specifically, we adopt the view of Jean Mandler — which we simply denote by the *Mandlerian View* in the remainder of the manuscript. Based on this particular perspective, we design *predicate-based* autotelic agents: goal-conditioned agents which represent their goals as a set of relational constraints based on semantic predicates.

The layout of this chapter respects the following sketch. First, we investigate several theories of prelinguistic concept formation in humans, focusing on the Mandlerian view (Section 3.1). Second, we formalize the idea of *semantic configurations* as tools to represent predicate-based goals following the Mandlerian view (Section 3.2). Finally, we introduce an architecture, which we call LGB, and which bridges the gap between behavior and language using semantic configurations (Section 3.3).

3.1 The Mandlerian View: A Theory of Prelinguistic Concept Formation

Toddlers are the best learners we know to exist. From a very young age, they exhibit impressive behaviors when interacting with their surroundings. Interestingly, these capacities are not only dependent on caregivers, but can also come from autonomous interactions with objects. So how can an infant learn concepts on her own based only on sensorimotor interactions with the physical world ? On this matter, research has been mainly divided in two conceptually and factually different perspectives: the *empiricist* view, suggesting that knowledge comes primarily from sensory experience, and the *nativist* view, stating that certain skills and abilities are hard-wired into the brain at birth.

Many approaches in developmental psychology attempted to reconcile between the nativisit and the empiricist point of view by providing a minimal set of innate prerequired cognitive processing capabilities [Spelke, 1994, Leslie, 1994, Carey, 2000, Leslie, 2005, Carey, 2009]. Spelke [1994] proposes four innate modules to handle physics, psychology, geometry and reasoning about numbers. Leslie [1994] suggests primitive modules that leverage causality, animacy and theory of mind. Carey [2000] first emphasizes the need for two innate mechanisms to handle intuitive mechanics and intuitive intentional causality, and a third innate module is added in Carey [2009] to handle the numerical cognitive system. However, all these works propose a rather static and domain-specific list of primitive capacities, which do not account for the flexibility of the continuous learning in children.

Recently, cognitive scientist Jean Mandler has proposed a new mechanism called *Perceptual Meaning Analysis* (PMA), which relies on a minimal collection of primitive core concepts embedded within infants [Mandler, 2012]. The Mandlerian view recognizes the empiricist claim that concepts are learned during the humans' lifetime, but argues that a minimal collection of primitives needs to be considered in order for this learning to kick off. The PMA mechanism transforms perceptual inputs into conceptual outputs, also known in the field of cognitive linguistics as *image-schemas*. The main difference that distinguishes PMA from earlier mechanisms is that it analyzes a few kinds of *spatiotemporal information* from a huge amount of information delivered by perceptual systems. This makes it a *minimalistic* and *domain-general* mechanism.

This section is organized as follows. First, we describe the theory of image-schemas from a developmental point of view (Section 3.1.1). Then, we focus on the Mandlerian view on the subject and introduce the PMA mechanism (Section 3.1.2). Finally, we argue that such a mechanism could be beneficial in artificial agents as it helps them categorize their sensory perceptions into semantic categories (Section 3.1.3). Our goal is not to implement image-schema extractors nor PMA mechanisms, but rather to extract some concepts behind these developmental theories that can inspire and constrain our models of how agents may learn to represent their environment.

3.1.1 The Image Schema Theory

To understand the meaning of *image-schemas*, we propose to disentangle the two composing terms. We attempt to lay the background of our next sections by defining the following concepts from a cognitive linguistics point of view: a schema, an image, and image-schemas.

The Schema Theory

The term *schema* and its plural *schemata* have greek roots associated with the terms “form” or “figure”. In the late eighteenth century, Kantian philosophy was interested in schemas, and introduced them as means to relate percepts to concepts [Kant, 1908]. More precisely, Kant starts from a known example to define a schema:

> *“The empirical conception of a plate is homogeneous with the pure geometrical conception of a circle, in as much as the roundness which is cogitated in the former is intuited in the latter.”Kant [1908]*

Kant considers a schema as a double-sided mediating representation, where one side is *sensuous* and the other is *intellectual*. These representations are *fixed templates* superimposed onto perceptions and conceptions to render meaningful representations.

Inspired by this philosophical point of view, cognitive science recently defined a schema as *“a cognitive representation comprising a generalization over perceived similarities among instances of usage”* [Kemmer and Barlow, 2000]. This term was actually adopted decades earlier by the swiss psychologist Jean Piaget, who specically used the french term “schème” to design a framework used to make sense of raw perceptual information [Piaget, 1923]. The main idea is that a schema defines a redescription of events (representation) that brings together in the same category the ones that share some common traits (similarities) to make behavioral generalization possible in humans. It can be viewed as a representation that allows humans to quickly organize new situations into categories without much effort. This organization, or *schematic processing*, starts at a very young age when toddlers first engage with their surroundings.

Mental Images

Humans have the capacity the continuously generate mental images of different concepts or events. This generation can be a recall from the memory of past experience, or an imaginary construction based on their understanding of certain concepts. The term *image* here should not be taken in its literal meaning. In fact, an image does not have to be exclusively visual, but can involve a perceptual collection of auditory, haptic, motoric, olfactory and gustatory experiences [Oakley, 2007]. In any case, images are necessarily grounded in perception, providing abstractions on which an individual could eventually add building blocks to frame new experiences. Let's try to better understand the concept of image with an example. Let's consider a detailed mental image of a pile of clothes that you may have left on your bed one morning when you were rushing out to work. This image is *specific* to those particular objects (clothes) and not to any other ones. Although this experience involves a particular set of objects, it still serves as an imaginative base for creating a schematized mental image of a stack of any other types of objects. In other words, the image is specific, but can be used with some other component to generalize the concept it represents.

Image-Schemas

It was initially the field of contemporary cognitive linguistics that got interested in combining the notions of schema and image [Arnheim, 1997, Johnson, 2013, Oakley, 2007]. *Image-schemas* can be viewed as mental images of abstract context-agnostic concepts. Thus, unlike an image, an image-schema is not specific nor fixed as it represents highly preconceptual and primitive patterns that enable reasoning in many contexts. To better understand the nuance, we consider the following example. Think of a blue lego brick being put on a red lego brick. This is an image: it is specific and fixed. The underlying image-schema can be the OBJECT ON OBJECT: it is not specific and not fixed. Hence, we are able to construct flexible and abstract representations from specific images. Other examples of image-schemas PATH-GOAL, ATTRACTION, CENTER, PERIPHERY and LINK [Geeraerts, 2006].

Most works in cognitive linguistics followed the Piagetian behaviorism point of view suggesting that concept formation in infants does not happen until language is acquired [Bogartz et al., 1997, Haith and Benson, 1998, Sloutsky, 2010]. However, decades of investigations on preverbal infant cognition emphasized the role of prelinguistic conceptualization in understanding the world [Leslie, 1994, Spelke, 1994, Mandler, 1999, 2012]. Notably, some of these works suggest that language comes later as an enrichment to these prelinguistic conceptualization, rather than being a prerequisite [Mandler, 2012, Mandler and Cánovas, 2014]. Interestingly, the same works argue that these prelinguistic image-schemas are strictly *spatial*, allowing preverbal toddlers to build their foundational conceptualization capacities by mapping spatial structure into conceptual structure [Mandler and Cánovas, 2014]. The process of formation of these image-schemas depends on some innate perceptual information responsible for monitoring toddlers attentions, as shown in experiments with 2 and 3 months children [Baillargeon, 1986, Hespos and Baillargeon, 2001]. Recently, a mechanism was proposed to describe the preverbal formation of spatial image-scheme: the Perceptual Meaning Analysis mechanism (PMA) [Mandler, 2012].

3.1.2 The Perceptual Meaning Analysis Mechanism

The Perceptual Meaning Analysis (PMA) [Mandler, 2012] is a general framework that accounts for the conceptual activity in the first years of life. It attempts to reconcile between the empiricist and the nativist views. On the one hand, the PMA mechanism recognizes that the concept formation in infants is learned, thus accounting for the empricist view. On the other hand, it argues that in order for this learning to kick off, a minimalist set of primitives need to be considered. In this section, we introduce three main properties of the PMA mechanism which are not leveraged by earlier approaches, thus making it more promising to handle prelinguistic concept formation in infants.

PMA **translates temporal information into an iconic spatial form**

In the introduction to this section, we highlighted that PMA considers only a set of primitives. A fair question arises: *what exactly are these primitives, and how do they make* PMA *more promising than other approaches?* The theory of prelinguistic concept formation established by PMA suggests that perception-based representation learning is based on *attended information*. In fact, it starts the conceptual system by directing the attention of infants to things moving on paths through space [Mandler, 2012]. A toddler sees for example the hand of her caregiver moving to grasp a toy. It is at the moment of touching the toy (establishing the "LINK" as described by Mandler [Mandler, 2012]) that the attention of the toddler gets focused on the specific perception of a hand touching an object. PMA translates this *temporal information* (hand moving towards the toy) to *iconic spatial form* (hand far from toy, hand in contact with the toy). Based on these thoughts, what is actually innate within infants is the *attention capacity towards temporal changes*, allowing them to distinguish different situations based on the contact. That is why the earliest concepts learned correspond to spatial relations [Mandler, 2012]. Compared to earlier approaches, PMA provides a domain-general mechanism, as infants may learn concepts in one situation (the example of the caregiver reaching the toy), and generalize it to any other situation including a physical contact between two objects. Figure 3.1 illustrates the idea behind the PMA module.

Language supports PMA**'s enrichment**

The PMA mechanism comes with a minimalistic collection of primitives that allow the acquisition of spatial image-schemas. However, PMA continues to develop as the infant grows up. Namely, the social situatedness that characterizes the early life of toddlers clearly affects their perceptual analysis. In particular, their confrontation to language, as they hear their caregivers engaging with them, unleashes a growing repertoire of new conceptualizations.

On the one hand, language enables the subdivision of global concepts. In fact, caregivers provide language descriptions of animate or inanimate objects which di-

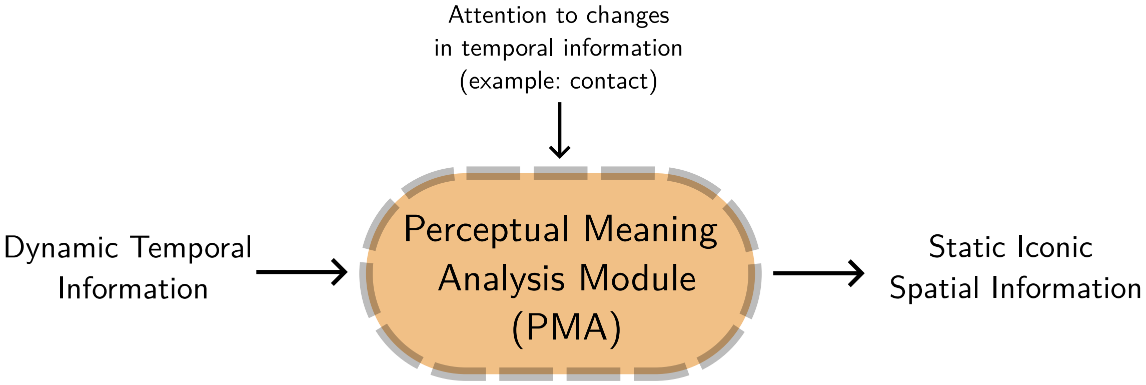

Attention to changes
in temporal information
(example: contact)
Dynamic Temporal Information
Perceptual Meaning Analysis Module (PMA)
Static Iconic Spatial Information

Figure 3.1: Illustration of the PMA module. It takes as input temporal information and translates it into iconic spatial static form. It only requires attention to temporal physical changes.

rect the toddlers' attention towards features that were originally neglected within their autonomously generated image-schemas. Experiences with 6 months infants show that they begin to use labels provided by adults to subdivide animals [Fulkerson and Waxman, 2007]. Although the child may be globally familiar with a concept, the consistent use of language-based distinctions from adults further directs the child's attentive analysis, enabling the discovery of novel properties that were originally overlooked.

On the other hand, language promotes the expansion of the conceptual system beyond spatial information. Recall that the PMA mechanism is initially strictly spatial. In fact, the PMA mechanism deals better with spatial perceptual information because they are usually structured. However, more unstructured sensory information such as colors, tastes and emotions have no primitives within the PMA mechanism. Although infants experience these unstructured information, there is no evidence, to my knowledge, for their conceptualization before language. Language labels provided by adults provide a symbolic system enabling children to map the unstructured perceived information to discrete categories.

PMA **supports Language Learning**

Infants come to the language learning task with a set of image-schemas translating their understanding on abstract concepts involving spatial primitives. Interestingly, grammatical relations within language are also abstract, which suggests that these same image-schemas could also play an important role in providing the relational notions that structure sentences. In fact, research on early language acquisition shows that children rely on notions that can be described in image-schema terms [Tomasello, 1992]. More importantly, the first explicit grammatical particles that appear in English-speaking toddlers are mainly prepositions such as *in* and *on* which respectively express *containment* and *support*. This perfectly fits with the idea that prelinguistic image-schemas which mainly involve spatial concepts support early language learning. To further reinforce this claim, researchers were interested in other languages that, unlike English, are not prepositional. For instance, some experiences involved Korean, within which containment and support are rather expressed by verbs translating a degree of fitness [Choi et al., 1999, McDonough et al., 2003]. The results showed that Korean infants too begin to acquire the common spatial morphemes of their language at the same age as English infants.

The claim that image-schemas support learning of languages such as English and Korean is possible because studies with these particular languages show that infants tend to first acquire spatial words (prepositions or verbs describing spatial relations). However, the existing variety of human languages makes the generalization of this claim somehow unsupported. In any case, I do not think that image-schemas are necessarily a prerequisite for language learning, even though they might potentially facilitate it.

3.1.3 Artificial Intelligence and Perceptual Meaning Analysis

The field of Artificial Intelligence (AI) confronts two opposing currents: *connectionist* AI that aims to learn as much as possible from data in an end-to-end fashion; *symbolic* AI which implements inductive biases and several hand-coded symbolic modules. This opposition is analogous to the one between empiricism and nativism described throughout this section (where empiricism is close to connectionist AI and nativism to symbolic AI). We align with the variety of research believing that these two currents are actually complementary. Recently, research has been investigating the middle grounding between symbolic AI and connectionist AI by incorporating symbolic representations within end-to-end computation tools such as neural networks. These approaches, called *neuro-symbolic* AI, are shown to be successful in many domains such as control [León et al., 2020], visual question answering [Andreas et al., 2016, Zhu et al., 2020] and theorem proving [Minervini et al., 2018].

As PMA describes a model for prelinguistic concept formation in infants, endowing artificial agents with a similar mechanism seems promising. More specifically, embodied artificial agents that are endowed with raw sensors might make use of a conceptual PMA-like mechanism. Such agents would be able not only to perceive their world as it is, but to build concepts and categorizations based on spatial relations. In Figure 3.2, we illustrate the potential capabilities of PMA-based agents compared with standard ones. PMA-based agents would in principle be able to categorize their sensory perceptions into semantic categories based on the underlying semantic features. This might facilitate skill acquisition, facilitate language grounding and increase behavioral diversity. All these points are discussed in the remainder of this section.

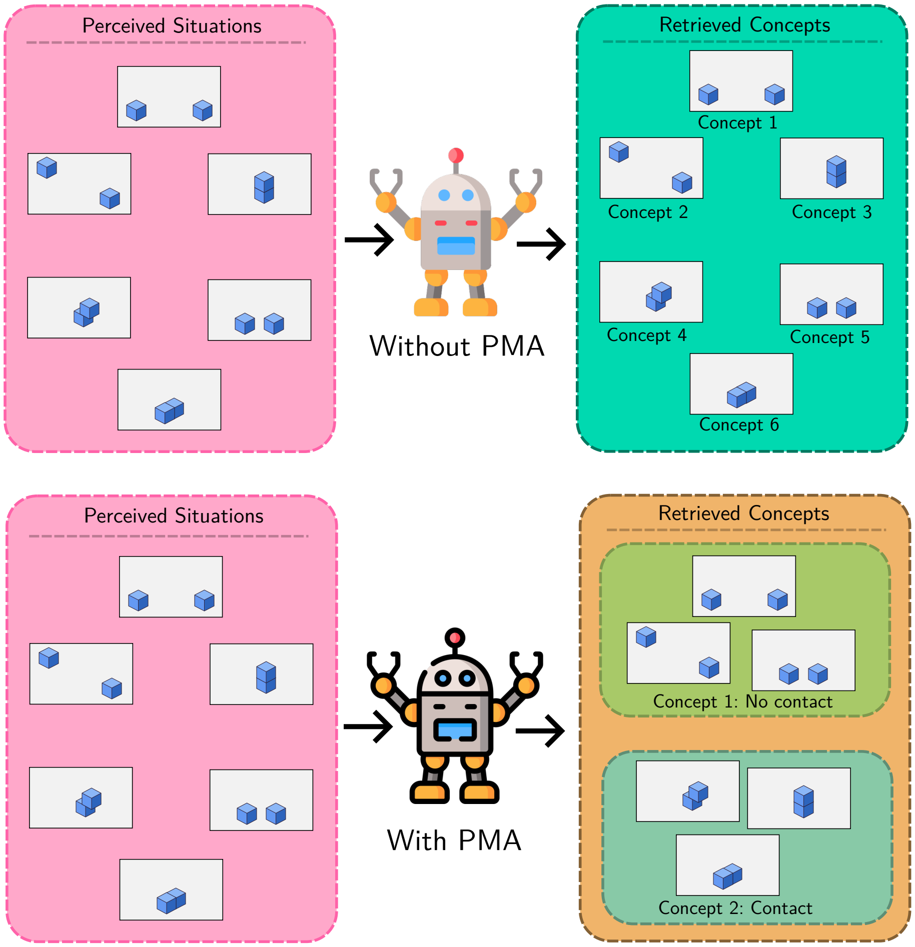


Figure 3.2: Illustration of agents' capacities to retrieve concepts from perceived sensory inputs (up) without a PMA module; (down) with a PMA module.

3.2 Formal Definition of Semantic Configurations

In Section 3.1, we described the *Mandlerian view* on prelinguistic concept formation in human toddlers. The central component of this theory is the PMA mechanism, an innate module encoded within children which translates the temporal information into an iconic form based on spatial image-schemas. We also argued that spatial relations — which represent high level concepts built upon raw perception — are the most primitive components around which children develop their cognitive capabilities, before and after language acquisition. Interestingly, other fields such as mathematics, philosophy, linguistics and computer science were interested in defining similar formal systems for reasoning about partitions and categorizations [Newell, 1994, Smullyan, 1995, Hodgson, 1995, Hughes et al., 1996, Grosof et al., 2003, Geeraerts, 2006, Mendelson, 2009]. In this section, we focus on a collection of formal systems known as *predicate logic*. These systems use quantified variables over non-logical objects, allowing to define specific relations.

The main objective of this section is to build the gap between our previous survey on prelinguistic concept formation from developmental point of view and our upcoming scientific studies on autotelic skill acquisition. We formally define the concept of *semantic configurations*: a predicate-based representation inspired from the Mandlerian view. We specifically focus on *spatial binary predicates*, which represent some boolean constraints characterizing spatial relations between *pairs* of physical objects. We provide formal definitions, properties and examples.

Binary predicates Consider a finite set of objects $O = \{o_1,\ o_2,\ ...,\ o_M\}$. A binary predicate p associated with a semantic relation $\mathbf{r}$ is an expression that takes as input any ordered pair of objects $(o_i,\ o_j) \in O^2$. $p(o_i,\ o_j)$ is said *true* if and only if "o_i $\mathbf{r}$ o_j" is verified. For simplicity, we refer to p and $\mathbf{r}$ interchangeably.

Examples of binary predicates. We consider the objects o_1 and o_2.

- The expression "o_1 is **close** to o_2" describes the predicate *close* evaluated on $(o_1,\ o_2)$.
- The expression "o_2 is **above** o_1" describes the predicate *above* evaluated on $(o_2,\ o_1)$.

Semantic mapping functions. To achieve symbol grounding into non-symbolic sensorimotor interactions using predicates, we define a *semantic mapping function* f associated with the binary predicate p as the probability that p is true given the states of the considered objects. Formally, if we consider the objects o_i, o_j and their respective states s_i, s_j, then:

$$f(s_i,\ s_j)\ =\ \mathrm{P}\ (p(o_i,\ \ o_j) \mid s_i,\ s_j).$$

We assume an oracle deterministic semantic mapping function, i.e. f is a Boolean function in $\{0, 1\}$. Practically, we hard-code a function, assumed internal to the agent, that uses predefined fixed thresholds to determine whether a predicate is true or false given the states of the considered objects. For example, for the *close* predicate, it outputs 1 if and only if the Euclidean distance between the two considered objects is below a defined threshold.

Symmetry and asymmetry. Consider a finite set of objects $O = \{o_1,\ o_2,\ ...,\ o_M\}$ and a binary predicate p. The predicate p is said *symmetric* if and only if, for any ordered pair of objects $(o_i,\ o_j) \in O^2$, "o_i $\mathbf{r}$ o_j" and "o_j $\mathbf{r}$ o_i" are equivalent. As a result, the corresponding semantic mapping function f needs to be symmetric, i.e. $f(o_i,\ o_j) = f(o_j,\ o_i)$. The predicate p is said *asymmetric* iff, for any ordered pair $(o_i,\ o_j) \in O^2$, "o_i $\mathbf{r}$ o_j" implies **not** "o_j $\mathbf{r}$ o_i".

Examples. We consider the 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.

Effective number of predicate relations. Let's consider a finite set of M objects $O = \{o_1, o_2, ..., o_M\}$ and a binary predicate p.

- If p is not symmetric, then the effective number of relations K_p that can be described without redundancy is equal to the number of **permutations** of 2 objects among M, i.e. $K_p = A_{M,2} = M(M-1)$.

- If p is symmetric, then the effective number of relations K_p is equal to the number of **combinations** of 2 objects among M, i.e. $K_p = \binom{M}{2} = \frac{M(M-1)}{2}$.

Semantic configurations based on spatial relations. Let $(p_i)_{i\in[1..P]}$ be a list of P binary predicates. The concatenation of the evaluations of the semantic mapping functions f_i on the K_{pi} pairs of objects forms a *semantic configuration.* It is an abstract representation of a scene which characterizes all relations defined by the (p_i) predicates among the M objects. This defines a binary *semantic configuration space* $\mathcal{C}_p = \{0,1\}^{K_c}$, where $K_c = \sum_{i=1}^{P} K_{p_i}$. If any world configuration can be mapped to $\mathcal{C}_p$, not all configurations are reachable (e.g. o_1 cannot be *above* and *below* o_2 at the same time).

Semantic representation space in *Fetch Manipulate*. In the *Fetch Manipulate* environment, we restrict semantic representations to the use of the *close* and *above* binary predicates applied on $M = 3$ objects. The resulting semantic configurations are formed by:

$$\begin{aligned} c_p = [&c(o_1,o_2),\ c(o_1,o_3),\ c(o_2,o_3),\ a(o_1,o_2),\\ &a(o_2,o_1),\ a(o_1,o_3),\ a(o_3,o_1),\ a(o_2,o_3),\ a(o_3,o_2)], \end{aligned}$$

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.

3.3 Grounding Language to Autonomously Acquired Skills Via Goal Generation

In the previous sections, we argued that artificial agents endowed with a PMA-like mechanism are able to distinguish spatial concepts within their surroundings. Furthermore, we proposed a formal definition of *semantic configurations*, which rely on binary relational predicates to produce abstract spatial representations. However, for these agents to be able to follow instructions from human teachers, we need to bridge the gap between these spatial representations and language. In this section, we present a novel architecture called LGB for *Language-Goal-Behavior*. LGB uses semantic configurations as a pivotal representation between embodied interactions and language instructions. On the one hand, it enables agents to learn skills by targeting configurations from the semantic representation space. On the other hand, agents can learn to generate valid semantic configurations matching the constraints expressed by language instructions. This generation can be the backbone of behavioral diversity: a given sentence might correspond to a whole set of matching configurations. This is what we propose in this chapter.

3.3.1 Motivation and Contributions

Developmental psychology investigates the interactions between learning and developmental processes that support the slow but extraordinary transition from the behavior of infants to the sophisticated intelligence of human adults [Piaget, 1977, Smith and Gasser, 2005]. Inspired by this line of thought, the central endeavour of developmental robotics consists in shaping a set of machine learning processes able to generate a similar growth of capabilities in robots [Weng et al., 2001, Lungarella et al., 2003]. In this broad context, we are more specifically interested in designing learning agents able to: 1) explore open-ended environments and grow repertoires of skills in a self-supervised way and 2) learn from a tutor via language commands.

The design of intrinsically motivated agents marked a major step towards these goals. The Intrinsically Motivated Goal Exploration Processes family (IMGEPs), for example, describes embodied agents that interact with their environment at the sensorimotor level and are endowed with the ability to represent and set their own goals, rewarding themselves over completion [Forestier et al., 2017]. Recently, goal-conditioned reinforcement learning (GC-RL) appeared like a viable way to implement IMGEPs and target the open-ended and self-supervised acquisition of diverse skills [Colas et al., 2022b].

Goal-conditioned RL approaches train goal-conditioned policies to target multiple goals [Kaelbling, 1993, Schaul et al., 2015]. While most GC-RL approaches express goals as target features (e.g. target block positions [Andrychowicz et al., 2017], agent positions in a maze [Schaul et al., 2015] or target images [Nair et al., 2018]), recent approaches started to use language to express goals, as language can express sets of constraints on the state space (e.g. *open the red door*) in a more abstract and interpretable way [Luketina et al., 2019].

However, most GC-RL which represent goals as language embeddings — widely known as *Language-Conditioned Reinforcement Learning* (LC-RL) — are not intrinsically motivated and receive external instructions and rewards. The IMAGINE approach is one of the rare examples of intrinsically motivated LC-RL approaches [Colas et al., 2020b]. In any case, the language conditioning suffers from three drawbacks. 1) It couples skill learning and language grounding. Thus, it cannot account for goal-directed behaviors in pre-verbal infants [Mandler, 1999]. 2) Direct conditioning limits the behavioral diversity associated with language input: a single instruction leads to a low diversity of behaviors only resulting from the stochasticity of the policy or the environment. 3) This lack of behavioral diversity prevents agents from switching strategy after a failure.

Contributions

We present a novel conceptual RL architecture named LGB for Language-Goal-Behavior and pictured in Figure 3.3 (right). This LGB architecture enables an agent to decouple the intrinsically motivated acquisition of a repertoire of skills (Goals → Behavior) from language grounding (Language → Goals), via the use of a semantic goal representation. To our knowledge, the LGB architecture is the only one to combine the following four features:

- It is intrinsically motivated: it selects its own (semantic) goals and generates its own rewards,
- It decouples skill learning from language grounding, accounting for infants learning,
- It can exhibit a diversity of behaviors for any given instruction,
- It can switch strategy in case of failures.

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, the DECSTR agent evolves in a 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 [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 Variational Auto-Encoder (C-VAE) to generate semantic goals from language instructions. Finally, we can evaluate the agent in an *instruction-following* phase by composing the two first phases. The experimental section investigates three questions: how does DECSTR perform in the three phases? How does it compare to end-to-end LC-RL approaches? Do we need intermediate representations to be semantic?

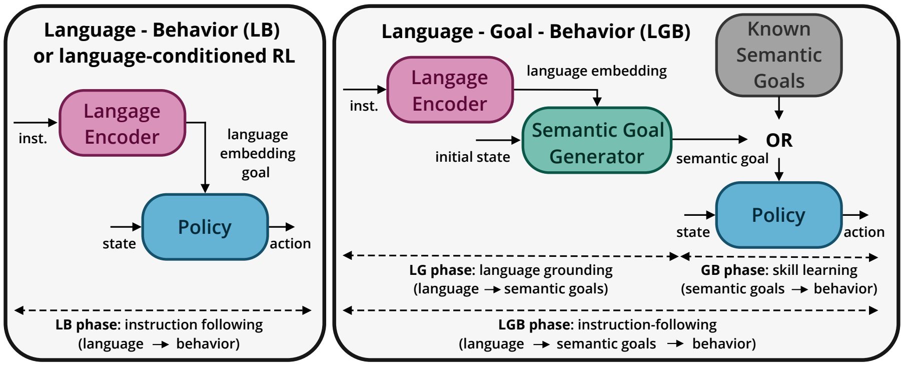


Figure 3.3: A standard language-conditioned RL architecture (left) and our proposed LGB architecture (right).

3.3.2 Related Work

Standard language-conditioned RL. Most approaches from the LC-RL literature define *instruction following* agents that receive external instructions and rewards [Hermann et al., 2017, Chan et al., 2019, Bahdanau et al., 2018, Cideron et al., 2019, Jiang et al., 2019, Fu et al., 2019], except the IMAGINE approach which introduced intrinsically motivated agents able to set their own goals and to imagine new ones [Colas et al., 2020b]. In both cases, the language-condition prevents the decoupling of language acquisition and skill learning, true behavioral diversity and efficient *strategy switching* behaviors. Our approach is different, as we can decouple language acquisition from skill learning. The language-conditioned goal generation allows behavioral diversity and strategy switching behaviors.

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

Decoupling language acquisition and skill learning. Several works investigate the use of semantic representations to associate meanings and skills [Alomari et al., 2017, Tellex et al., 2011, Kulick et al., 2013]. While the two first use semantic representations as an intermediate layer between language and skills, the third one does not use language. While DECSTR acquires skills autonomously, previous approaches all use skills that are either manually generated [Alomari et al., 2017], hand-engineered [Tellex et al., 2011] or obtained via optimal control methods [Kulick et al., 2013]. Closer to us, Lynch and Sermanet [2020] also decouple skill learning from language acquisition in a goal-conditioned imitation learning paradigm by mapping both language goals and images goals to a shared representation space. However, this approach is not intrinsically motivated as it relies on a dataset of human tele-operated strategies. The deterministic merging of representations also limits the emergence of behavioral diversity and efficient strategy-switching behaviors.

3.3.3 Methods

This section presents our proposed Language-Goal-Behavior architecture (LGB) represented in Figure 3.3 and a particular instance of the LGB architecture called DECSTR. We first present the environment it is set in, then describe the implementations of the three modules composing any LGB architecture: 1) the semantic representation; 2) the intrinsically motivated goal-conditioned algorithm and 3) the language-conditioned goal generator. We finally present how the three phases described in Figure 3.3 are evaluated.

The Language-Goal-Behavior Architecture

The LGB architecture is composed of three main modules. First, the *semantic representation* defines the behavioral and goal spaces of the agent. Second, the intrinsically motivated GC-RL algorithm is in charge of the skill learning phase. Third, the language-conditioned goal generator is in charge of the language grounding phase. Both phases can be combined in the instruction following phase. 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 3.3 and Appendix B. Instances of the LGB architecture should demonstrate the four properties listed in the introduction: 1) be intrinsically motivated; 2) decouple skill learning and language grounding (by design); 3) favor behavioral diversity; 4) allow strategy switching. We argue that any LGB algorithm should fulfill the following constraints. For LGB to be intrinsically motivated (1), the algorithm needs to integrate the generation and selection of semantic goals and to generate its own rewards. For LGB to demonstrate behavioral diversity and strategy switching (3, 4), the language-conditioned goal generator must efficiently model the distribution of semantic goals satisfying the constraints expressed by any language input.

Environment

The DECSTR agent evolves in the *Fetch Manipulate* environment: a robotic manipulation domain based on MUJOCO [Todorov et al., 2012] and derived from the

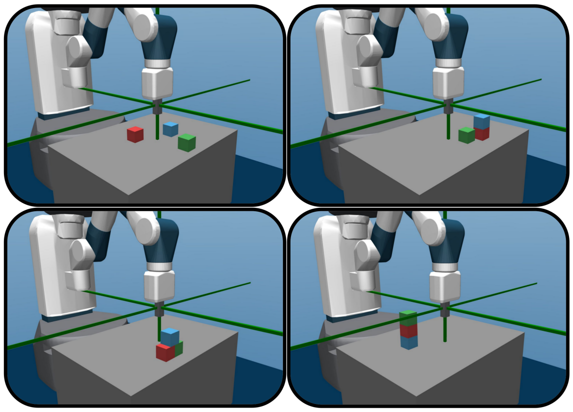


Figure 3.4: Example configurations. Top-right: (111000100).

Fetch tasks [Plappert et al., 2018], see Figure 3.4. 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. Inspired by the framework of *Zone of Proximal Development* that describes how parents organize the learning environment of their children [Vygotsky, 1978a], 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 21% of times, stacks of 3 blocks 9% of times, and a block is initially put in the agent's gripper 50% of times. This help is not provided during offline evaluations.

Semantic Representation

Semantic predicates define the Behavioral Space. 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, to know which behaviors can be achieved and which ones cannot, nor to define reward functions for each of them.

Semantic predicates in DECSTR. We assume the DECSTR agent to have 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 [Mandler, 2012]: the *close* and the *above* binary predicates. These predicates are applied 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 3.4. 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 B provides formal definitions and properties of predicates and semantic configurations.

Intrinsically Motivated Goal-Conditioned Reinforcement Learning

This section describes the implementation of the intrinsically motivated goal-conditioned RL module in DECSTR. It is powered by the Soft-Actor Critic algorithm (SAC) [Haarnoja et al., 2018] that takes as input the current state, the current semantic configuration and the goal configuration, for both the critic and the policy. We use Hindsight Experience Replay (HER) to facilitate transfer between goals [Andrychowicz et al., 2017]. DECSTR samples goals via its curriculum strategy, collects experience in the environment, then performs policy updates via 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 B.

Goal selection and curriculum learning. The DECSTR agent can only select goals among the set of semantic configurations it already experienced. We use an automatic curriculum strategy [Portelas et al., 2020] inspired from the CURIOUS algorithm [Colas et al., 2019]. The DECSTR agent 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 (via HER) is biased towards goals associated with high absolute LP and low C.

Automatic bucket generation. To facilitate robust estimation, LP is usually estimated on sets of goals with similar difficulty or similar dynamics [Forestier et al., 2017, Colas et al., 2019]. While previous works leveraged expert-defined *goal buckets*, we cluster goals based on their time of discovery, as the time of discovery is a good proxy for goal difficulty: easier goals are discovered earlier. Buckets are initially empty (no known configurations). When an episode ends in a new configuration, the $N_b = 5$ buckets are updated. Buckets are filled equally 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, the DECSTR agent 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 $\text{C} = \text{C}_{\text{recent}}$.

LP is estimated as the difference between C_{recent} and the one evaluated over the first half of the history ($\text{C}_{\text{earlier}}$). This is a crude estimation of the derivative of the C curve w.r.t. time: LP = C_{recent} - $\text{C}_{\text{earlier}}$. The sampling probability P_{i} for bucket i is:

$$P_i = \frac{(1 - C_i) * |LP_i|}{\sum_j ((1 - C_j) * |LP_j|)}.$$

In addition to the usual LP bias [Colas et al., 2019], this formula favors lower C when LP is similar. The absolute value ensures resampling buckets whose performance decreased (e.g. forgetting).

Object-centered architecture. Instead of fully-connected or recurrent networks, DECSTR uses for the policy and critic an *object-centered architecture* similar to the ones used in Colas et al. [2020b], Karch et al. [2020], adapted from Deep-Sets [Zaheer et al., 2017]. For each pair of objects, a shared network independently encodes the concatenation of body and object features and current and target semantic configurations, see Appendix Figure B. This shared network ensures efficient transfer of skills between pairs of objects. A second inductive bias leverages the symmetry of the behavior required to achieve $\textit{above}(o_i,\ o_j)$ and $\textit{above}(o_j,\ o_i)$. 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 $\textit{above}(o_i,\ o_j)$ for $i < j$) and the other half with the goals containing the other side (all $\textit{above}(o_j,\ o_i)$ for $i < j$). As a result, the $\textit{above}(o_i,\ o_j)$ predicates fall into the same slot of the shared network inputs as their symmetric counterparts $\textit{above}(o_j,\ o_i)$, 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 are automatically ensured. Appendix B further describes these inductive biases and our modular architecture.

Language-Conditioned Goal Generation

The language-conditioned goal generation module (LGG) is a generative model of semantic representations conditioned by language inputs. It is trained to generate semantic configurations matching the agent's initial configuration and the description of a change in one object-pair relation.

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 object relations from the initial configuration c_i to the final one c_f. The set of possible descriptions contains 102 sentences, each describing, in a simplified language, a positive or negative shift for one of the 9 predicates (e.g. *get red above green*). This leads to a dataset $\mathcal{D}$ of 5000 triplets: $(c_i,\ d,\ c_f)$. From this dataset, the LGG is learned using a conditional Variational Auto-Encoder (C-VAE) [Sohn et al., 2015]. Inspired by the context-conditioned goal generator from Nair et al. [2019], we add an extra condition on language instruction to improve control on goal generation. The conditioning instruction is encoded by a recurrent network that is jointly trained with the VAE via a mixture of Kullback-Leibler and cross-entropy losses. Appendix B provides the list of sentences and implementation details. By repeatedly sampling the LGG, a set of goals is built for any language input. This enables skill diversity and *strategy switching*: if the agent fails, it can sample another valid goal to fulfill the instruction, effectively switching strategy. This also enables goal combination using logical functions of instructions: *and* is an intersection, *or* is an union and *not* is the complement within the known set of goals.

Evaluation of the Three LGB phases

Skill learning phase G→B: DECSTR explores its semantic representation space, discovers achievable configurations and learns to reach them. Goal-specific performance is evaluated offline across learning as the success rate (SR) over 20 repetitions for each goal. The global performance $\overline{\text{SR}}$ is measured across either the set of 35 goals or discovery-organized buckets of goals, see Section 3.3.3.

Language grounding phase L→G: DECSTR trains the LGG to generate goals matching constraints expressed via language inputs. From a given initial configuration and a given instruction, the 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) The *Precision* is the probability that a goal sampled from the LGG belongs to $\mathcal{C}_f$ (true positive / all positive); 2) The *Recall* is percentage of elements from $\mathcal{C}_f$ that were found by sampling the LGG 100 times (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 B):

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 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 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 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 the LGG, then pursues it with its goal-conditioned policy. We consider three evaluation settings: 1) performing a single instruction; 2) performing a sequence of instructions without failure; 3) performing a logical combination of instructions. The *transition* setup measures the success rate of the agent when asked to perform the 102 instructions 5 times each, resetting the environment each time. In the *expression* setup, the agent is evaluated on 500 randomly generated logical functions of sentences, see the generation mechanism in Appendix B. In both setups, we evaluate the performance in 1-shot (SR_1) and 5-shot (SR_5) settings. In the 5-shot setting, the agent can perform *strategy switching*, to sample new goals when previous attempts failed (without reset). In the *sequence* setup, the agent must execute 20 sequences of random instructions without reset (5-shot). We also test behavioral diversity. We ask DECSTR to follow each of the 102 instructions 50 times each and report the number of different achieved configurations.

3.3.4 Experiments

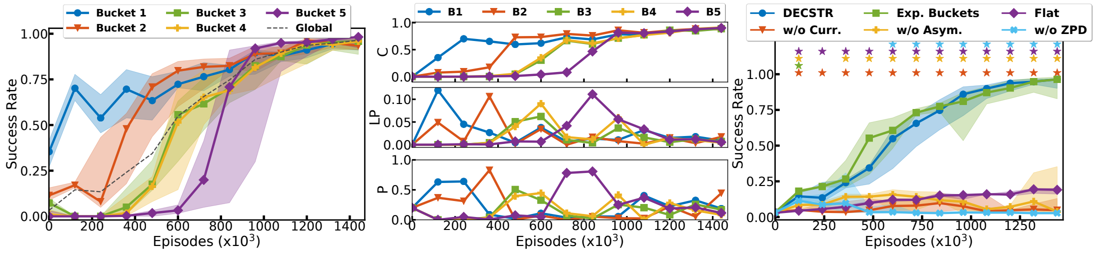
Our experimental section investigates three questions:

- How does DECSTR perform in the three phases?
- How does it compare to end-to-end language-conditioned approaches?
- Do we need intermediate representations to be semantic?

How does DECSTR perform in the three phases?

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

Skill learning phase G→B: Figure 3.5 shows that DECSTR successfully masters all reachable configurations in its semantic representation space. Figure 3.5a shows the evolution of $\overline{\text{SR}}$ computed per bucket. Buckets are learned in increasing order, which confirms that the time of discovery is a good proxy for difficulty. Figure 3.5b 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 3.5c presents the results of ablation studies. Each condition removes one component of DECSTR: 1) *Flat* replaces our object-centered modular architectures by flat ones; 2) *w/o Curr.* replaces our automatic curriculum strategy by a uniform goal selection; 3) *w/o Sym.* does not use the symmetry inductive bias; 4) In *w/o SP*, the social partner does not provide non-trivial initial configurations. In the *Expert buckets* condition, the curriculum strategy is applied on expert-defined buckets, see Appendix B. The full version of LGB performs on par with the *Expert buckets* oracle and outperforms significantly all its ablations. Appendix B presents more examples of learning trajectories, and dissects the evolution of bucket compositions along training.



(a) (b) (c)

Figure 3.5: **Skill Learning**: (a) $\overline{\text{SR}}$ 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 $\alpha = 0.05$.

Table 3.1: L→G phase. Metrics are averaged over 10 seeds, stdev < 0.06 and 0.07 respectively.

Metrics	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

Table 3.2: L→G→B phase. Mean ± stdev over 10 seeds.

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

Language Grounding Phase L→G: The LGG demonstrates the 5 types of generalization from Table 3.1. From known configurations, agents can generate more goals than they observed in training data (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 DECSTR generalizes well in a variety of contexts and shows good behavioral diversity.

Instruction Following Phase L→G→B: Table 3.2 presents the 1-shot and 5-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 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). 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.

Do we need an intermediate representation?

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

The LB baseline. To limit the introduction of confounding factors and under-tuning concerns, we base this implementation on the DECSTR code and incorporate defining features of IMAGINE, a state-of-the-art language conditioned RL agent [Colas et al., 2020b]. We keep the same HER mechanism, object-centered architectures and RL algorithm as DECSTR. We just replace the semantic goal space by the 102 language instructions. This baseline can be seen as an oracle version of the IMAGINE algorithm where the reward function is assumed perfect, but without the imagination mechanism.

Comparison in the instruction following phase L→B vs L→G→B: After training the LB baseline for 14K 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 to or far from each other, but consistently fails to stack them. Adding more attempts does not help: $\text{SR}_5 = 0.76 \pm 0.001$. The LB baseline 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, in turns, 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 leads to a reduced *diversity* in the set of mastered instructions. Decoupling thus brings significant benefits to LGB architectures.

Do we need a semantic intermediate representation?

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

The LGB-C baseline. The LGB-C baseline uses *continuous* goals expressing target block coordinates in place of semantic goals. The skill learning phase is thus equivalent to traditional goal-conditioned RL setups in block manipulation tasks [Andrychowicz et al., 2017, Colas et al., 2019, Li et al., 2019, Lanier et al., 2019]. Starting from the DECSTR algorithm, LGB-C adds a translation module that samples a set of target block coordinates matching the targeted semantic configuration which is then used as the goal input to the policy. In addition, we integrate defining features of the state-of-the-art approach from Lanier et al. [2019]: non-binary rewards (+1 for each well placed block) and multi-criteria HER, see details in Appendix B.

Comparison in skill learning phase G→B: The LGB-C baseline successfully learns to discover and master all 35 semantic configurations by placing the three blocks to randomly-sampled target coordinates corresponding to these configurations. It does so faster than DECSTR: $708 \cdot 10^3$ episodes to reach SR$= 95\%$, against $1238 \cdot 10^3$ for DECSTR, see Appendix Figure B.3. This can be explained by the denser learning signals it gets from using HER on continuous targets instead of discrete ones. In this phase, however, 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 travelled 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).

Table 3.3: 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

Comparison in language grounding phase L→G: We train the LGG to generate continuous target coordinates conditioned on language inputs with a mean-squared loss and evaluate it in the same setup as DECSTR's LGG, see Table 3.3. Although it maintains reasonable precision in the first two testing sets, the LGG achieves low recall – i.e. diversity – on all sets. The lack of semantic representations of skills might explain the difficulty of training a language-conditioned goal generator.

Conclusion. The skill learning phase of the LGB-C baseline is competitive with the one of DECSTR. However, the poor performance in the language grounding phase prevents this baseline to perform instruction following. 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.

3.3.5 Discussion and Conclusion

This section contributes LGB, a new conceptual RL architecture which introduces an intermediate semantic representation to decouple sensorimotor learning from language grounding. To demonstrate its benefits, we present DECSTR, a learning agent that 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 GC-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 efficiently learn abstract goals in an opportunistic manner. 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., 2019, Lanier et al., 2019] and a specific curiosity mechanism [Li et al., 2019]; 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]. The set of semantic predicates is, of course, domain-dependent as it characterizes the space of behaviors that the agent can explore. However, we believe it is easier and requires less domain knowledge to define the set of predicates, i.e. the dimensions of the space of potential goals, than it is to craft a list of goals and their associated reward functions.

A new approach to Language Grounding

The approach proposed here is the first simultaneously enabling to decouple skill learning from language grounding and fostering a diversity of possible behaviors for given instructions. 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*, to find alternative approaches to satisfy the same instruction when 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. It would be of interest to simultaneously perform language grounding and skill learning, which would result in "overlapping waves" of sensorimotor and linguistic development [Siegler, 1998].

Semantic configurations of variable size

Considering a constant number of blocks and, thus, fixed-size configuration spaces is a current limit of DECSTR. Future implementations of LGB may handle inputs of variable sizes by leveraging Graph Neural Networks as in Li et al. [2019]. Corresponding semantic configurations could be represented as a set of vectors, each encoding information about a predicate and the objects it applies to. These representations could be handled by Deep Sets [Zaheer et al., 2017]. This would allow to target partial sets of predicates that would not need to characterize all relations between all objects, facilitating scalability.

Conclusion

In this chapter, we have shown that introducing abstract goals based on relational predicates that are well understood by humans can serve as a pivotal representation between skill learning and interaction with a user through language. 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 next chapter, we consider a more challenging setup where skills are harder to be discovered and acquired in the first place. We investigate the influence of the goal space representation and the architecture of the agents' internal models on the transfer and generalization capabilities across the goal space.

Chapter Summary

This chapter introduced the concept of *predicate-based autotelic agents*, a sub-family of autotelic agents that represent their goals as sets of relational spatial predicates. We first investigated the nativist approach to developmental psychology which argues for the presence of innate capacities in human children. These capacities enable toddlers to understand and reason about their world from a very young age. More specifically, we focused on works of Jean Mandler, which we called the *Mandlerian view* (Section 3.2). Her theories are based on a novel mechanism known as the *perceptual meaning analysis* (PMA). The PMA mechanism translates raw temporal information into iconic forms. Mandler argues that the first concepts that are formed with the PMA mechanism are spatial concepts. We argued that endowing artificial intelligence with such capabilities could facilitate both skill learning and language grounding. Then, we proposed a formal definition of *semantic configurations*, which are state representations based on binary spatial predicates (Section 3.2). Finally, we introduced the *language-goal-behavior* (LGB) architecture which bakes these semantic configurations into autotelic agents. We show that these predicate-based representations play a pivotal role between language and behavior, and improve their language grounding and skill learning capabilities (Section 3.3).

Chapter 4

Transfer and Generalization in Autotelic Agents

Among the properties of *Teachable Autotelic Agents* outlined in Chapter 2, open-ended learning and few-shot learning are crucial to develop growing repertoires of skills. Learning agents should be able to efficiently generalize to novel situations and transfer their learned skills. Without these properties, such agents would always have to learn from scratch, even though they have already mastered primitive skills that could potentially be leveraged to acquire more complex ones.

Combining primitive skills and building upon them to solve harder tasks is a key challenge within artificial intelligence. In the context of *autotelic agents*, transfer and adaptibility seem to depend on two key features: *the goal space design*, and *the policy architecture*. On the one hand, the goal representation — whether it is learned or predefined — should encapsulate an adequate structure that defines a specific topology in the goal space. We already investigated a family of predicate-based goal spaces in Chapter 3, which seem to be promising as they assess abstract relational features between goals.

On the other hand, since the behavior of artificial agents does not only depend on how they represent their goals, but also on how they take actions, we investigate *Graph Neural Networks* (GNNs) as technical tools to model policies in autotelic agents. This choice is also motivated by developmental approaches, as research in psychology shows that humans perceive their world in a structured fashion [Winston, 1970, Palmer, 1975, Navon, 1977, Markman, 1989, Kemp and Tenenbaum, 2008, Tenenbaum et al., 2011, Battaglia et al., 2016, 2018, Godfrey-Smith, 2021].

This chapter is organized as follows. In Section 4.1, we start by introducing GNNs as technical tools to endow artificial agents with relational inductive biases. Besides, we present an overview on the use of GNNs in the field of RL. In Section 4.2, we investigate the impact of both goal space choice and policy architecture design in the context of autotelic agents in multi-object manipulation domains.

4.1 Graph Neural Networks

Recently, deep learning methods have been used to solve a significant amount of problems in different domains. Ranging from image classification [Redmon et al., 2016, Ren et al., 2015] and video processing [Zhang et al., 2016] to speech recognition [Hinton et al., 2012] and neural machine translation [Luong et al., 2015, Wu et al., 2017], these methods use parameterized neural networks as building blocks. Consequently, such methods are usually end-to-end, requiring few to no assumptions. They feed their networks with raw streams of data which are usually represented in the Euclidean Space. However, many applications rather represent data in non-Euclidean domains and use graphs with complex relationships and interdependencies. Standard usage of deep learning techniques usually struggle with this type of unstructured representations.

Interestingly, research has been interested in leveraging graph-based information using neural networks. Namely, *Graph Neural Networks* (GNNs) were proposed as computational frameworks that handle unstructured data using neural networks that they share between nodes and edges [Wang et al., 2016b, Battaglia et al., 2016, Santoro et al., 2017, Zaheer et al., 2017, Hamrick et al., 2017, Sanchez-Gonzalez et al., 2018, Battaglia et al., 2018, Zambaldi et al., 2018, Wang et al., 2018, Bapst et al., 2019, Li et al., 2019, Colas et al., 2020b, Akakzia et al., 2020b, Akakzia and Sigaud, 2022]. Although these methods are all based on the same idea, they use different techniques depending on how they handle computations within their GNNs' definition. There exist several surveys that propose different taxonomies for GNNs-based methods [Bronstein et al., 2017, Hamilton et al., 2017, Battaglia et al., 2018, Lee et al., 2018, Wu et al., 2020]. In this section, rather than presenting an exhaustive survey of GNNs, our goal is to define the building blocks including definitions and computational schemes. Besides, we focus on applications in RL and present a short overview of standard methods.

4.1.1 Relational Inductive Bias with Graph Neural Networks

First, we propose a definition for the central component of GNNs: the graph.

Graph. A graph is a mathematical structure used to model *pairwise relations* between *objects*. More formally, we denote a graph by an ordered pair $G = (V, E)$, where V is the set of vertices or nodes — the objects — and E is the set of edges — the pairwise relations. We denote a single node by $v_i \in V$, and an edge traveling from node v_i to node v_j as $e_{ij} \in E$. We also define the neighborhood of a node v_i to be the set of nodes to which v_i is connected by an edge. Formally, this set is defined as $\mathcal{N}(v_i) = \{v_j \in V \mid e_{ij} \in E\}$. Finally, we consider some global features which characterize the whole graph, and we denote them by u.

Undirected and Directed Graphs. The definition above suggests that the edges of a graph G are inherently directed from a *source* node to a *recipient* node. In some special scenarios, a graph can be *undirected*: that is, $e_{ij} = e_{ji}$ for each pair of nodes v_i and v_j. In this case, the relation between nodes is said to be *symmetric*. If the edges are distinguished from their inverted counterparts ($e_{ij} \neq e_{ji}$), then the graph is said to be *directed*.

Graph Input

The input of a graph corresponds to the parsed input features of all its nodes, all its edges and some other global features characterizing the whole system. Active lines of research that are orthogonal to our work are exploring methods that enable the extraction of such parsed features from raw sensory data [Watters et al., 2017, Van Steenkiste et al., 2018, Li et al., 2018, Kipf et al., 2018]. To simplify our study, we suppose the existence of a *predefined feature extractor* that automatically generates input values for each node and edge. For simplicity, we respectively denote the input features of node i, edge $i \rightarrow j$ and global features by v_i, e_{ij} and u.

Graph Output

Depending on the graph structure and the task at hand, the output of the graph can focus on different graph levels. If the functions used to produce this output are modeled by neural networks, then we speak about GNNs.

Node-level. This level focuses on the nodes of the graph. In this scenario, input features including node, edge and global features are used to produce a new embedding for each node. This can be used to perform regression and classification at the level of nodes and learn about the physical dynamics of each object [Battaglia et al., 2016, Chang et al., 2016, Wang et al., 2018, Sanchez-Gonzalez et al., 2018].

Edge-level. This level focuses on the edges of the graph. The output of the computational scheme in this case are the updated features of each node after propagating the information between all the nodes. For instance, it can be used to make decisions about interactions among the different objects [Kipf et al., 2018, Hamrick et al., 2018].

Graph-level. This level focuses on the entire graph. The output corresponds to a global embedding computed after propagating the information between all nodes of the graph. It can be used by embodied agents to produce actions in multi-object scenarios [Akakzia et al., 2020b, 2021], to answer questions about a visual scene [Santoro et al., 2017] or to extract the global properties molecules in chemistry [Gilmer et al., 2017].

Graph Computation

So far, we have formally defined graphs and distinguished three types of attention-levels which define their output. Thereafter, we explain how exactly the computation of this output is conducted. The computational scheme within GNNs involves two main properties. First, it is based on *shared* neural networks which are used to compute the updated features of all the nodes and edges. Second, it uses *aggregation functions* that pool these features in order to produce the output. These two properties provide GNNs with good combinatorial generalization capabilities. In fact, not only it enables good transfer between different nodes and edges (based on the shared networks), but also it leverages permutation invariance (based on the aggregation scheme).

We denote the shared neural networks between the nodes by NN_{nodes}, the shared neural networks between edges by NN_{edges}, and the readout neural network that produces the global output of the GNN by $NN_{readout}$. Besides, we focus on *graph-level output.* The full computational scheme is based on three steps: the *edge updates*, the *node updates* and the *graph readout.*

The edge update step. The edge update step consists in using the input features involving each edge $i \rightarrow j$ to compute its updated features, which we note e'_{ij}. More precisely, we consider the global input feature u, the input features of the source node v_i and the input features of the recipient node v_j. We use the shared network NN_{edges} to compute the updated features of all the edges. Formally, the updated features e'_{ij} of the edge $i \rightarrow j$ are computed as follows:

$$e'_{ij} \;=\; NN_{edges}(v_i, v_j, e_{ij}, u).$$

The node update step. The node update step aims at computing the updated features of all the nodes. We note v'_i these updated features for node i. To do so, the input features of the underlying node, the global features as well as the aggregation of the updated features of the incoming edges to i are considered. The incoming edges to i correspond to edges whose source nodes are necessarily in the neighborhood of i, $\mathcal{N}(i)$. The shared network NN_{nodes} is used in this computation. Formally, the updated features v'_i of the node i are obtained as follows:

$$v'_i \;=\; NN_{nodes}(v_i, Agg_{i \in \mathcal{N}(i)}(e'_{ij}), u).$$

The graph readout step. The graph readout step computes the global output of the graph. This quantity is obtained by aggregating all the updated features of the nodes within the graph. It uses the readout neural network $NN_{readout}$. Formally, the output o of the GNN is computed as follows:

$$o = NN_{readout}(Agg_{i \in graph}(v'_i)).$$

The computational steps we described above can be used in some other order. For example, one can first perform the node update using the input features of edges, then perform the edge updates using the updated nodes features. This choice usually depends on the domain and task at hand. Besides, our descriptions above are categorized within the family of *convolutional* GNNs [Bruna et al., 2013, Henaff et al., 2015, Defferrard et al., 2016, Kipf and Welling, 2016a, Levie et al., 2018, Gilmer et al., 2017, Akakzia and Sigaud, 2022], which generalize the operation of convolution from grid data to graph data by pooling features of neighbors when updating each node. There exist other categories of GNNs, such as *graph auto-encoders* [Cao et al., 2016, Wang et al., 2016a, Kipf and Welling, 2016b, Pan et al., 2018, Li et al., 2018], *spatio-temporal* GNNs [Yu et al., 2017, Li et al., 2017, Seo et al., 2018, Guo et al., 2019] and *recurrent* GNNs [Scarselli et al., 2005, Gallicchio and Micheli, 2010, Li et al., 2015, Dai et al., 2018]. Finally, the aggregation module used to perform node-wise pooling can be either some predefined permutation-invariant function such as sum, max or mean, or a more sophisticated self-attention-based function that learns attention weights for each node [Veličković et al., 2017].

4.1.2 Overview on Graph Neural Networks in RL

Recently, Graph Neural Networks have been widely used in Reinforcement Learning. In fact, they promote sample efficiency, especially in multi-object manipulation domains, where object invariance becomes crucial for generalization. In this paragraph, we introduce an overview over recent works in RL using GNNs. We divide the works in two categories: GNNs used for *model-based* RL and for *model-free* RL.

Model-based Reinforcement Learning. The idea of using GNNs in model-based reinforcement learning settings mainly amounts to representing the perceived world of the artificial agents with graphs. Recent papers have been using GNNs to learn prediction models by construction graph representations using the bodies and joints of the agents [Wang et al., 2016b, Hamrick et al., 2017, Sanchez-Gonzalez et al., 2018]. This approach is shown to be successful in prediction, system identification and planning. However, these approaches struggle when the structure of the components and joints of the agent are different. For example, they work better on the Swimmer environment than HalfCheetah, since the latter contains more joints corresponding to different components (back leg, front leg, head ...). Other approaches use Interaction Networks [Battaglia et al., 2016], which are a particular type of GNNs (which we further describe in Section 4.2) to implement transition models of the environment which they later use for imagination-based optimization [Hamrick et al., 2017] or planning from scratch [Wang et al., 2016b]

Model-free Reinforcement Learning. GNNs are also used in model-free reinforcement learning to model the policy and / or the value function [Wang et al., 2018, Zambaldi et al., 2018, Bapst et al., 2019, Li et al., 2019, Colas et al., 2020b, Akakzia et al., 2021]. On the one hand, like the model-based setting, some approaches use them to represent the agent's body and joints as a graph where the different components interact with each other to produce an action [Wang et al., 2018]. On the other hand, other approaches use it to represent the world in term of separate entities and attempt to capture the relational features between them [Zambaldi et al., 2018, Bapst et al., 2019, Li et al., 2019, Colas et al., 2020b, Akakzia et al., 2021].

Limitations

In spite of their generalization capacities provided by their permutation invariance, GNNs still show some limitations to solve some classes of problems such as discriminating between certain non-isomorphic graphs [Kondor and Trivedi, 2018]. Moreover, notions like recursion, control flow and conditional iteration are not straightforward to represent with graphs, and might require some domain-specific tweaks (for example, in interpreting abstract syntax trees). In fact, symbolic programs using probabilistic models are shown to work better on these classes of problems [Tenenbaum et al., 2011, Goodman et al., 2014, Lake et al., 2015]. But more importantly, a more pressing question is about the origin of the graph networks that most of the methods work on. In fact, most approaches that use GNNs use graphs with predefined entities corresponding to structured objects. Removing this assumption, it is still unclear how to convert sensory data into more structured graph-like representations. Some lines of active research are exploring these issues [Watters et al., 2017, Van Steenkiste et al., 2018, Li et al., 2018, Kipf et al., 2018].

4.2 Autotelic Behaviors with Graph Neural Networks

Although humans live in an open-ended world and endlessly face new challenges, they do not have to learn from scratch each time they face the next one. Rather, they have access to a handful of previously learned skills, which they rapidly adapt to new situations. In artificial intelligence, autotelic agents — which are intrinsically motivated to represent and set their own goals — exhibit promising skill adaptation capabilities. However, these capabilities are highly constrained by their policy and goal space representations. In this section, we propose to investigate the impact of these representations on the learning capabilities of autotelic agents. We study different implementations of autotelic agents using four types of Graph Neural Networks policy representations and two types of goal spaces, either geometric or predicate-based. We show that combining object-centered architectures that are expressive enough with semantic relational goals enables an efficient transfer between skills and promotes behavioral diversity. We also release our graph-based implementations to encourage further research in this direction.

4.2.1 Motivations and Contributions

A central challenge in artificial intelligence (AI) consists in designing artificial agents capable of solving an unrestricted set of tasks in a continual and open-ended skill learning process. In principle, these processes should be domain-agnostic. Reinforcement learning (RL) seems to be an adequate paradigm to solve a single sequential decision problem from a reward signal [Sutton et al., 1999a]. Nevertheless, this signal is usually predetermined and highly grounded to its designer's aspirations. Thus, the extension of the RL framework to an open-ended sequences of unpredictable tasks raises difficult questions.

Recently, a promising line of research has been interested in the design of *autotelic agents*, borrowing older ideas from [Steels, 2004]. These agents are intrinsically motivated to represent, set and pursue their own goals. Usually, they do not depend on any external reinforcement signal, since they autonomously reward themselves over the completion of their own goals. Autotelic agents are known to be open-ended learners. Through RL, they manage to acquire goal-directed behaviors which can transfer to domains sharing similar goal spaces. However, this transfer is deeply bound to their representational capabilities.

From that perspective, a key challenge consists in endowing autotelic agents with appropriate inductive biases to enhance their representational power. To enable efficient transfer, such biases should express a set of general and structured features. On the one hand, the design of the autotelic agents' goal spaces should

leverage the power of structured semantic representations. Namely, recent works in AI [Akakzia et al., 2021, Alomari et al., 2017, Kulick et al., 2013, Tellex et al., 2011] introduced symbolic high-level object-centered representations to explicitly capture abstract spatial relations such as proximity and aboveness, where the latter is used to refer to the quality of being directly above. By contrast, other works use plain spatial target coordinates specific to each of the available objects [Colas et al., 2019, Li et al., 2019, Lanier et al., 2019].

On the other hand, although neural networks are flexible tools to learn latent representations, their raw usage is insufficient to capture disentangled representations from high-dimensional structured input. Recently, Graph Neural Networks (GNNs) have been introduced to implement relational inductive biases in neural networks. They mainly rely on shared networks to transfer features among the input components. Besides, they follow efficient computation schemes: through their neighborhood aggregation and graph-level pooling schemes, they easily capture the existing relationships between nodes.

Contributions

We provide a systematic study of the use of GNNs in autotelic learning within a multi-object manipulation domain. More specifically, we investigate 4 variants of GNNs: *full graph networks*, *interaction networks*, *relation networks* and *deep sets*. Furthermore, we consider two different types of goal spaces: 1) *semantic goals* based on binary predicates describing spatial relations between physical objects; 2) *continuous goals* corresponding to specific target positions for each object. Finally, we assess the transfer capabilities of the best performing GNN-based agents by introducing three sets of held-out semantic goals defining three different scenarios: 1) transfer to combinations of configurations (such as a stack and a pyramid); 2) transfer from goals based on pair-wise relations to goals based on triple-wise relations (pyramids); 3) transfer to higher order stacking of objects.

Our results show that

- Semantic goal spaces induce a higher level of abstraction than continuous goal spaces, enabling lighter GNN-based architecture to perform on par with the ones that use the whole computational scheme.
- Performing the edge update step is sufficient for good transfer to combinations of previously seen goals.
- Node updates promote the transfer from goals based on pair-wise relations to goals based on triple-wise relations, as information flows not only between pairs, but also between all the nodes.
- Relation networks outperform full graph and interaction networks in transferring to higher order stacks on objects.

These results suggest that coupling semantic goal spaces with sufficiently representative graph-based networks helps to learn more complex goals and yields better transfer capabilities.

4.2.2 Related Work

This investigation relies on several previous works from different areas of research within AI. Namely, we consider recent findings in automatic curriculum learning, semantic goal representations, graph neural networks and graph-based autotelic learning, and combinations of several of these aspects.

Automatic Curriculum Learning

Adaptability is a key characteristic enabling humans to display an exceptional capacity to learn [Elman, 1993] and works in AI attempted to leverage similar automatic curriculum learning (ACL) schemes in artificial agents [Portelas et al., 2020].

Most of these approaches leverage forms of intrinsic motivations to power their exploration and learning progress (LP) [Bellemare et al., 2016, Achiam and Sastry, 2017, Nair et al., 2018, Burda et al., 2018, Pathak et al., 2019, Colas et al., 2019, Pong et al., 2019]. In this paper, our agents borrow the LP-based curriculum learning algorithm introduced in Colas et al. [2019] when targeting continuous goals, but we show this in not necessary when targeting semantic goals.

Semantic Goal Representations

Studies in developmental psychology suggest that notions such as proximity, animacy and containment are innately grounded in the perceptual world of the infant [Mandler, 2012]. Inspired by this line of thought, recent works in AI introduced symbolic high-level representations to explicitly capture abstract spatial relations [Tellex et al., 2011, Kulick et al., 2013, Alomari et al., 2017, Akakzia et al., 2021]. We borrow the semantic goal representations used in Akakzia et al. [2021] and based on the predicates *close* and *above*. Such semantic representations are more abstract than the classic goal-as-state representations, as they account for the underlying relations between objects independently of their perceived states, such as their geometric positions.

Graph Neural Networks

GNNs are powerful tools to implement strong inductive biases that focus on structured representations [Battaglia et al., 2018]. At the price of more computations, they efficiently foster combinatorial generalization and improve sample efficiency over standard architectures in different machine learning domains [Gilmer et al., 2017, Scarselli et al., 2005, Zaheer et al., 2017, Li et al., 2019]. GNNs parse the stream of input features into several objects, called *nodes*. They also capture the relational features between pairs of these objects which they store in the corresponding *edges*. They usually involve three computational schemes: 1) **Edge updates** using the initial features of the edge and both features of the nodes involved within that edge; 2) **Node updates** using the initial features of the node and the aggregated features of the edges that enter that nodes; 3) **Graph output** using an aggregation of either all the nodes or the edges features. The first two steps involve shared networks, which enable transfer between the different nodes and edges. Depending on the order and the nature of the computational steps, there exist many variants of GNNs. In this paper, we only consider 4 of these variants: *full graph networks* [Battaglia et al., 2018], *interaction networks* [Battaglia et al., 2016], *relation networks* [Santoro et al., 2017] and *deep sets* [Zaheer et al., 2017]. In general, these variants are shown to outperform flat architectures when combined with object-centered representations. Details about the implementations of these variants are provided in Section 4.2.6.

Graph-based Autotelic RL

GNNs have been used to solve RL problems [Zambaldi et al., 2018, Li et al., 2019, Colas et al., 2020b, Akakzia et al., 2021]. By contrast to Li et al. [2019], Colas et al. [2020b], Akakzia et al. [2021] — which explicitly associate a node to each object in an object manipulation domain — the approach in Zambaldi et al. [2018] attempts to solve the StarCraft II mini-games [Vinyals et al., 2017] without object-centered inductive bias. In the latter, the nodes do not correspond to specific objects, but rather to randomly scattered boxes of pixels. In this paper, we rather join the former group.

Structured Policies and Representations

Close to our work, [Bapst et al., 2019] study the combination of structured representations and graph-based policies and show that it outperforms setups that use less induced structure. On the one hand, like us, they consider both continuous and semantic settings. In the former, while they add a node for each target goal, we encode continuous goal features within the edges of our graphs. In the latter, while they use an additional conversion layer to feed the policy with the converted geometric features, we directly use binary semantic predicates as inputs to both our critics and actors. On the other hand, by contrast to their graph computation scheme which involves an encode-process-decode architecture [Battaglia et al., 2016], our graphs are simpler as they do not use any form of recurrence. In fact, through only one step of computation involving one edge update and one node update, inputs are converted into either actions (for the actor) or q-values (for the critic). Finally, in this paper, we consider many types of GNNs which use different computation schemes and we aim at assessing their transfer capabilities to previously held-out goals.

4.2.3 Methods

In this section, we state the problem we address in this paper, then we introduce the object manipulation environment and the two goal spaces that we consider (Section 4.2.5). Finally, we present the graph-based implementations of our autotelic agents (Section 4.2.6)

4.2.4 Problem statement

We address hard exploration problems where an agent is expected to learn a large diversity of complex behaviors. We cast the problem into the framework of goal-conditioned reinforcement learning [Colas et al., 2020c] and particularly consider autotelic agents which can represent, set and pursue their own goals. These agents learn from a sparse reward signal, i.e. they are only rewarded for reaching the goal they have set. More formally, these sparse reward autotelic agents are facing a rewardless Markov Decision Process MDP $= \{S, A, T\}$ where states $s \in S$ and actions $a \in A$ are continuous valued and the transition function $T : S \times A \rightarrow \Pi(S)$ defines the probability of reaching any state after performing an action from a state. Agents themselves are implemented as a goal-conditioned policy $\pi(a|s, g)$ and are rewarded with a function $r(s, g)$ which determines whether state s satisfies goal g. The key feature of autotelic agents is that they choose on which goal g to work at any moment. In this paper, goals are either semantic, i.e. they are represented as a set of binary predicates describing the features of the scene that matter for the agent's tasks, or continuous, i.e they directly correspond to a subset of the features perceived by the agent.

4.2.5 Environment and goal spaces

The Fetch Manipulate Environment

All agents studied in this paper evolve in the *Fetch Manipulate* domain from Akakzia et al. [2021], which is a variant of the standard *Fetch* domain [Plappert et al., 2018]. We extend it to a 5-object setup: the agent is a 4-DoF robotic arm facing 5 colored objects on a table. It perceives features of its body and of the surrounding objects. These features include geometric positions, orientations and velocities.

Semantic versus continuous Goals

From the perceived features, agents using semantic goals build high-level binary representations that assert the presence (1) or absence (0) of the binary spatial relations *above* and *close* between objects. As the latter is symmetric (*close(A, B)* = *close(B, A)*), we only consider 10 combinations of objects for this predicate. However, we consider all the 20 ordered pairs of objects for the *above* predicate. This yields semantic goal vectors of 30 dimensions. The resulting configuration space contains 2^{30} elements, among which ~ 75.000 are physically reachable. These semantic representations are inspired by the work of Mandler [2012] on a minimal set of spatial primitives children seem to be born with, or to develop early in life. Initially empty, the set of discovered semantic goals gets gradually filled each time an agent encounters new configurations.

By contrast with semantic goals, continuous goals directly use the perceived features, i.e. goals correspond to precise target positions for each available object. To succeed, agents have to place every object in its corresponding target position, see Figure 4.1 for an illustration. These goal spaces are used in many works attempting to solve multi-object manipulation problems [Colas et al., 2019, Li et al., 2019, Lanier et al., 2019].

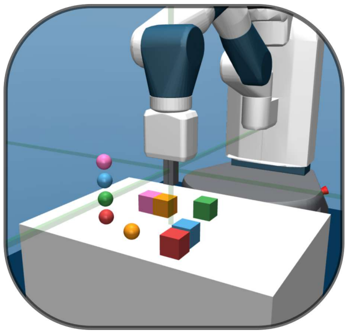


Figure 4.1: Illustration of objects, represented by colored blocks, and targets, represented by colored spheres.

Autotelic learning

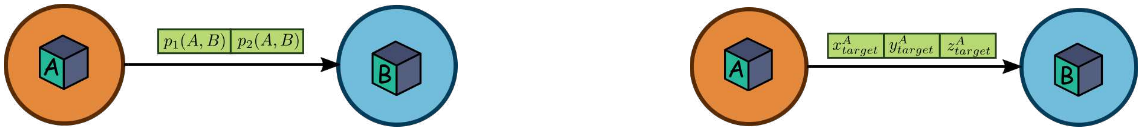
All studied agents autonomously select and attempt to master goals from the set of discovered goals. Agents using semantic goals simply reward themselves for each object for which all the predicates involving that object are verified. An episode ends successfully if all predicates about all objects are verified before a time limit. At the beginning of an episode, the blocks are procedurally placed on the table so that they are never initially stacked.

By contrast, the autotelic learning process of agents using continuous goals is more involved. First, we assume that these agents are initially aware that they can construct stacks using the available objects, and that the maximum number of objects stacked corresponds to the number of available objects. Second, at the beginning of each episode, these agents autonomously select how many objects they want to stack (from 0 up to 5 in this paper). Accordingly, target positions are generated for each object. These agents reward themselves for each object placed correctly within a range of its corresponding target position. An episode ends successfully if all objects are placed correctly before a time limit. To further accelerate the learning process, we consider biased initializations as part of a way to adapt the difficulty of the task to the learner's skills: at the beginning of each episode, and with a probability of 0.2, blocks are arranged into a stack of up to 5 objects. We call this non-trivial scene reset. To stabilize the learning process, we use an automatic LP-based curriculum [Colas et al., 2019]: based on their learning progress estimations, agents can choose to target goals with no stacks, a stack of 2, 3, 4 or 5 objects where all target positions that are not involved in stacks are automatically generated directly on the table. As shown in Appendix C, calling upon this additional ACL process is necessary for learning to work when using continuous goals.

4.2.6 Graph-based autotelic learning

In this section, we describe the implementation of the intrinsically motivated goal-conditioned RL module using GNNs. This module is powered by the Soft-Actor Critic algorithm (SAC) [Haarnoja et al., 2018] where both the critic and the policy networks are GNNs. We use the Multi-criteria Hindsight Experience Replay algorithm (MC-HER) to facilitate transfer between goals [Lanier et al., 2019]. MC-HER extends the Hindsight Experience Replay (HER) [Andrychowicz et al., 2017] strategy to multi-object scenarios, enabling further transfer between partial features of the goal vector.

Graph structure



(a) Input edge features as target predicates where p_1 =close and p_2 =above.

(b) Input edge features as target geometric positions.

Figure 4.2: Illustration of a single directed edge for semantic goals (a) and continuous goals (b).

All our agents use a fully connected graph structure: every object corresponds to a node, and all nodes are connected. First, each node holds the features of a particular object in the scene. Second, each edge linking a source and a recipient node holds partial features of the goal. As illustrated in Figure 4.2, for semantic goals, these features correspond to the predicates that involve both the source and the recipient node, while for continuous goals, they correspond to the target position of the block corresponding to the source node. Finally, the global features correspond either to the agent's body state (in the case of the policy) or to a concatenation of the agent's body state and the action (in the case of the critic). We respectively denote the node features, edges features and global features with X, E and U.

Graph computations

Although all our agents rely on the same graph structure, they use different computation schemes. In this paper, we focus on four particular types of GNNs: full graph networks (GN), interaction networks (IN), relation networks (RN) and deep sets (DS). Figure 4.3 illustrates the different computation steps for each architecture.

Full Graph Network (GN). As its name suggests, this architecture uses the whole computation scheme within a standard graph network block. See Figure 4.3 for an illustration. First, an *edge update* step is performed. A shared network NN_{mp} is used to compute the update features of each edge. It takes as input the

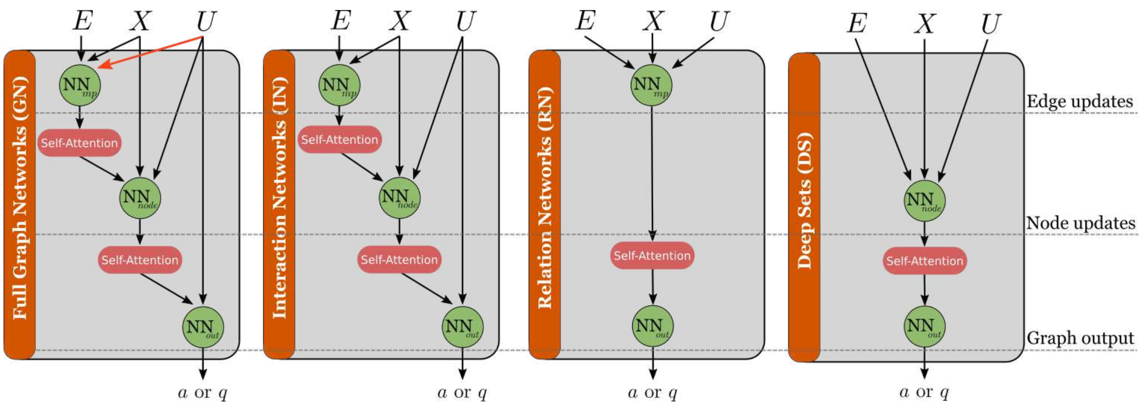


Figure 4.3: Illustration of the different computational schemes for (from left to right) GN, IN, RN and DS. E, X and U respectively correspond to the edge features, node features and global features. Note that GN uses U to update edges features (red arrow), while IN does not and RN only updates edges features, while DS only updates nodes features.

concatenated input features of each edge (goal features), the involved source and recipient nodes (object features) and the global features. Second, a *node update* step is performed for each node using a second shared network NN_{node}. It takes as input the concatenated input features of the considered node, the global features and an aggregation of the updated features of the incoming edges. Third, the *graph output* step is performed, where the updated features of the nodes are pooled, concatenated with the global features and fed to a readout network NN_{out}. The output quantity corresponds to either the action (in the case of the actor) or the q-value (in the case of the critic). In this paper, we use self-attention to compute the weighing scores used in all the aggregation steps [Vaswani et al., 2017, Veličković et al., 2017].

Interaction Network (IN). This architecture resembles the one described in the GN architecture. The only difference is that, during the edge update step, the global features are not used as inputs to the shared network NN_{mp}.

Relation Network (RN). This architecture entirely bypasses the node update step. It only performs the edge update step using the shared network NN_{mp}, which takes as inputs the initial node, edge and global features. The output vector is aggregated using a self-attention module, then fed to a readout network NN_{out}.

Deep Sets (DS). This architecture entirely bypasses the edge update step. It only performs node updates using the shared network NN_{node}. The latter takes as input the node, edge and global features, outputs a vector is later fed to a self-attention module to compute attention scores. Finally, the aggregated vector is fed to a readout network NN_{out}.

Pseudo code

The autotelic learning mechanisms with semantic and continuous goals respectively are presented in Algorithms 2 and 3. Algorithms 4 and 5 further describe how goals are sampled and updated in Algorithm 2.

Algorithm 2 Learning Semantic Goals

```
Require Env $E$, number of trajectories per step $n$, replay function $R_{mcher}$
Initialize policy $\Pi$, Uniform goal sampler $\mathcal{G}^s_{unif}$, buffer $B$.
$L_{disc} = []$
loop
    $L_{goals} \leftarrow$ Sample goals($L_{disc}$, $n$)
    $L_{traj} \leftarrow E$.rollout($\Pi$, $g$)
    $L_{disc} \leftarrow$ Update goals($L_{traj}$)
    $B$.update($L_{traj}$)
    $\mathcal{D}_{train} \leftarrow B$.sample($R_{mcher}$)
    $\Pi$.update($\mathcal{D}_{train}$)

return $\Pi$
```

Algorithm 3 Learning Continuous Goals

```
Require Env $E$, Goal classes $C_g$, number of trajectories per step $n$, replay function $R_{mcher}$
Initialize policy $\Pi$, LP-based goal sampler $\mathcal{G}^s_{LP}$, buffer $B$.
loop
    $L_{cl} \leftarrow \mathcal{G}^s_{LP}$.sample_classes($C_g$, $n$)
    $L_{goals} \leftarrow generate\_positions(L_{cl})$
    $L_{traj} \leftarrow E$.rollout($g$)
    $\mathcal{G}^s_{LP}$.update($L_{traj}$)
    $B$.update($L_{traj}$)
    $\mathcal{D}_{train} \leftarrow B$.sample($R_{mcher}$)
    $\Pi$.update($\mathcal{D}_{train}$)
return $\Pi$
```

Algorithm 4 Sample goals

```
Require discovered goals list $L_{disc}$, number of goal samples $n$,
$L_{samples} = []$
for $i$ in $[1, .., n]$ do
    if $L_{disc}$ is empty then
        $L_{samples}$.append(zeros)
    else
        $g \leftarrow L_{disc}$.sample_uniform()
return $L_{samples}$
```

Algorithm 5 Update goals

```
Require Trajectory list $L_\tau$, discovered goals list $L_{disc}$ $\mathcal{G}^s_{LP}$, buffer $B$.
for $\tau$ in $L_\tau$ do
    $(s, a, s', g_{achieved}, g_{desired}) \leftarrow \tau$
    Last_$g_{achieved} \leftarrow g_{achieved}[-1]$
    if Last_$g_{achieved}$ not in $L_{discovered}$ then
        $L_{disc}$.append(Last_$g_{achieved}$)
return $L_{disc}$
```

4.2.7 Experiments and Results

We first describe the experimental setup used in this paper. Then, we present the results obtained when training autotelic agents with semantic and continuous goals. Finally, we assess the transfer capabilities on the different architectures with three different scenarios. Additional studies and ablations are provided in Appendix C.

Experimental setup

We train 4 graph-based autotelic agents in the *Fetch Manipulate* domain with 5 objects using the graph architectures described in Section 4.2.6. We consider both the semantic and continuous goal spaces introduced in Section 4.2.5.

Evaluation Classes. To evaluate the agents, we define several evaluation classes for both semantic and continuous goals. First, for semantic goals, we consider classes of configurations where exactly i pairs of blocks are close (C_i), configurations containing stacks of size i (S_i), configurations containing pyramids of size 3 (P_3) and combinations of these. These classes are disjoint and their union does not cover the entire semantic configuration space, but they are representative enough and they enable fair comparisons between the agents. Second, for continuous goals, we consider classes of configurations where there are no stacks and where there is a stack of size i ($\widetilde{S}_i$, where the symbol $\sim$ is for continuous).

Evaluation Metrics. Evaluations are performed each 50 cycles. During one cycle, the agents perform 2 rollouts of 200 timesteps with 2 goals sampled autonomously. At test time, the per-class performance of the agent is computed on 24 goals of each evaluation class (264 semantic goals and 120 continuous goals). The measure of the agent's global success rate (SR) is the average of all the per-class successes. Testing is conducted offline and with deterministic policies.

Baseline. For both semantic and continuous goals, we consider a flat baseline, where all the perceived features are concatenated and directly fed to the neural networks. We call Semantic-Flat (S-FLAT) and Continuous-Flat (C-FLAT) the flat baseline using semantic and continuous goals respectively.

Networks Capacity. Independently of their computational scheme, we make sure all the agents have the same network capacity in terms of number of parameters to be optimized. As full graph networks use the highest number of parameters in principle, we provide the other agents with a sufficient budget to match them. Concretely, we add an additional node updater and edge updater for respectively deep sets and relation networks architecture, and we make the flat networks sufficiently deep.

Transfer Scenarios. To assert the transfer capabilities of the different agents, we investigate how GNN-based agents with semantic goal spaces perform to reach goal configurations used as test goals though they have never trained on them before. To this end, we consider three different scenarios:

1. **Transfer to combinations of constructions**, where agents are prevented from training on any goal including combinations of known constructions. The corresponding set of test goals includes the classes $S_2\&S_2$, $P_3\&S_2$ and $S_2\&S_3$.

2. **Transfer to pyramids**, where agents are prevented from training on any goal including pyramids. The underlying set of test goals includes the classes P_3, and $P_3\&S_2$.

3. **Transfer to higher stacks**, where agents are prevented from training on goals including stacks of size 3 and more. The set of corresponding test goals includes S_3. See Table 4.1 for details.

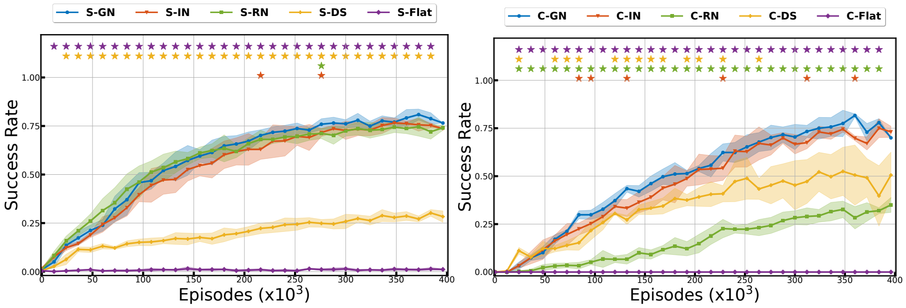
Table 4.1: Testing classes and their sizes.

Scenario-Class	Size
1 - S_2 & S_2	60
1 - S_2 & S_3	120
(1, 2) - P_3 & S_2	60
2 - P_3	30
3 - S_3	60

In practice, we make sure none of the testing goals are sampled during training by simply preventing any episode where a test goal was encountered (at any time step) from getting stored in the replay buffer. This also prevents HER 's future strategy from selecting these goals during replay. The set of training goals for each scenario may include all the other possibly encountered configurations.

Global performance metrics

In this section, we study the global performance of the different graph-based autotelic agents. Figure 4.4 presents the average SR across evaluation classes for both semantic goals (Figure 4.4a) and continuous goals (Figure 4.4b).



(a)

(b)

Figure 4.4: Global SR across training episodes with (a) Semantic (S) goals and (b) Continuous (C) goals for the considered agents. Mean ± standard deviations are computed over 5 seeds. Stars highlight statistical differences w.r.t S-GN agents (Welch's t-test with null hypothesis $\mathcal{H}_0$: no difference in the means, $\alpha = 0.05$).

Semantic Goals. The S-FLAT agents are not able to learn to reach any semantic configuration as their global SR does not increase during training (Figure 4.4a, purple curve). This is not surprising since simple MLP networks that take as input high dimensional concatenations of multiple objects struggle in disentangling the learned representations. By contrast, all the GNN-based agents are able to increase their global SR, this is consistent with previous results showing that object-centered graph-based architectures are better suited for multi-object manipulation domains [Li et al., 2019]. On the one hand, the global SR of S-DS agents gets stuck at around 25% (Figure 4.4a, orange curve). This result highlights the importance of the edge update step, as it allows one to focus on pairwise relations embedded within the semantic relational predicates used as input features to the edges. On the other hand, S-GN, S-IN and S-RN agents have similar performance across training episodes (Figure 4.4a, blue, red and green curves), and show statistical differences only rarely (see stars on Figure 4.4a). This result suggests that, when dealing with semantic relational goals, *the edge update step is crucial*. In fact, S-RN agents — which have a lighter computational scheme but exclusively perform pairwise edge updates — manage to perfectly catch with the more heavy architecture S-GN and S-IN.

Continuous Goals. Similar to semantic goals, the C-FLAT agents fail to learn any interesting behavior. However, interestingly, only GNN-based architectures that use the full computational scheme — that is, C-GN and C-IN — have the best performance. On the one hand, by contrast to the semantic goals setup, C-RN agents get stuck at around 30% of the maximum global SR (Figure 4.4b, green curve). This suggests that the edge update step is not sufficient to capture interesting features when geometric target goals are encoded within the input edge features. On the other hand, C-DS agents perform better than their C-RN counterparts, with an average global SR of 50% (Figure 4.4b, orange curve). This is probably due to the fact that the node update step — which is conducted by C-DS but not C-RN — enables information about every target geometric goal to flow to every node in the graph, which helps agents increase their performance. However, C-DS agents show higher variance compared to all the other GNN-based agents. This instability is probably explained by the role of the edge update step — which is bypassed by C-DS — in disentangling useful pairwise features that help stabilize the learning. This is made even more likely as agents that perform the whole computational scheme provide the best results and the least instabilities (Figure 4.4b, blue and red curves).

Per Class Performance Metrics

The global performance metrics show that the average SR across evaluation classes gets stuck at around 75% for both semantic and continuous agents. To investigate this, we zoom on the per class performance metrics.

Semantic Goals. Figure 4.5 shows the per-class performance of S-GN, S-IN, S-RN and S-DS. First, and as the global performance metrics suggest, S-GN, S-IN and

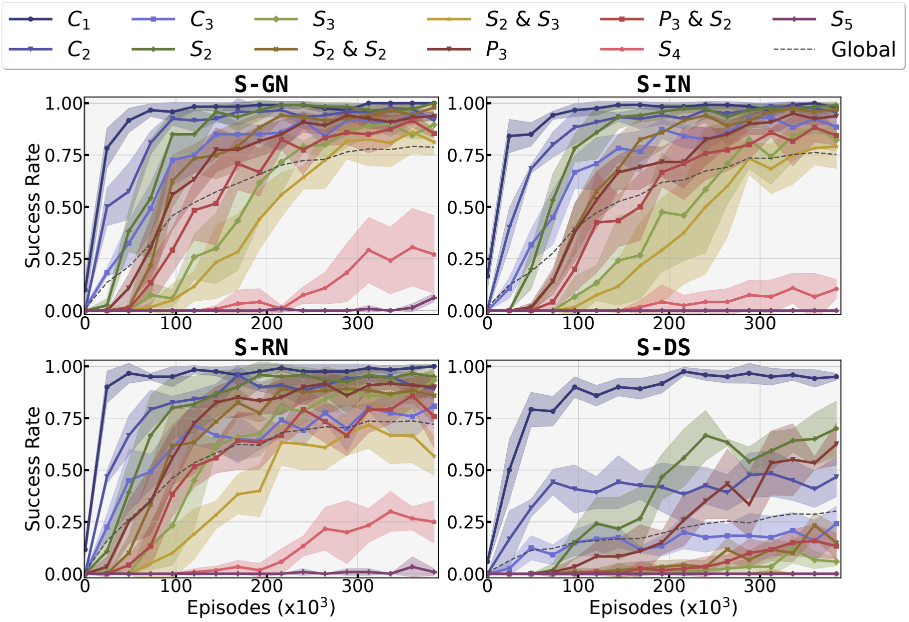


Figure 4.5: Local SR for each class across training episodes with continuous goals. Mean ± standard deviations are computed over 5 seeds.

S-RN show very similar local performance. They are all able to master all the classes (with at least 65% of SR) except for S_4 and S_5. This failure occurs because the learned policies are sub-optimal. In fact, when rewarding themselves for each object placed correctly, the critics would most likely be *greedy*: incremental rewards should come fast, even if this means not constructing stacks in the trivial order (from base upwards). As a result, agents would start by constructing the upper part of a stack, then placing it on the base object. This is not a problem for S_3 since robotic arms can pick and place a stack of two blocks. However, in S_4, it is impossible to pick and place a stack of three blocks. See Figure 4.6 for an illustrative example. Second, the S-DS agents struggle with classes that involve many constraints to be satisfied. This is because they lack enough representational power to disentangle pairwise relations between objects.

Continuous Goals. Figure 4.7 shows the per-class performance of C-GN, C-IN and C-DS. All agents first master the easy classes, before moving up to the more difficult ones. This results from these agents leveraging automatic curriculum learning, using their LP estimation as a proxy to choose goals that are at an affordable level of complexity. However, as opposed to semantic goals, there is less interference between classes and transfer is poorer (per-class SR increases sequentially). On the one hand, C-DS agents show a lot of instabilities beyond the $\widetilde{S}_2$ class. This further supports the idea that the node update step alone in deep sets does not provide enough representational power. On the other hand, both C-GN and C-IN manage to reach goals in all the evaluation classes, from no stacks at all to stacks of 5

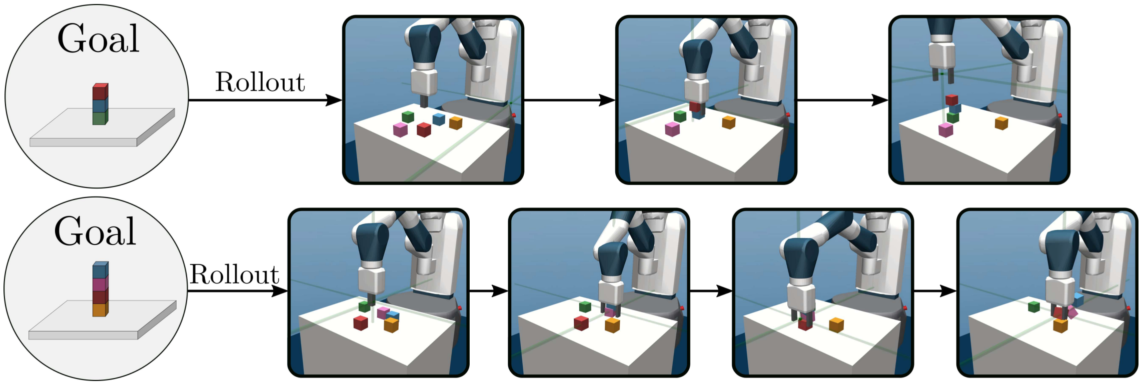


Figure 4.6: Example of sub-optimal behavior with semantic goals when targeting a goal in S_3 (up) and in S_4 (down). The agent tries to pick and place a stack of three objects and fails (down).

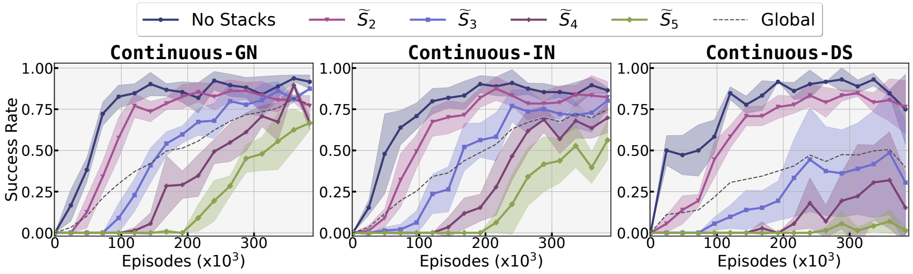
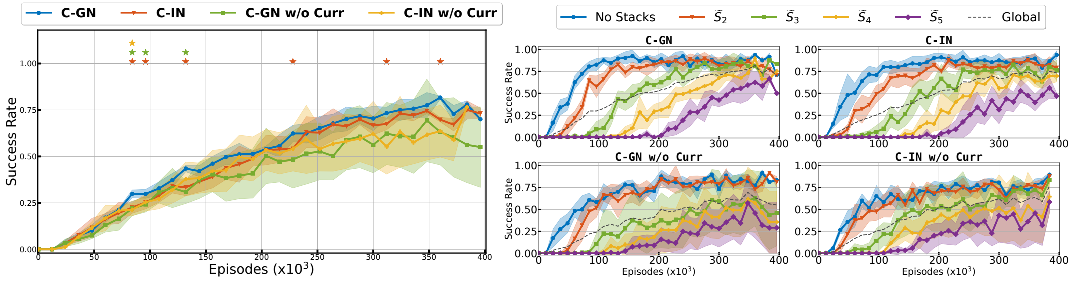


Figure 4.7: Local SR for each class across training episodes with continuous goals. Mean ± standard deviations are computed over 5 seeds.

objects. However, they are both unable to maximize their per-class performance. This suggests that learning policies that can achieve all evaluation classes at the same time with continuous goals is difficult and requires more training budget.

Curriculum Ablation

To study the relative importance of the LP-based curriculum learning mechanism used with continuous goals, we introduce ablations of C-GN and C-IN which uniformly sample a class of goals without any particular prioritization. We only consider architectures based on GN and IN in this ablation study since they show the best results with reference to the global performance metrics. Figure 4.8 presents the global performance metrics for C-GN, C-IN and their ablation counterparts. Autotelic agents using continuous goals but no curriculum clearly show an increased variance in their global performance. Figure 4.8 zooms on the local performance on each class for the considered agents. Compared to C-GN and C-IN, the shaded areas in the ablations are larger, suggesting that the learning process of the latter agents is not stable. Precisely, this is true in stacks of size 3 or higher. In fact, ablations face catastrophic forgetting as they engage with harder goals. The curriculum learning



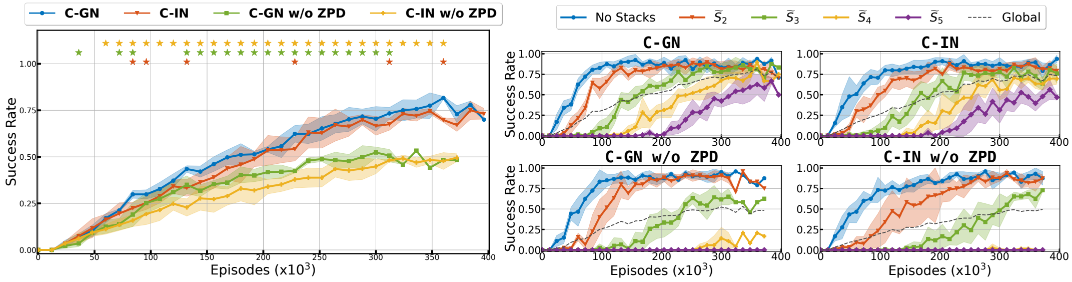
(a) Global SR. Stars highlight statistical differences w.r.t C-GN agents.

(b) Per class SR.

Figure 4.8: Performance metrics for C-GN, C-IN and their curriculum ablations. Mean ± standard deviations are computed over 5 seeds.

mechanism helps stabilize the learning process by focusing on goals of moderate level of complexity, including the ones that the agents are likely to forget during training. Note that this issue is specific to continuous goals, which shows that they are not well suited to transfer between different goals.

Non-Trivial Scene Resets Ablation



(a) Global SR. Stars highlight statistical differences w.r.t C-GN agents.

(b) Per class SR.

Figure 4.9: Performance metrics for C-GN, C-IN and their ablations where non-trivial scene reset is removed. Mean ± standard deviations are computed over 5 seeds.

To assess the importance of the non-trivial scene reset scheme for continuous goals, we consider the C-GN and C-IN agents — the best performing GNN-based architectures so far — and remove the biased initialization scheme: blocks are placed without any initial stacks in the resulting ablations. Figure 4.9 shows performance metrics for these agents. The global SR of both ablations increases slower than that of C-GN and C-IN (Figure 4.9b). Besides, it gets stuck at around 50% of the maximal performance while their corresponding full versions manage to reach 75%. Zooming on the per-class performance metrics shows the considerable decrease in the capability to reach complex goals when removing the non-trivial reset scheme

(Figure 4.9b): the ablations struggle to transfer between easy goals ($\widetilde{S}_2$ and $\widetilde{S}_3$) and harder ones ($\widetilde{S}_4$ and $\widetilde{S}_5$).

Transfer Capabilities

Transfer capabilities

To investigate the transfer capabilities of the GNN-based agents, we consider the best performing architectures with semantic goal spaces: S-GN, S-IN and S-RN (See the global performance metrics). Tables 4.2 and 4.3 respectively show the global and the per class performance metrics for the considered agents. The values presented in these tables correspond to the evaluation of the training policies on the held-out goals once the training is stabilized.

Table 4.2: Global SR metrics, averaged over 5 seeds.

Agents	Scenario 1	Scenario 2	Scenario 3
S-GN	**0.93 ± 0.04**	0.78 ± 0.09	0.28 ± 0.14
S-IN	0.89 ± 0.01	**0.82 ± 0.09**	0.47 ± 0.13
S-RN	0.89 ± 0.02	0.68 ± 0.12	**0.64 ± 0.10**

Table 4.3: Per class SR metrics, averaged over 5 seeds.

Scenario-Class	S-GN	S-IN	S-RN
1 - S_2 & S_2	**0.97 ± 0.02**	0.92 ± 0.04	0.96 ± 0.03
1 - S_2 & S_3	**0.89 ± 0.04**	0.87 ± 0.01	0.82 ± 0.07
1 - P_3 & S_2	**0.93 ± 0.05**	0.88 ± 0.04	0.90 ± 0.05
2 - P_3	0.80 ± 0.09	**0.85 ± 0.10**	0.67 ± 0.14
2 - P_3 & S_2	0.75 ± 0.12	**0.80 ± 0.10**	0.68 ± 0.11
3 - S_3	0.28 ± 0.14	0.47 ± 0.13	**0.64 ± 0.10**

Transfer Scenario 1. All the considered agents are good at transferring to combinations of constructions that they have encountered separately during training (Column 2 of Table 4.3). This probably results from both the representational power of GNNs and the self-attention aggregation schemes during both edge and node updates. On the one hand, all these graph-based architectures are permutation-invariant. On the other hand, for a goal within one of the combination classes, each construction is independent from the other. Consequently, agents would attend to one of them, accomplish it, and then focus on the other, as they learned to construct them separately during training.

Transfer Scenario 2. During training within this scenario, agents never encounter nor train on configurations involving pyramids. In other words, an object is never placed simultaneously on top of two other objects that are close to each other. First, we compare S-RN to both S-GN and S-IN. S-RN has lower overall transfer capabilities in this scenario compared to S-GN and S-RN (Column 1 of Table 4.2). They struggle in transferring to held-out goals from both P_3 and P_3&S_2 classes (Table 4.3). On the one hand, S-RN agents bypass the node update step which aggregates, for a particular node, the flowing information from the other nodes. Consequently, they tend to focus more on pairwise relations and less on the global relations of a particular configuration. On the other hand, a pyramid involves three objects. In fact, to be able to put an object i on top of j and k, the latter two need to be close to each other. As a result, exclusive edge updates without allowing information to flow between the different nodes is insufficient. Second, we compare S-GN to S-IN. As the former concatenates the global features to the input features of the edge update, the size of its corresponding network would be greater. This would probably lead to overfitting on the training data and transferring less to new situations.

Transfer Scenario 3. During training, agents never encountered a configuration involving a stack of 3 objects. Column 3 of Table 4.2 shows that S-RN outperforms both S-GN and S-IN in this scenario. This can be explained in a threefold fashion. First, we defined the *above* predicate as being *directly above*. Hence, a goal from the class S_3 would involve only two above predicates of the form $above(i, j)$ and $above(j, k)$ — which are fed to two different edges — to be true. Second, S-RN agents focus on pairwise relations that exist within edges. In other words, if they have learned to stack two blocks, they are capable of sequentially performing two independent stacks of two. Third, a stack of three objects can be constructed independently from the order of the stacks (i.e. first B above C then A above B or first A above B then A/B on C). By contrast S-GN and S-IN tend to transfer less as they overfit more on the training data due to their heavier computational scheme.

4.2.8 Conclusion

In this section, we studied several GNN-based goal-conditioned architectures for both the policy and critic in multi-object manipulation domains. More specifically, we considered four different computational schemes: full graph networks, interaction networks, relation networks and deep sets. We evaluated our agents using two different goal space structures: 1) continuous geometric goal spaces corresponding to per-object target positions; 2) semantic relational goal spaces based on the binary predicates close and above. Finally, we studied the transfer capabilities in three different scenarios by assessing the performance of agents on different sets of held-out semantic goals.

Semantic and Continuous Goal Spaces

Autotelic agents benefit from the abstract structured representation within semantic goal spaces. In fact, our results show that, when dealing with continuous goals, full graph and interaction networks, which adapt the full graph computational scheme, are the best performing agents. For this particular type of goal space, information about all the objects needs to flow to each node in order to learn more complex goals. By contrast, with semantic goal spaces, performing the edge update step is sufficient to capture useful pair-wise relations between objects. In fact, relation networks perform on par with full graph and interaction networks. Finally, additional ingredients such as non-trivial scene resets and LP-based goal selection were necessary for agents to learn complex continuous goals. However, their counterparts that use semantic goals are more flexible as they do not need these additional ingredients.

Transfer Capabilities with Semantic Goals

Our transfer study suggests three main results. First, full graph, interaction and relation networks are all able to transfer to combinations of previously seen goals. Second, relation networks struggle in adapting to new goals that require reasoning about triplets of objects, such as building pyramids. These agents bypass the node update step, which seems to be crucial to pool information about all the nodes in the graph. Finally, relation networks outperform the other GNN-based architecture in transferring to goals of higher stacks. In fact, their light computational scheme enables them to overfit less on the training data. Consequently, they are more flexible in sequentially combining pair-wise skills.

This study helps understand the impact of key design choices towards open-ended autotelic agents capable of efficient transfer between abstract goals. However, the agents studied here only leverage their physical interactions with the environment. This does not account for the extraordinary human capacities to learn from social interactions [Vygotsky, 1978a, Bruner, 1973, Tomasello, 2005].

We believe adding social learning mechanisms as suggested in [Sigaud et al., 2021] is a promising line of research towards more capable open-ended agents. This is the object of the second part of the manuscript.

Chapter Summary

This chapter investigated the transfer and generalization capabilities in autotelic agents. This is important because the ability to learn in open-ended environments is a key property in teachable autotelic agents, as outlined in Chapter 2. Learning in open-endedness requires continuously acquiring new skills, which implies that the learning agents are flexible and their policy transfers well to new situations. We argued that such properties depend on the definition of the goal space representation and on the design of the agents' internal models, including their policy. We first introduced *Graph Neural Networks* (GNNs), a technical tool that bakes graph-based data into neural networks (Section 4.1). More specifically, we defined the founding components and computational schemes in GNNs, and presented an overview on their usage in the reinforcement learning (RL) setups. Then, we tackled the autotelic acquisition of skills using GNNs (Section 4.2). We compared different combinations of goal space representations and policy architectures in multi-object manipulation scenarios. On the one hand, we considered two types of goal spaces: continuous position-based goals and discrete predicate-based goals. On the other hand, we investigated four types of GNNs: *full graph networks*, *interaction networks*, *relation networks* and *deep sets*. We showed that endowing autotelic agents with relational inductive biases within their goal space (using the predicate-based goals) enables them to use less computationally expensive architectures. Besides, we showed that these architectures overfit more on the training goals than the more expensive ones, allowing the learning agents to transfer their skills to previously unseen goals.

Part Summary

The central objective of this part is to investigate the design of *autotelic agents* that could easily benefit from external teaching signals. We argued that such agents should represent their goals in a way that makes social interactions easier, and that they should use specific architectures that enable them to transfer between their learned skills and generalize to new ones. First, we started by confronting standard reinforcement learning (RL) to the open-ended learning challenges (Chapter 1). We suggested that combining goal-conditioned RL (GC-RL) and intrinsic motivations should enable agents to grow diverse repertoires of skills. Then, we introduced *teachable autotelic agents*, which are goal-based agents that can learn from their own goals and from external social interventions (Chapter 2). We outlined the key properties such agents should leverage, and argued that they are missing in standard RL algorithms. After that, we introduced *predicate-based autotelic agents*, a sub-family of autotelic agents that represent their goals as sets of binary spatial predicates (Chapter 3). We showed that these agents are able to efficiently learn diverse sets of skills and to ground language into their learned behavior. Finally, we investigated the transfer and generalization capabilities of artificial agents depending on their goal space representation and the design of their policies (Chapter 4). We showed that combining Graph Neural Networks (GNNs) with semantic predicate-based goal spaces allows better learning and transfer.

In the next part, we attempt to incorporate external social interventions within the training process of predicate-based autotelic agents. In Chapter 5, we present the autotelic exploration with external signals as *hybrid goal exploration processes*. In Chapter 6, we present a novel social interaction protocol where autotelic agents can efficiently couple their intrinsic motivations with external signals in order to promote their exploration and grow their repertoires of skills.

Part II

Teaching Autotelic Reinforcement Learning Agents

In the first part of the manuscript, we introduced *teachable autotelic agents* (TAAs), a family of goal-based artificial agents that can learn goals in an autonomous fashion or by leveraging external social signals from an expert social partners. We showed that *predicate-based autotelic agents* whose policies are modeled as *Graph Neural Networks* (GNNs) exhibit promising teachability potential. On the one hand, such agents benefit from the structure of goal spaces represented as sets of constraints based on binary spatial relations. This facilitates their language grounding and increases their behavioral diversity. On the other hand, they use object-centered representations to model their policies, enabling them to efficiently transfer between goals and increasing their sample-efficiency.

In the second part of the manuscript, we continue our study of TAAs and investigate how such agents can benefit from external guiding signals when training. More specifically, we consider autotelic exploration with social interventions as a *goal exploration process*, which we call *Hybrid Goal Exploration Process* (HGEP). We argue that such agents should be endowed with *internalization mechanisms* that enables them to take ownership of externally provided goals and *query mechanisms* to efficiently ask for assistance when they are not progressing alone. We also argue that social interventions should be minimal, limiting the burden on external social partners. Besides, they should help the learning agents *consolidate* their previously learned skills and progress towards their *zone of proximal development* (ZPD). We organize this part as follows:

- Chapter 5 formalizes HGEPs as a family of algorithms for exploring and learning about multiple goals from external and internal sources. We investigate the influence of such hybrid frameworks on the learning and exploration capabilities of artificial agents. We show that exclusively relying on internal goals might lead to poor exploration capabilities, especially in environments where random exploration fails. Actually, exclusively relying on internal goals limits the distribution of reachable goals to the distribution of discovered goals. Hence, additional tricks and techniques to enhance exploration should be used to avoid getting stuck in specific regions of the goal space. By contrast, including external sources of goals enables agents to discover new regions of their goal space, since their goal exploration process relies on goal babbling Rolf et al. [2010].

- Chapter 6 builds on the idea of HGEPs and introduces a novel family of autotelic agents that are able to *internalize* external goals and rehearse them in the absence of their social partners. Besides, these agents can estimate their learning progress, and use it to query the social partner for help whenever they get stuck. Additionally, we introduce *Help Me Explore*, a social interaction protocol where social partners build models of exploration limits of the learners. They use such models to first propose goals that are stepping stones towards novel goals, then continue with new goals that are easily reachable from these stepping stones. This defines a ZPD management scheme, where social partners progressively assist the artificial agents in discovering new regions of their goal space and growing their repertoire of skills.

Chapter 5

Hybrid Goal Exploration Processes

Autotelic agents aim at growing their repertoire of skills. This implies that they need not only to discover as many goals as possible, but also to learn to achieve each of these goals. When these agents evolve in environments where they have no clue about which goals they can physically reach in the first place, it becomes challenging to handle the *exploration-exploitation* dilemma. Actually these agents are usually trained to optimize a fitness measure with reference to the distribution of goals they have physically encountered. Yet this distribution shifts as they discover more and more goals. Consequently, such agents have to make sure they still perform well on what they have already encountered, but continue to grow their distribution of discovered goals. This becomes rapidly problematic in hard exploration environments with bottlenecks and sparse rewarding signals, and agents usually require additional tricks to overcome these exploration obstacles. Interestingly, Chapter 2 presents works in developmental psychology and education sciences highlighting the importance of *guided-play* in the early skill acquisition of infants. Human caregivers help toddlers overcome their exploration limitation by assisting them whenever it becomes necessary. Our goal is to introduce a goal exploration framework for autotelic agents that benefit from external signals while keeping their intrinsically motivated characteristics. This chapter introduces preliminary studies in which we introduce *Hybrid Goal Exploration Processes* (HGEPs), a novel family of algorithms that handle exploration of multiple goals from internal and external sources. We study the impact of having this type of coupled exploration on the learning and diversity of goals in different robotics environments.

5.1 Motivation

Goal-conditioned Reinforcement Learning (GC-RL) is a framework where artificial agents use reinforcement learning (RL) methods to learn a goal-conditioned policy that leverages multiple skills (see Section 1.3). The main objective of agents evolving within the GC-RL framework is to learn to reach as many goals as possible, thus growing their repertoires of skills. In doing so, these agents maximize their *control* over their environment, as they would be able to conduct specific series of actions that yield to any goal they have learnt. In particular, we consider scenarios where the goal space is known beforehand but the subset of reachable goals is unknown. In such scenarios, works usually consider the goal sampler — the module that generates physically reachable goals for the agents — to be either external or internal to the agents.

On the one hand, *externally motivated* (EM) agents are asked to pursue specific goals. They require an *external program* that generates, at each episode, a physically reachable goal for the agents to pursue. This program can simply consist in a hard-coded function within the environment that generates target features and passes them as *target goals* to the agent [Schaul et al., 2015, Andrychowicz et al., 2017, Plappert et al., 2018, Nair et al., 2018]. Furthermore, the external program can also be represented as a human instructor that uses language to express sets of constraints on the state space (e.g. *close the drawer*) in a more abstract and interpretable way [Luketina et al., 2019]. These agents are called *instruction-following* agents, and can be conditioned or language embedding or a more abstract set of semantic predicates as described in Section 3.3.

On the other hand, *internally motivated* (IM) agents are endowed with an internal goal sampler and pursue their own goals. We outlined in Section 1.5 that this family of agents can further be decomposed into two sub-families: 1) agents that know the set of reachable goals beforehand but organize their learning autonomously, and 2) agents that have no clue about the set of reachable goals. The first family of IM agents exclusively aim at maximizing their control over the set of reachable goals they already know. They organize their learning into efficient curricula, progressively moving towards more complex goals. These methods are broadly known as *Automatic Currciulum Learning* methods [Lanier et al., 2019, Li et al., 2019, Colas et al., 2019, Zhang et al., 2020]. The second family of IM agents aim at *exploring* their goal space and *controlling* it. They usually start with an arbitrary distribution of reachable goals that is close the distribution of initial state representations, and use noise-based exploration techniques to progressively cover their goal space [Florensa et al., 2018, Colas et al., 2020b, Akakzia et al., 2021]. However, the simple use of noise-based exploration can be insufficient, especially in hard environments where discovering complex goals requires long sequences of actions. To circumvent this issue, some approaches attempt to start exploring from sparsely visited regions by resetting the environment or by keeping counts of the number of visits for each state and returning to the least visited ones [Ecoffet et al., 2019, Pitis et al., 2020]. Other approaches, which use specific structured representations of the goal space, leverage imagination mechanism to combine different features of their previously visited goals and form new ones [Nair et al., 2018, Colas et al., 2020b].

Interestingly, research in psychology, philosophy and robotics have highlighted the stimulating role of socio-cultural environments in human development [Wood et al., 1976, Vygotsky, 1978b, Tomasello, 1992, Lindblom and Ziemke, 2002]. In Chapter 2, we discussed the design of goal-conditioned agents that follow their on IM but still benefit from external teaching signals. However, in the context of GC-RL, there is still no clear framework of how such agents would learn.

In this chapter, we introduce *Hybrid Goal Exploration Processes* (HGEPs), a family of algorithms that couple both competence-based intrinsic and external motivations within the learning agents. The denotation "hybrid" comes from the crossbreed of two different goal sources. HGEPs agents:

- Generate and pursue their own goals to explore their environment, discover and grow their repertoire of skills.
- Follow external goals generated by an external program. This goal is reachable but still not discovered by the agent.

In this chapter, we suppose the selection between following internal and external goals is arbitrary. Our main objective is to investigate the impact of having two different sources of goals on the agents learning and exploration capabilities. We leave the investigation of actively selecting between goal sources to Chapter 6

5.2 Related Work

Research in GC-RL has been investigating the idea of biasing the goal exploration process beyond the physically reachable set of goals. Mainly, many works studied *goal imagination*, *learning from expert demonstrations* and *learning from expert feedback*.

Goal Imagination. The goal space — which we consider to be predefined in our study — usually incorporates a specific structure that might in principle help extract topological relationships between goals. For instance, if the goal space corresponds to target positions in navigation tasks, the physically encountered goals give an idea on the support of the goal distribution. Thus, training generative models in this scenario can help produce new goals that are within the goal distribution's support but never encountered before [Florensa et al., 2018]. Similarly, approaches handling image-based goals also train generative models of image states to model goal distributions and support [Nair et al., 2018, Pong et al., 2019, Nair et al., 2020]. Other approaches make use of the compositionality of language to combine sentences and form new ones that they treat as goals [Colas et al., 2020b]. In such scenarios, imagined goals correspond to sets of constraints on the state space that were encountered separately before but never at the same time.

Learning with Expert Demonstrations. Expert demonstrations provide not only useful guidance for reaching hard goals, but also might unlock new regions of the goal space. In fact, decomposing a trajectory into sub-goals help show the way to the final goal, but these sub-goals might actually be new goals as well. Such an intuition motivates research in AI, which made use of expert trajectories to further extract new goals that were not physically encountered beforehand [Paul et al., 2019]. Other approaches trained generative models on expert trajectories to produce new ones, thus enabling their trained agents to overcome some hard exploration problems including bottlenecks [Ding et al., 2019].

Learning with Expert Feedback The external signals that GC-RL agents might receive can come in hindsight. That is, after performing a rollout where they target a specific goal, agents can receive feedback which can help guide their exploration. Namely, learning can be guided by hindsight instructions, preferences or external rewards that encourage agents to visit specific states rather than others [Christiano et al., 2017, Cideron et al., 2019, Röder et al., 2022].

5.3 Methods

We present our methods in this section. First, we state the problem of learning multiple goals in the standard scenario and extend it to the case of HGEPs (Section 5.3.1). Then, our new framework that leverages the exploration of goals coming from both internal and external sources (Section 5.3.2).

5.3.1 Problem Statement

We consider the problem of learning multiple goals within a fixed environment. We consider the following augmented MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0, \mathcal{G}, p^d_{\mathcal{G}}, \phi\}$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$ and $\mathcal{G}$ respectively design the state space, the action space, the environment transition distribution and the goal space, ρ_0 and $p^d_{\mathcal{G}}$ respectively design the distribution of initial states and the distribution of desired goals, ϕ designs a tractable mapping function that maps a state to a specific goal embedding. Recall that $\mathcal{G} = Z_{\mathcal{G}} \times R_{\mathcal{G}}$, where $Z_{\mathcal{G}}$ denotes the space of goal embeddings and $R_{\mathcal{G}}$ the set of goal achievement functions associated with the goal space $\mathcal{G}$.

We define the problem of learning multiple goals within the MDP $\mathcal{M}$ as a Goal-Conditioned Reinforcement Learning (GC-RL) problem: learn a goal-conditioned policy $\pi \ : \ \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow [0, 1]$ that maximizes the cumulative reward over the distribution of desired goals:

$$\pi^* = arg \max_{\pi} \ \mathbb{E}_{\substack{g \sim p^d_{\mathcal{G}},\ s_0 \sim \rho_0 \\ a_t \sim \pi(.\ \mid\ s_t, g) \\ s_{t+1} \sim \mathcal{T}(s_t, a_t)}} \Big[\sum_t \gamma^t R_{\mathcal{G}}(\phi(s_{t+1}) \mid z_g) \Big] \tag{5.1}$$

In EMGEPs, the distribution of desired goals is defined beforehand. It consists of the goals generated externally from the agent during the training. However, in IMGEPs, the distribution of desired goals is not defined beforehand. It corresponds to the goals that the agent physically discovers. Hence, this distribution changes over time depending on the agent's current exploration.

In this study, we suppose that the goals on which goal-based agents can train can originate either intrinsically from the set of physically discovered goals, or externally from a predefined set of goals. We denote by $q \in [0, 1]$ the probability of a goal being sampled from external sources. Consequently, the training goal distribution we consider is the following:

$$p_{goal} = (1 - q) \times p_{dis} + q \times p_{ext}, \tag{5.2}$$

where p_{dis} and p_{ext} denote the distributions of respectively the physically discovered goals and the externally predefined goals. The problem we aim to solve in this study is to find the optimal policy of Equation 5.1 where goals are sampled under p_{goal}.

5.3.2 Hybrid Goal Exploration Processes

In this section, we introduce Hybrid Goal Exploration Processes (HGEPs), a framework for learning multiple goals that come from both internal and external sources. Figure 5.1 illustrates the goal exploration procedure within HGEPs.

The *bootstrapping phase* is conducted to initialize the external and internal goal spaces, denoted respectively by $\mathcal{G}_{ext}$ and $\mathcal{G}_{int}$. On the one hand, the definition of $\mathcal{G}_{ext}$ depends on an external program, which can be for example a simulated function within the environment, a simulated embodied agent, or a human instructor. This external program has external goals, which can be represented in a different way from that of the agent (for example, using language, images or a set of binary constraints). These goals are passed through an interpretation module which translates them into a representation that is understood by the agent. On the other hand, the initialization of $\mathcal{G}_{int}$ depends on an arbitrary policy π_ϵ, which interacts with the environment during a fixed amount of time. These interactions produce trajectories, which are passed to an outcome extraction module which extracts the goals that were arbitrarily discovered. The distribution of such goals is usually close to the distribution of goals extracted from initial states (through the probability distribution ρ_0.

The *babbling loop* is conducted for a sufficient amount of time to allow the agent to cover entirely the goal space. It consists in iteratively applying the following cycle. First, a goal space is selected via a *selector*. The latter controls whether the upcoming episode would pursue an internal goal (discovered physically by the agent) or and external goal (proposed by the external program). Although we can think of scenarios where agents actively use the selector as an internal module to query

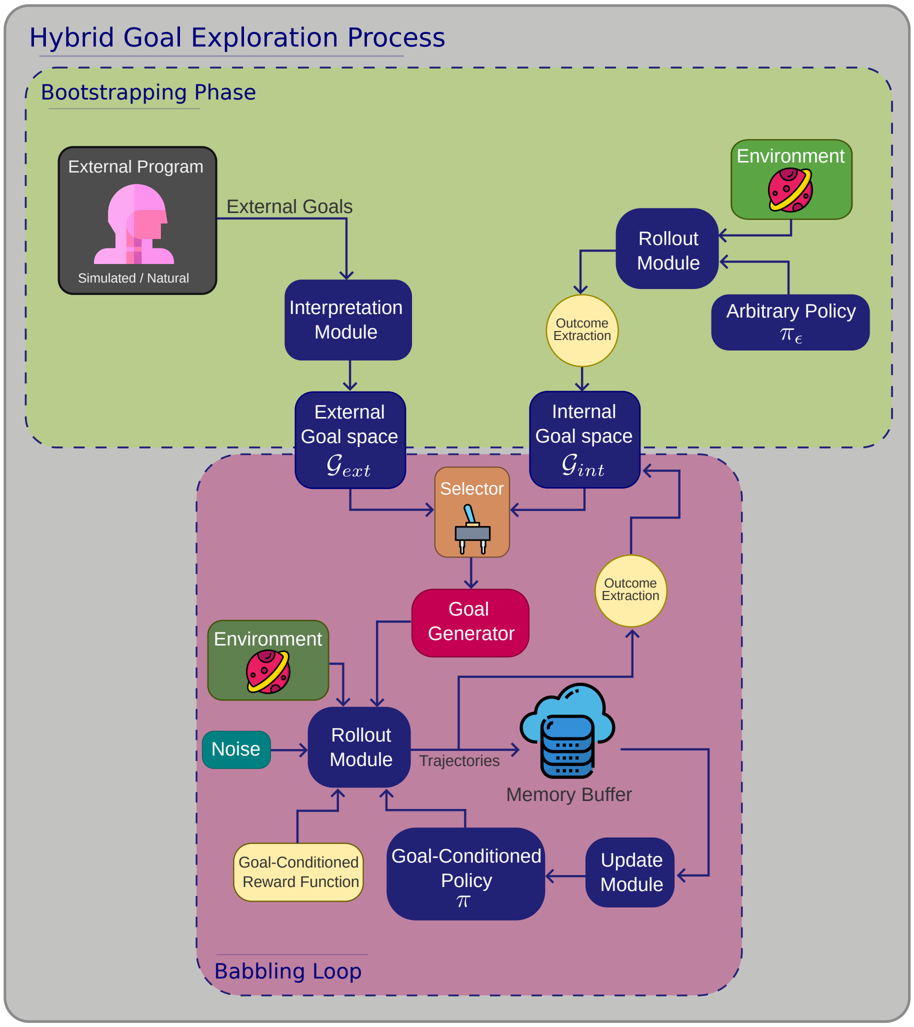


Figure 5.1: Illustration of Hybrid Goal Exploration Processes (HGEPs)

external help when needed, we focus on this study on arbitrary selectors. Once the source of goals is selected, a goal is generated via a goal generator, and the agent is ready to act within the environment using its goal-conditioned policy π and the goal-conditioned reward function, which consists of the goal achievement function associated with the selected goal. Note that noise is added to the actions in order to push the agent towards discovering new goals. After performing a rollout, the underlying trajectories are stored in a memory buffer. Besides, they are passed to the outcome extractor in order to extract the new goals that the agent has discovered during the performed rollout. Finally, transitions are sampled from the replay buffer and used to update the agent's internal models, typically its policy π.

5.4 Experimental Setup

This section is organized as follows. In Section 5.4.1, we present the multi-goal environments that we considered in our experiments. For each environment, we describe the training procedure, including the goal generation process. In Section 5.4.3, we present the evaluation metrics that we use to assess the relative importance of using hybrid goal generation sources, and we motivate our choice for each metric. Finally, in Section 5.4.4, we present and discuss the results that we obtained. For further details and additional results, we refer the reader to Appendix D.

5.4.1 Environments

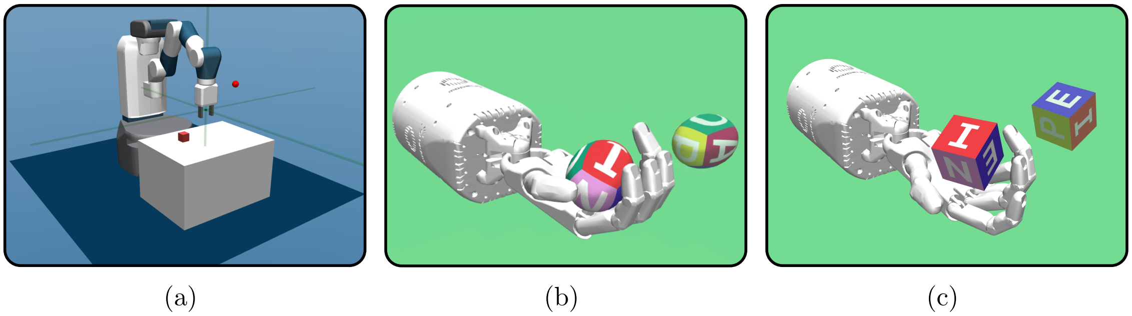

(a) (b) (c)

Figure 5.2: Considered environments: (a) FetchPickAndPlace-v0 (E1); (b) HandManipulateEggFull-v0 (E2); (c) HandManipulateBlockRotateXYZ-v0 (E3).

Figure 5.2 illustrates the three environments that we considered in this study. All these environments are part of OpenAI Gym [Brockman et al., 2016] and use the MuJoCo physics engine for fast and accurate simulation [Todorov et al., 2012].

Block Pick and Place. We consider the *FetchPickAndPlace-v0* environment (E1) (Figure 5.2a), which is part of the *Fetch* environments [Andrychowicz et al., 2017]. The agent is modeled as a 7-DoF robotics arm with a two-fingered gripper. The goal is the target position represented by its 3-dimensional Cartesian vector. The target position describes where we want the agent to place the object. It can be in any place that is physically reachable by the manipulated block. The agent obtains a reward of 1 whenever it succeeds in placing the block in its target position, and 0 otherwise. The actions that the agent can take are 4-dimensional, describing the desired gripper movement in Cartesian coordinates (3 dimensions) and the opening and closing of the gripper (1 dimension). The observation space includes features of the agent's body as well features of the manipulated object. As for the agent's body, position and velocity of the gripper are considered. As for the manipulated object, the Cartesian position, the velocity and the Euler angles-based rotation.

Egg Manipulation. We consider the *HandManipulateEggFull-v0* environment (E2) (Figure 5.2b), which is based on the Shadow Dexterous Hand [1]. The agent controls an anthropomorphic robotic hand with 24 degrees of freedom. An egg-shaped object is placed on the palm of the hand. The goal is represented as a 7-dimensional vector including a target position of the egg (3 dimensions) and a target rotation (4 dimensions). The task of the agent is to move the object to its target position and to rotate it in order for it to match a specific target orientation. The agent obtains a reward of 1 if the object matches the goal in terms of both the position and the rotation, and 0 otherwise. The actions that the agent can take are 20-dimensional for all non-coupled joints of the hand. The observation space includes features of the hand and of the object. As for the hand, we consider positions and velocities of all the joints. As for the object, we consider the position, rotation and velocity. For more details, we refer the reader to the original paper introducing the environment [Plappert et al., 2018].

Egg Manipulation. We consider the *HandManipulateBlockRotateXYZ-v0* environment (E3) (Figure 5.2c). The agent is exactly identical to the one used in E2. However, the used object is a cubic block. The goal vector is also 7-dimensional, as in E2. However, for simplicity, the Cartesian coordinates are ignored when defining the goal achievement function. Consequently, the task of the agent is only to rotate the block in order to match a specific target orientation. This choice makes our study easier, since object geometry makes a significant difference in the complexity of the problem (manipulating a block is harder than manipulating and egg-shaped object).

5.4.2 Training Procedure

For all the environments that we consider in our experiments, the training procedure is the same and it follows the following scheme. First, if agents are IMGEPs or HGEPs, they first conduct a bootstrapping phase where they discover easy goals using an arbitrary policy. We fix the amount of the bootstrapping steps to make sure that agents start with a diversified set of initial goals. Concretely, we run the arbitrary policy for 9.5k episodes, which represents 10% of the total training budget in E1 and 2.5% in E2 and E3. Second, the agents engage in a babbling loop for a fixed amount of time (100k episodes for E1 and 350k for E2 and E3). The bubbling phase is organized in the following cycle:

- A goal is selected to target. It can be a self-generated goal or an externally generated one. Self-generated goals are uniformly sampled from the running buffer of goals physically discovered by the agents. Externally generated goals are sampled by the simulator, they are not necessarily physically encountered but are valid goals.

[1] https://www.shadowrobot.com/products/dexterous-hand

- The agent performs rollout for each sampled goal. Actions are taken according to the agent's policy. For exploration, the agent performs random action with a probability $\epsilon = 0.3$. The underlying rollouts are stored in a replay buffer.
- Transitions are sampled from the replay buffer and used to update the policy and the critic. For replay, we use the Hindsight Experience Replay (HER) replay scheme [Andrychowicz et al., 2017]. As an RL algorithm, we use the Deep Deterministic Policy Gradient (DDPG) algorithm. We made this choice to decouple exploration from policy updates (instead of entropy-based methods).

Additional details about the training procedure and implementation are depicted in Appendix D.

5.4.3 Evaluation

We are interested in assessing the learning capabilities and exploration of agents that use hybrid goal exploration processes. To this end we define 4 variants of HGEPs which use different ratios of external goals. Namely, we denote by HGEP X% the hybrid agents that sample external goals during training with a ratio of X%. The values of X we consider in this study are 5, 10, 20 and 50. We also consider two baselines: the IMGEP baseline (similar to HGEP 0%) and the EMGEP baseline (similar to HGEP 100%).

To evaluate the performance of these agents, we consider the following metrics.

Success Rate on External Goals (External SR). The success rate (SR) corresponds to the average of success of the agent. It is computed over a fixed number of episodes, where a different goal is pursued at each episode. We are interested in the SR on the external goals: that is, goals that are defined by an external distribution defined within the simulator. This distribution is diverse and describes all the goals that the agent can potentially learn. It reflects the ability of the agent to master its whole goal space.

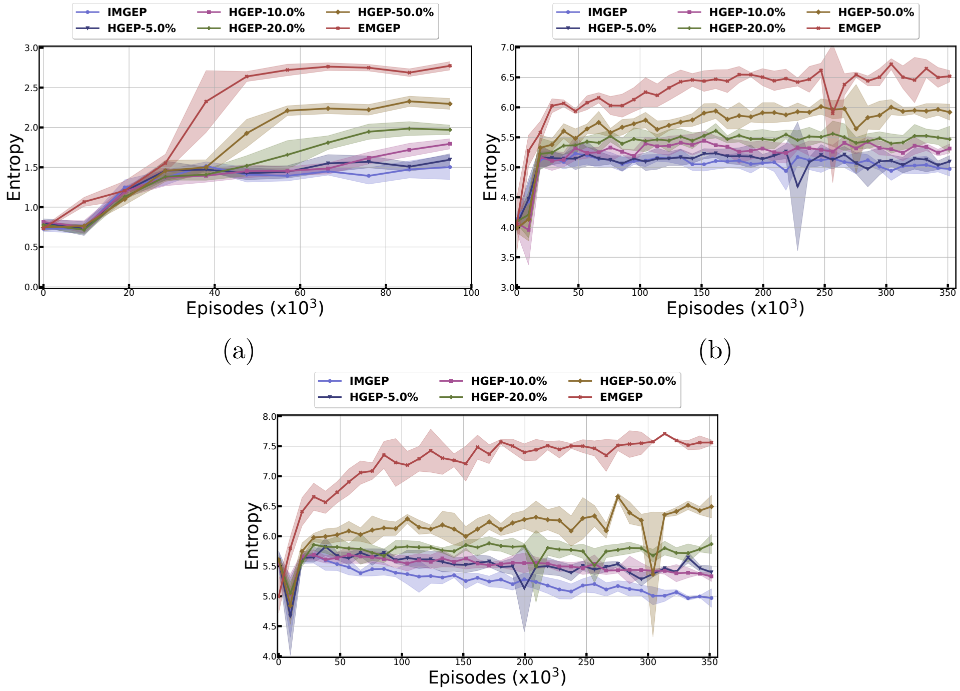
Entropy of Discovered Goals Distribution. We are interested in measuring the task-agnostic exploration of the agents we consider in this study. To this end, we consider the entropy of the discovered goals distribution as a proxy. More specifically, we consider the goals discovered by the agents as random variables generated by an unknown distribution, and we aim at evaluating the *Shannon Entropy* of these random variables [Shannon, 1948]. Since goals are high-dimensional and continuous, we can't compute exactly the Shannon entropy. However, we estimate its value using the Kozachenko-Leonenko k-nearest neighbour estimator [Kozachenko and Leonenko, 1987]. The basic idea is to consider the set of discovered goals during training as observations, and evaluate the average distance between neighbouring data points. The larger is the distance, the larger is the dispersion of data points and this the larger is the entropy [2].

Distribution of Discovered Goals. We are interested in visualizing the distribution of the goals that are discovered by the agents. Since this is not trivial for the hand manipulation environment, we only focus on the fetch pick and place environment. We plot the positions in which the block was placed by the agent.

5.4.4 Results

In this section, we depict the results of our experiments on the three considered environments. Our goal is to answer the following questions: How does HGEPs affect the distribution of discovered goals ? What does this distribution look like ? How does HGEPs affect agents' mastery of the goal space ?

How do HGEPs affect the entropy of discovered goals distribution ?



(a)

(b)

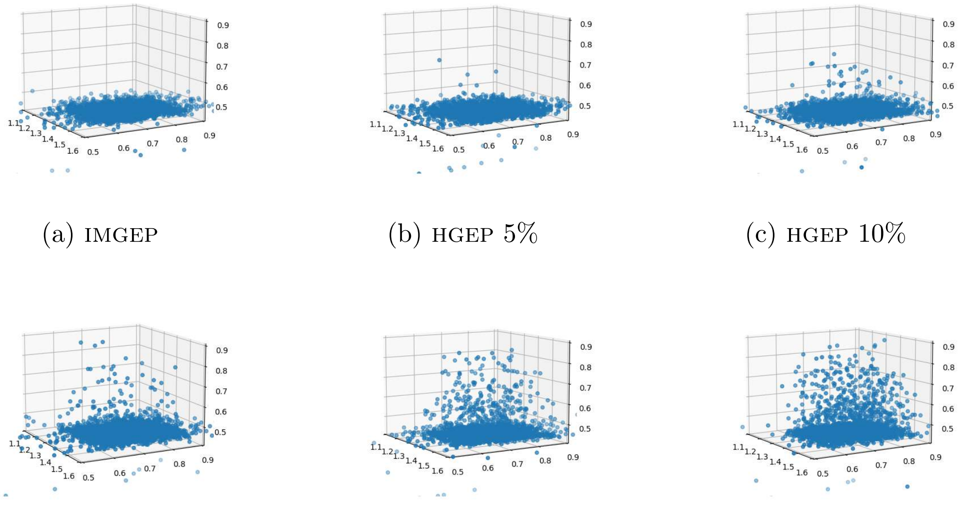
(c)

Figure 5.3: Entropy of the discovered goals distribution in the (a) E1; (b) E2; (c) E3.

[2] https://github.com/paulbrodersen/entropy_estimators

To quantify the diversity of discovered goals, we consider the entropy metrics. Figure 5.3 shows the estimated entropy for each agent on each of the considered environments. The value is computed with reference to the discovered goals at each epoch — that is every 1900 episodes. In all the considered environments, augmenting the ratio of external goal generation helps increase the entropy. This means that when agents pursue goals that were provided to them externally, they are most likely to physically encounter and discover new goals that they have not discovered yet. Agents that do not use external goals (IMGEP) or use only a few portion (HGEP 5% and HGEP 10%) find there entropy stuck at a plateau (Figures 5.3a and 5.3b) or decreasing (Figure 5.3c). In fact, these agents keep attending to the distribution of goals they have physically discovered, which is narrow especially at the beginning of the training. Besides, as training advances, policies become less stochastic and the distribution of physically discovered goals might become even narrower, hence the decreasing entropy in the case of E3.

How are the discovered goals distributed ?



(a) IMGEP

(b) HGEP 5%

(c) HGEP 10%

(d) HGEP 20%

(e) HGEP 50%

(f) EMGEP

Figure 5.4: Distribution of discovered goals during the last epoch of training (1900 episodes) in environment E1.

We are interested in visualizing how the physically discovered goals are distributed. To do so, we only consider the fetch pick and place environment E1, since it is easier to plot the 3d positions of the block in this environment. Figure 5.4 highlights the distribution of discovered goals during the last epoch of training for the different agents we considered in this study. These results confirm that agents which focus exclusively or mostly on physically discovered goals have their discovered goal distribution narrower than the others. In fact, the goals discovered by the IMGEP agents are concentrated in the plane $z = 0.45$ (Figure 5.4a), which corresponds approximately to the height of the table in the fetch pick and place environment. As the ratio of external goals increases, the agents discover more goals in the air.

How do HGEPs affect the mastery of the goal space ?

Table 5.1: Success Rate on externally provided goals for the different agents.

IMGEP (SR)	HGEP 5 (SR)	HGEP 10 (SR)	HGEP 20 (SR)	HGEP 50 (SR)	EMGEP (SR)
Environment E1					
0.63 ± 0.10	0.74 ± 0.11	0.83 ± 0.12	0.89 ± 0.04	**0.99 ± 0.01**	**0.99 ± 0.01**
Environment E2					
0.93 ± 0.05	0.87 ± 0.03	0.97 ± 0.03	0.92 ± 0.03	0.99 ± 0.01	0.97 ± 0.03
Environment E3					
0.53 ± 0.08	0.50 ± 0.04	0.51 ± 0.12	0.53 ± 0.10	**0.63 ± 0.09**	**0.67 ± 0.15**

Table 5.1 depicts the SR on the goals that are externally generated. Recall that these goals are sampled with a hard-coded function within the simulator that makes sure to cover the entire space and provides diverse sets of goals. Results show that for environments E1 and E3 (first and third row in Table 5.1), the SR increases as the amount of external goals provided to the agents increases. In fact, agents that perform goal babbling using new goals that they have not encountered before are more likely to discover new ones. This increases the diversity of discovered goals, which are the goals that the agents use to train their internal models (policy and critics). Consequently, these agents become more successful when dealing with external goals. Interestingly, the difference in SR metrics in E2 is not significant (row 2 in Table 5.1). Recall that E2 corresponds to the manipulation of the egg-shaped object. Moving the hand's fingers arbitrarily would most likely move the object and rotate it (contrarily to a cubic-shaped object, where rotating the object requires more dexterous hand manipulation). As a result, the physically discovered goals during the bootstrapping phase are already diverse enough to enable agents to train on an master different goals.

5.5 Conclusion and Discussion

We presented a new family of algorithms for learning multiple goals called *Hybrid Goal Exploration Processes* (HGEPs). HGEPs enable goal-based artificial agents to learn from multiple goal sources: either from internal goals that they autonomously generate based on their internal goal sampling module, or from external goals that they receive from an external program. The latter can either be a hard-coded goal generation module within the environment, which generates valid goals, or a human instructor that suggests goals to the learning agents. We compared HGEPs to *intrinsically motivated goal exploration processes* (IMGEPs) and to *externally motivated goal exploration processes* (EMGEPs). More specifically, we were interested in assessing the exploration and learning capabilities of hybrid agents compared to IMGEPs and EMGEPs. We considered 3 control environments. Our results suggest that, depending on the complexity of exploration within the environment, external goals can help unlock new regions of the goal space, since the agents wouldn't have babbled these goals without external signals. This seems to be important in endowing agents with the capacity to overcome bottlenecks, and represents a step towards leveraging the *exploration-exploitation* trade-off in *goal-conditioned reinforcement learning* (GC-RL). The goal source selection module that we considered in our experiments was arbitrary. In other words, agents have no control whatsoever on the source of goals from which they draw during training. We believe this deprives artificial agents from their *autonomy.* Allowing artificial agents to evaluate their need for external assistance, and to actively select queries whenever they are stuck, is an open challenge in RL with a human in the loop. In the next chapter, we attempt to tackle this issue.

Chapter Summary

The central question we wanted to tackle in this chapter is the following: *How can we formally define the interaction between a goal-conditioned artificial agent and a social partner as a goal exploration process ?* To this end, we proposed *Hybrid Goal Exploration Process* (HGEP), a family of algorithms for exploring and learning multiple goals. HGEPs couples self-supervised goal generation and external goal generation, which might be an important backbone for *teaching autotelic agents.* Actually, the latter need to juggle between their intrinsic motivations external teaching signals. HGEPs involve a *goal selection module* which plays the role of a *switch* controlling the source of goals. In the standard definition that we presented in this chapter, HGEPs use an arbitrary goal selection module. At the beginning of an episode, agents flip a coin independently of their training status, they then decide whether to self-generate a goal, or to follow an externally generated goal.

In the next chapter, we want such agents to be more *intrinsically motivated* when choosing the source of goals. We propose to base the external goal proposals on *active queries* based on the agents' own estimation of their *learning progress.* Besides, we want the externally proposed goals to be adequate to the agents' current capabilities.

Chapter 6

Guiding Exploration in Autotelic Agents

In the quest for autonomous agents learning open-ended repertoires of skills, most works take a *Piagetian perspective*: learning trajectories are the results of interactions between developmental agents and their *physical environment*. The *Vygotskian perspective*, on the other hand, emphasizes the centrality of the *socio-cultural environment*: higher cognitive functions emerge from transmissions of socio-cultural processes internalized by the agent. In this chapter, we argue that both perspectives could be coupled within the learning of *autotelic agents* to foster their skill acquisition. We contribute a novel social interaction protocol called *Help Me Explore* (HME), where autotelic agents can benefit from both individual and socially guided exploration. In *social episodes*, a social partner suggests goals at the frontier of the learning agent's knowledge. In *autotelic episodes*, agents can self-generate internal goals based on their own intrinsic motivations. We also endow these autotelic agents with *internalization mechanisms* enabling them to take ownership of goals proposed by the social partner regardless of whether they have physically reached them or not. Besides, we endow these agents with *internal goal source selectors*, enabling them to actively select queries to their social partners. The active query mechanism is build upon an internal *learning progress estimator*: agents ask for help from their social partners whenever they estimate that they are not progressing enough on the set of goals they already know.

6.1 Motivations

Artificial Intelligence (AI) strives to design embodied artificial agents that grow diverse repertoires of skills. These skills might be of distinctive complexity, ranging from easy ones (e.g. *move to the center of the playground*) to more challenging ones (e.g. *construct a pyramid next to a tower using building blocks*). Research in AI have supported the potential of *goal-conditioned reinforcement learning* (GC-RL) in developing such capable agents which learn a single policy to accomplish multiple goals [Schaul et al., 2015, Andrychowicz et al., 2017, Liu et al., 2022]. However, manually defining these goals before learning limits behavioral diversity and imposes a manual engineering burden, especially in rich environments with a high number of possible outcomes. Taking a peek at the field of developmental psychology, *open-ended learning* — a process of continuously learning skills of growing difficulty — seems to be of good inspiration. Actually, human toddlers *freely* engage in rich physical environments where they rely on their *intrinsic motivations* to uniquely establish their own intents and purposes.

Baking intrinsic motivations into GC-RL forms the family of *autotelic artificial agents* [Steels, 2004, Colas, 2021]. These agents need to represent, select and master self-generated goals [Colas, 2021]. Depending on whether they know their goal space and the set of reachable goals beforehand, there exist three scenarios for autotelic agents. First, if they know the goal space and the set of goals they can physically reach, their challenge is to smoothly self-organize these goals to progressively maximize their number of skills. This process has been coined "Automatic Curriculum Learning" (ACL) and relies on methods that use proxies such as learning progress or novelty to generate efficient learning curricula [Lopes et al., 2012, Bellemare et al., 2016, Burda et al., 2018, Colas et al., 2019]. Second, if they don't know the goal space, they need to discover skills and learn their own representations based on information theory methods [Eysenbach et al., 2018, Pong et al., 2019]. Third, if they know the goal space but have no clue about which goals they can physically reach, autotelic agents need to explore and discover skills by themselves [Ecoffet et al., 2019, Colas et al., 2020b, Akakzia et al., 2020b, Akakzia and Sigaud, 2022]. Consequently, the distribution of discovered goals is necessarily bound to the distribution of physically encountered goals [Pitis et al., 2020, Campos et al., 2020]. This is why such agents usually need additional features such as the ability to imagine new goals based on previous ones [Colas et al., 2020b], or to start exploring from sparsely visited regions of the goal space [Ecoffet et al., 2019, Pitis et al., 2020, Akakzia et al., 2020b]. However, imagining new goals is not always straightforward depending on the structure of the goal space, and sparsely visited regions are not necessarily the only stepping stones leading to new goals.

Interestingly, research in psychology, philosophy, linguistics and robotics have insisted on the important role of rich socio-cultural environments in human development [Wood et al., 1976, Vygotsky, 1978b, Berk, 1994a, Tomasello, 1999, Lindblom and Ziemke, 2003, Lupyan, 2012, Colas, 2021]. Actually, human toddlers are not only physical learners in the *Piagetian* sense — they learn to represent their goals through embodied interactions with their surroundings [Piaget, 1955] — but also benefit from their caregivers' external social signals. These signals construct a roadmap for toddlers in their *guided-play* experience [Weisberg et al., 2013, Yu et al., 2018], progressively *scaffolding* their previous skills to fashion new more complex ones. Caregivers smoothly carry toddlers into their *Zone of Proximal Development*, which is defined as the space between what a learner can do alone and what they can do with guidance or collaboration with more capable peers [Vygotsky, 1978b].

Recently, arguments have been made on the need of incorporating guided-play characteristics in the training process of artificial agents [Sigaud et al., 2021]. Actually, *teachable autotelic agents* is introduced as a family of autotelic agents that are both autonomous and can benefit from external social signals carrying their learning process. Such agents need to exhibit several *teachability properties*. Namely, they need to be endowed with some *internalization mechanisms* enabling them to rehearse external signals in the absence of their social partners (SP). Besides, they need to *actively communicate* with their caregivers, requesting help whenever needed.

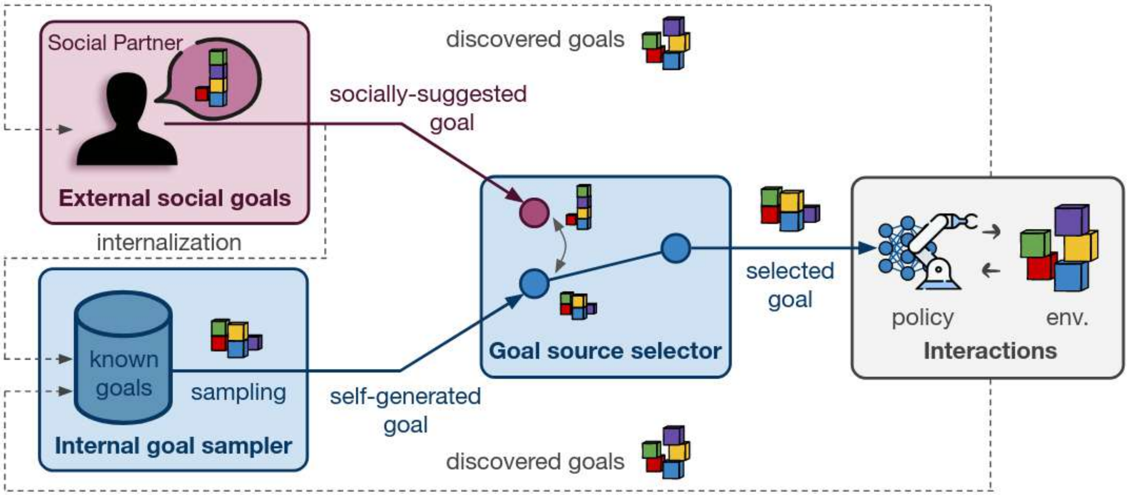


Figure 6.1: The HME social interaction protocol and the social autotelic agent. The autotelic agent (blue) interacts with its world and discovers new configurations (grey box, right). Through its goal source selector (blue box, middle), the agent decides whether to sample from its known goals (blue box, bottom left) or to query the social partner for a social goal (pink box, top left). Socially-suggested goals are further internalized within the agent for later use during individual episodes.

Contributions

In this paper, we present a practical framework for teachable autotelic agents. We formulate the problem of learning multiple goals in the presence of a caregiver as a *goal exploration* problem. We investigate the influence of externally proposed goals on both the learning process and exploration capabilities of autotelic agents. Our contributions are twofold:

- **H**elp **M**e **E**xplore (HME), a novel family of algorithms for learning multiple goals from both internal and external sources. HME couples individual autotelic learning and assisted social learning through goal proposals. It requires an expert SP who knows a set of reachable goals and is able to maintain a model of the current exploration capabilities of the learners. During *social episodes*, the

SP first suggests a stepping stone goal, which we call *frontier goal*. If reached, they chain with an adjacent goal that the learner never encountered, which we call *beyond goal*. During *autotelic episodes*, the autotelic agent autonomously selects which goal to pursue.

- **G**o**A**l-conditioned **N**eural **G**raph with **S**emnaT**ic** **R**epresentations (GANGSTR), an autotelic agent based on *Graph Neural Networks* (GNNs), endowed with an *internalization mechanism* and an *intrinsic goal source selector*. GANGSTR uses the Soft Acotr-Critic algorithm (SAC, Haarnoja et al. [2018]) as a backbone, where both its actor and critic are implemented as Relation Networks (RNs, Santoro et al. [2017]). On the one hand, the internalization mechanism enables GANGSTR to remember externally proposed goals even if they were not physically reached. On the other hand, GANGSTR automatically estimates its current learning progress and uses its intrinsic goal selector to actively query the SP for new goals.

When following HME, GANGSTR internalizes the beyond goals proposed by the SP even though it fails at reaching them. It can later autonomously *rehearse* these goals during autotelic episodes. Besides, GANGSTR aggregates its estimation of success over the physically discovered goals and uses it as a proxy to query the SP for new goals whenever it is not progressing enough. Our method is illustrated in Figure 6.1. We evaluate our approach on several mazes and block manipulation benchmarks. We show that, for the block manipulation environment with 5 objects, GANGSTR requires relatively few social interactions (0.5% of the total training time) to efficiently explore and learn about all the possible configurations in the goal space, including stacks of 4 objects, 5 objects and combinations of stacks and pyramids. We denote the combination of HME and GANGSTR by HME-GANGSTR.

6.2 Related work

In this section, we position our word with reference to *autotelic reinforcement learning*, *hard exploration problems*, *interactive reinforcement learning* and existing approaches that *combine social learning and intrinsic motivations*.

Autotelic Reinforcement Learning

The quest for autotelic agents first emerged in the field of developmental robotics [Steels, 2004, Oudeyer and Kaplan, 2007, Nguyen and Oudeyer, 2012, 2014b, Baranes and Oudeyer, 2013b, Forestier and Oudeyer, 2016b, Forestier et al., 2017] and recently converged with state-of-the-art deep reinforcement learning (RL) research [Eysenbach et al., 2018, Warde-Farley et al., 2018, Nair et al., 2018, 2020, Pong et al., 2019, Colas et al., 2019, 2020b,c]. Such agents rely on goal-conditioned RL algorithms [Schaul et al., 2015, Andrychowicz et al., 2017, Colas et al., 2020c] and automatic curriculum mechanisms [Portelas et al., 2020]. Following the Piagetian tradition in psychology, most of these approaches consider individual agents interacting with their physical environment and leveraging forms of intrinsic motivations to power their exploration and learning progress [Bellemare et al., 2016, Achiam and Sastry, 2017, Nair et al., 2018, 2020, Burda et al., 2018, Pathak et al., 2019, Colas et al., 2019, Pong et al., 2019]. Unless all their goals are manually defined, which usually represents an engineering burden, these approaches usually struggle to explore rich goal spaces that include goals requiring long sequences of specific actions to be discovered.

Hard Exploration Problems

Hard exploration problems refer to a family of domains where rewards are very sparse, deceptive or adversarial. Standard reinforcement learning (RL) methods usually perform poorly in such scenarios [Bellemare et al., 2016, Amodei et al., 2016], especially since random exploration is rarely efficient to discover successful states and obtain meaningful feedback. To circumvent this issue, a typical approach has been intrinsic motivation, where artificial agents self-generate intrinsic rewards to encourage exploration [Oudeyer and Kaplan, 2007, Barto, 2013, Bellemare et al., 2016]. Another line of research handled the hard exploration problem in a sequential fashion, where they first return to infrequently visited states and then explore from these states in search for new ones [Ecoffet et al., 2019, Pitis et al., 2020]. Other works investigate the design of evolutionary algorithms capable of discovering representations with good potential to produce a diversity of new ones [Cully and Demiris, 2017, Gajewski et al., 2019]. These representations are said to be of high *evolvability*, which is the idea of finding genomes that are very likely to produce diverse new solutions through simple mutations [Altenberg et al., 1994, Wagner and Altenberg, 1996, Ebner et al., 2001, Reisinger et al., 2005, Lehman, 2012, Mengistu et al., 2016]. For instance, these methods can focus their sampling procedure on states that recently led to discovering new ones [Cully and Demiris, 2017].

Interactive Reinforcement Learning

Interactive RL is a framework where artificial agents use RL algorithms to learn policies based on non-expert users who communicate and teach their preferences and expectations. It investigates and models the ways human tutors can guide their agents' learning process [Thomaz and Breazeal, 2008b, Argall et al., 2009, Celemin and Ruiz-del Solar, 2015, Najar et al., 2016, Christiano et al., 2017]. A category of teaching signals requires the SP to point out sequences of actions and states using demonstrations or preferences [Argall et al., 2009, Christiano et al., 2017, Fournier et al., 2019]. Another category assists the development of goal-based agents by providing them with guiding instructions [Grizou et al., 2013]. All these approaches usually rely on *direct teaching* methods, which depend too much on the tutor to drive the agent's behavior, depriving them from their autonomy.

Social Learning and Intrinsic Motivations

Combining social learning and intrinsic motivations has been studied as a way to enhance skill acquisition in artificial agents [Schaal et al., 2003, Thomaz et al., 2006, Thomaz and Breazeal, 2008b, Peters and Schaal, 2008, Kober and Peters, 2011, Stulp and Schaal, 2011, Nguyen and Oudeyer, 2014a]. The main motivation is that social guidance provided by humans can drive the learner to new regions of its state space, where its own random exploration alone is usually insufficient. Intrinsic motivations can then be used to further explore these new areas. Such a combination profits from both the broader aspect of intrinsic motivations (explore as many goals as possible) and the specialized aspect of social learning (follow a specific instruction). This was studied not only in the context of learning one single skill [Schaal et al., 2003, Peters and Schaal, 2008, Kober and Peters, 2011, Stulp and Schaal, 2011], but also in achieving a variety of goals [Thomaz et al., 2006, Thomaz and Breazeal, 2008b, Nguyen and Oudeyer, 2014a]. Most of these works rely on human demonstrations or upstream feedback as a form of social guidance.

6.3 Methods

The central objective of this paper is to study the *goal exploration* problem with two different sources of goals: *internal goals* that are autonomously generated by agents, and *external goals* that are provided by an expert SP. Besides, as part of the *internalization mechanism*, our agents remember and can self-generate external goals they failed to reach. In this section, we detail the implementation of our family of algorithms. First, we present an overview of our methods. Second, we state the problem we attempt to solve and provide useful definitions. Third, we explain how our agents select to query their SPs for assistance. Fourth, we present the exploration procedure that our agents follow during their training. Fifth, we describe which goals the SP proposes when queried by the agents. Finally, we present the learning algorithm and the architecture used in our study. Additional details about our methods and pseudo-codes can be found in Appendix E.

Overview

We propose to couple social learning and intrinsically motivated learning in the context of multi-goal learning in autotelic agents. To this end, we define a social interaction protocol, HME, which relies on an expert SP proposing specific goals that help scaffold the learning agents' skills. However, we want our agent, HME-GANGSTR, to deliberately choose to query for assistance whenever needed. This enables them to preserve the *autonomous* characteristic, and accounts for the *guided-play* process in human toddlers.

Initially uninformed about which goals they can physically reach within their goal space, HME-GANGSTR performs random actions and unlock easy goals. Once these goals are discovered, the interaction protocol can be triggered. Agents can follow either *autotelic episodes* where they autonomously select their goals from their own goal memory, or *social episodes* where their SP suggest specific goals based on the agents' current knowledge. Actually, the SP continuously builds a model of the agents' current exploration limits, which enable them to propose goals within the agents' ZPD —goals that they are not likely to achieve unless provided with help from experts. Furthermore, HME-GANGSTR is endowed with an *internalization mechanism* of the SP's goals, which enables it to rehearse these goals during the absence of the SP even though it has not physically encountered them yet.

Problem Statement

We model the problem of learning multiple goals within a fixed environment as a goal-augmented MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0, \mathcal{G}, p_{\mathcal{G}}, \phi\}$: $\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$ and ρ_0 respectively denote the state space, the action space, the environment transition distribution and the distribution of initial states; $\mathcal{G} = Z_{\mathcal{G}} \times R_{\mathcal{G}}$ denotes the *training goal space*, where goals are defined by pairs of goal embeddings in $Z_{\mathcal{G}}$ and goal achievement functions in $R_{\mathcal{G}}$ [Forestier et al., 2017, Colas et al., 2022b, Liu et al., 2022]; ϕ denotes a tractable mapping function that maps a state to a specific goal embedding.

Standard GC-RL approaches attempt to learn a goal-conditioned policy $\pi : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow [0, 1]$ that maximizes the discounted sum of rewards with reference to the training goal distribution $p_{\mathcal{G}}$:

$$J(\pi) = \sum_{t=0}^{H} \gamma^t \, \mathbb{E}_{\substack{g \sim p_{\mathcal{G}},\ s_0 \sim \rho_0 \\ a_t \sim \pi(. \ \mid \ s_t, g) \\ s_{t+1} \sim \mathcal{T}(s_t, a_t)}} \Big[R_{\mathcal{G}}(\phi(s_{t+1}) \mid z_g) \Big], \tag{6.1}$$

where $H < \infty$ denotes the finite time horizon and $R_{\mathcal{G}}$ compares the embedding of the next state ($\phi(s_{t+1})$) to the target goal embedding (z_g) in order to assess the achievement of the target goal. Standard GC-RL methods usually consider a fixed training goal space $\mathcal{G}$ and training goal distribution $p_{\mathcal{G}}$. By contrast, autotelic agents *only train on the goals they have discovered so far in training.* Consequently, in autotelic agents, the training goal space and the training goal distribution continuously change. We define a *training step* as sequentially performing N rollouts in the environments followed by M policy updates. We consider our agents as *goal exploration processes* that need to continuously discover new goals at each training step based on their previous experience and on external suggestions by their SP. The latter understand the environment in terms of the structure of the basic achievable goals (for example, they know one can go from a stack of two objects to a stack of three objects by simply adding a block on top). We call the *Assignment Goal Space* (AGS), the subspace of basic goals the SP is aware of. Besides, the SP can construct a model of their autotelic learner's current exploration limits by examining their behavior. Consequently, the distribution of the SP's suggested goals depends on the training step.

We denote by $\mathcal{G}_{SP}$ and $\mathcal{G}_a^i$ respectively the AGS and the discovered goal space by an agent at training step i. Furthermore, we denote by $p_{\mathcal{G}_a^i}$ and $p^i_{\mathcal{G}_{SP}}$ respectively the distributions of the agent's goals and its SP's suggestions at training step i. Note that the index i in $p^i_{\mathcal{G}_{SP}}$ is associated with the distribution and not the goal space, because the AGS is fixed but the way the SP suggests goals depends on the agent's current exploration limits.

How does HME-GANGSTR select queries ?

HME-GANGSTR uses its learning progress estimation (LP) as a proxy to decide whether it needs external assistance from its expert SP. The main principle is that, whenever its LP is low, it is most likely that HME-GANGSTR is not learning nor discovering anything new by itself. More specifically, HME-GANGSTR estimates its *global* LP by aggregating its LP for each of the goals it has discovered so far. We use a neural network parameterized by θ, which we note NN_θ. It takes as input a goal embedding z_g and produces an estimation of the success on the corresponding goal. We train NN_θ by sampling a batch of transitions from the replay buffer and optimizing the mean squared error loss

$$\mathcal{L}(NN_\theta) = \mathbb{E}_{\{s_t,a_t,s_{t+1},g\}_{t<H}\sim\mathcal{D}}\Big[\frac{1}{2}(NN_\theta(z_g) - R_{\mathcal{G}}(\phi(s_{H-1})|z_g))^2\Big], \tag{6.2}$$

where $\mathcal{D}$ denotes the replay buffer containing online trajectories. The LP for a single goal g at training time i is defined as $\text{LP}^i_k(g) = |NN^i_\theta(g) - NN^{i-k}_\theta(g)|$, where k denotes the window size [Colas et al., 2019, Akakzia et al., 2020b]. We compute the *query probability* at training time i by aggregating the LP estimations of goals in $\mathcal{G}_a^i$ ussing the following formula:

$$f^i_{\beta,k} = e^{-\beta\ \frac{1}{|\mathcal{G}_a^i|}\sum_{g\in\mathcal{G}_a^i}\text{LP}^i_k(g)}, \tag{6.3}$$

where β is a hyper-parameter defining the level of tolerance to stagnation in the LP. Actually, for higher values of β, the LP needs to be lower in order for agents to query the SP more often.

How does HME-GANGSTR explore ?

During each training step, HME-GANGSTR needs to select goals to pursue. First, it uses its estimated query probability $f^i_{\beta,k}$ to choose the goal source: either its own goals, or goals suggested by its SP. Consequently, the generated goal at training step i depends on the distribution of the agent's goals $p_{\mathcal{G}_a^i}$ and on the distribution of its SP's suggestions $p^i_{\mathcal{G}_{SP}}$. Formally, a goal g is sampled according to the following generative model:

$$g \ \sim\ f^i_{\beta,k} \times p^i_{\mathcal{G}_{SP}} + (1 - f^i_{\beta,k}) \times p_{\mathcal{G}_a^i}. \tag{6.4}$$

How does a SP suggests goals ?

The SP holds a model of the learning agent current exploration capabilities. Namely, whenever they are queried, they look at the set of goals that the agent has discovered so far and project them into their AGS. This enables them to determine which goals that the agent know are stepping stones to new goals. Consequently, they first select a stepping stone, which we refer to as a *frontier goal*, and suggest it to the learning agent. If the latter succeeds in reaching the frontier goal, the SP partner continues with an adjacent new goal, which we refer to as a *beyond goal* that the agent needs to reach starting from the frontier goal they just reached. This procedure describes how the SP take the agent into its ZPD, guiding it to states from which they can achieve harder goals. If the agent fails at reaching the beyond goal, they memorize it in the same buffer containing the goals it has physically reached. In this way, they can rehearse these goals later during autotelic episodes when the SP is absent.

How does HME-GANGSTR Learn ?

The goal-conditioned module in HME-GANGSTR is powered by the Soft Actor-Critic algorithm (SAC) [Haarnoja et al., 2018], which is a state of the art off-policy RL algorithm. We extend it to goal-conditioned scenarios, and train a goal-conditioned actor and two goal-conditioned critics to overcome the overestimation issues [Fujimoto et al., 2018]. Based on recent finds on the effectiveness of GNNs in leveraging good transfer capabilities [Akakzia and Sigaud, 2022], we model both the actor and the critic as relation networks. Finally, we use the Multi-Criteria Hindsight Experience Replay algorithm [Lanier et al., 2019] when retrieving transitions from buffers. This has showed to provide clear improvements to sample efficiency.

6.4 Experimental Setup

In this section, we describe the experimental setup that we consider to conduct our study on the influence on social interactions on the learning and exploration of autotelic agents. Our objective is to answer the following scientific questions: How do social episodes affect the performance of agents ? How do they affect their exploration? Can *automatic curriculum learning methods* (ACL) replace social interventions? How effective is the ZPD management procedure in HME? How do active queries affect the amount of social interventions?

This section is organized as follows. First, we describe the environment. Here we focus on a 5-object manipulation environment with a rich goal space of different levels of complexity. Evaluations of our methods on other maze-based environments are presented in Appendix E. Second, we describe the different variants of HME-GANGSTR and the baselines used for comparison. Third, we present the evaluation metrics on which we focus. Finally, we introduce and investigate the results.

6.4.1 Environment

We consider the *Fetch Manipulate* environment with 5 colored blocks used in [Akakzia and Sigaud, 2022]. Goal embeddings are represented by a set of constraints on the position of the blocks using the *close* and *above* binary predicates. To achieve a single goal, the agent — a 7-DoF robotic arm with a gripper as an end effector — needs to manipulate all the blocks so that all pairwise relations respect the constraints outlined in the goal definition. We also consider the unary predicate *on_table*, which is evaluated on each of the five blocks and assesses the presence or absence of contact between an object and the table. Consequently, each configuration of objects is represented by a single set of 35 constraints: 5 for the *on_table* predicate, 20 for the *above* predicate (one for each permutation of two objects) and 10 for the *close* predicate (one for each combination of two objects). Note that the resulting goal representation space is discrete and contains 2^{35} configurations, which are not all physically feasible (two objects can't be above each other on the same time). Hence, the agent has to discover which goals are physically reachable through sensorimotor interactions. At the beginning of each episode, the blocks are all set on the table without any pre-constructed stack.

6.4.2 Agents

In this section, we describe all the agents trained during this study.

Our Agents. Our agents couple autotelic and social episodes based on the query probability defined in Equation 6.3. This quantity is parameterized by β, which controls the level of tolerance to stagnation in LP: for lower values of β, agents are more likely to query for social interventions even if their estimation of LP fluctuates, while for higher values of β, these fluctuations need to be absent for queries to kick off. We denote these agents by HME-β (to simplify HME-GANGSTR-β). The values of β we consider in this study are within the list $[20, 50, 100, 200]$.

Influence of Social Interventions. To study the influence of social interventions on both the learning and exploration of artificial agents, we introduce two baselines: the *autotelic baseline* (AB), which does not use any external goals but rather exclusively rely on the self-generated ones, and the *social baseline* (SB), which is an instruction-following agent that only explores and learns goals suggested by the SP. Note that the autotelic and social baselines correspond to variants of our agents with respectively $\beta = \infty$ and $\beta = 0$.

ACL Study. We confront social interventions to ACL, where both influence goal selection during training of autotelic agents. To this end, we introduce two ACL baselines: 1)*Learning Progress* (LP) baseline: this baseline uses the neural network NN_θ, which predicts the success for each goal (see Section 6.3, to estimate the

LP for each goal. Then, the agent selects the goals with a high LP to train on and learn about. In practice, we sample among the 50 goals with the highest LP; 2) *Value Disagreement-based Sampling* (VDS) baseline: We train 3 Q-value networks to evaluate the difficulty of goals. During training, at the beginning of each episode, the agent samples 1000 goals in the list of discovered goals, evaluates their Q-values and computes a score reflecting the uncertainty about each goal. Applying softmax to the obtained scores yields a probability distribution that the goal sampler uses to more often select goals presenting higher uncertainty. Additional details are presented in Appendix E.

ZPD Management Study. The ZPD management of HME is based on the SP goals proposals. It involves proposing a goal within the frontier of the agents' exploration limits, that is the goals that they have discovered and that are stepping stones to new goals. However, one can think of several other definitions of frontier. In this study, we introduce two other definitions. First, we consider the set of goals that represent sparsely visited regions of the goal space [Ecoffet et al., 2019, Pitis et al., 2020], which we denote by F2. Second, we consider the set of goals with the highest LP. They represent goals that are neither too hard nor too easy for the current agents' policies [Baranes and Oudeyer, 2013b, Forestier and Oudeyer, 2016b, Florensa et al., 2018, Colas et al., 2019, Akakzia et al., 2020b]. We denote this set by F3. Based on these two definitions, we introduce two baselines F2&RANDOM and F3&RANDOM, within which agents first self-generate a goal in F2 or F3, and once there, they perform random actions. We also introduce a third baseline which we call UNIFOM&RANDOM, where agents first select a goal randmly, and once they reach it, they continue with random actions.

Active Query Study. To assess the advantages of using progress-based active queries, we introduce variants of our socially sensitive autotelic agents that send random queries to the SP. Mainly, for these agents, we consider a fixed query probability all along the training episodes. We consider three values for such variants: $P_{query} \in \{0.05, 0.06, 0.07\}$. We chose these values best on the best performing HME-based agents to match the same budget of social interventions.

6.4.3 Evaluation Metrics

In this section, we present the evaluation metrics that we considered in our studies. We train our agents for a fixed budget of time (20 hours). Evaluations are conducted each 50 cycles, where each cycle includes 2 episodes (100 episodes between each evaluation). More details about the implementation and hyper-parameters are presented in Appendix E. In all our experiments, means and standard deviations are computed over 5 seeds. Stars highlight statistical differences with reference to the far left agents (Welch's t-test with null hypothesis $\mathcal{H}_0$: no difference in the means, $\alpha = 0.05$).

Class	C_1	C_2	S_2	S_3	S_2&S_2	S_2&S_3	P_3	P_3&S_2	S_4	S_5
# Goals	10	45	20	60	60	120	30	60	120	120

Table 6.1: Semantic classes used for evaluations. The C_i class regroups all configurations where exactly i pairs of blocks are *close*. The S_i class contains all configurations where exactly i blocks are stacked. The P_i class contains all configurations where i blocks form a pyramid. The *&* is a logical *AND*. All unspecified predicates are *false*, thus classes represent disjoint sets.

Manipulation Performance. The subset of the 2^{35} expressible configurations the agent can physically reach is hard to compute *a priori*. To evaluate the agents, we hand-define 10 classes of reachable configurations, each containing between 10 and 120 configurations. This includes configurations where exactly i pairs of blocks are considered *close* (C_i), configurations containing stacks of size i (S_i), configurations containing pyramids of size 3 (P_3) and combinations of these. These classes are disjoint and their union does not cover the entire semantic configuration space (e.g. configurations S_2&C_2 are not contained). They do contain, however, classes that we consider *interesting*; in the sense that they can be easily described by humans (e.g. P_3&S_2 could be *"a tower of two and a small pyramid"*. See Table 6.1 for more details. At test time, the agent is given 20 goal configurations uniformly sampled from each of the 10 classes (200 goals). A test episode is considered a success when the final configuration matches the goal configuration. The measure of the agent's success is the *global success rate* (SR); the average of the 10 success rates per class. Testing episodes are conducted without exploration noise (deterministic policy) and are not added to the replay buffer (offline evaluations).

Exploration Performance. In order to propose social goals, the SP relies on their own assignment goal space $\mathcal{G}_{SP}$. In our experiments, we define $\mathcal{G}_{SP}$ to be a set of 321 goal configurations including the path from no stacks at all to stacks of 5 objects for each permutation of objects. We also define the *oracle Goal Space* $\mathcal{G}^*$, a more broader goal space containing 12666 goal configurations. It includes more intertwined paths, from stacks of 2 objects to stacks of two other objects for instance. For the social baseline, we use the oracle goal space $\mathcal{G}^*$ as the assignment goal space. We give more details about the construction of such goal spaces in Appendix E. We are interested in assessing the coverage of $\mathcal{G}_{SP}$ and $\mathcal{G}^*$. We consider the goals discovered by agents at the end of training, perform the intersection of these goals with either $\mathcal{G}_{SP}$ or $\mathcal{G}^*$, and report the ratio of covered goals. We also evaluate the distribution of goals achieved by the agents during training for several evaluation classes. Namely, we consider the classes that are more challenging to be discovered individually: S_3, $S_2\&S_3$, S_4 and S_5. Finally, we report counts of the proposed frontier and beyond goals by the SP. We consider the ratio beyond frontier, which gives an idea on how well agents succeed in following the SP's instructions.

6.4.4 How do Social Episodes affect Agents Performance ?

To assess the importance of social episodes in growing diverse repertoires of skills, we compare GANGSTR agents evolving within HME using different values of β to the autotelic baseline (AB) and the social baseline (SB). We focus on the global SR and the per class SR.

Table 6.2: Amount of Social Interventions and global SR of the studied agents.

Agents Metrics	Social	HME-$\beta = 20$	HME-$\beta = 50$	HME-$\beta = 100$	HME-$\beta = 200$	Autotelic
% Social Episodes	100	51.62 ± 2.49	**6.98 ± 0.58**	0.5 ± 0.06	0.001 ± 0.001	0
Global SR	0.81 ± 0.01	0.92 ± 0.01	**0.93 ± 0.02**	0.87 ± 0.03	0.72 ± 0.03	0.75 ± 0.05

Global Performance. In Table 6.2, we report the percentage of social episodes conducted during the training of AB, HME-GANGSTR and SB. We also present the global SR for each of these agents. Note that for higher values of β, the amount of social interventions is lower. On the one hand, AB and SB struggle in maximizing their SR and get stuck respectively at 0.75 and 0.81 (respectively far left and far right columns). On the other hand, HME-based agents with sufficiently low values of β manage to further increase their SR. Namely, for β valued at 100, 50 and 20, the amount of social interventions is valued respectively at 0.5%, 6.98% and 51.62%. These agents receive enough social guidance, and by coupling autotelic and social episodes, they manage to grow their repertoires of skills and master new goals. Namely, for HME-$\beta = 50$, the global SR reaches its highest value of 0.93. In the next paragraph, we zoom at which new skills get unlocked with social interventions.

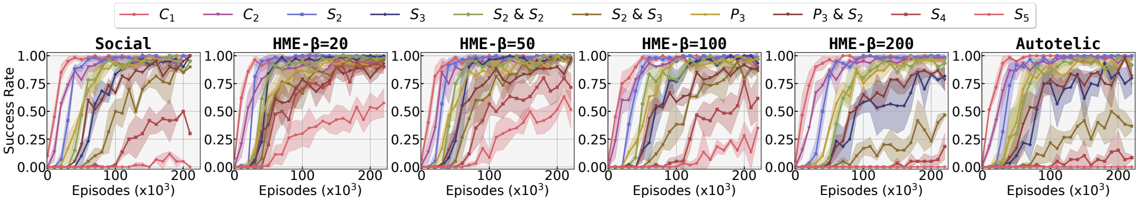


Figure 6.2: Per class SR across training episodes for different ratios of social interventions.

Per-class Performance. Figure 6.2 shows the per-class performance of AB, SB and HME-GANGSTR with different values of β. On the one hand, we note that AB and HME-$\beta = 200$ agents struggle in learning the most complex goals including the classes S_4 and S_5. On the other hand, SB seems to perform relatively better on these classes. However, the SB is slow and less sample efficient than HME-$\beta = 20$, HME-$\beta = 50$ and HME-$\beta = 100$ which manage to faster learn the more complex goals. This suggests that social interventions are responsible for increasing the performance on some evaluation classes, and that only 0.5% of social episodes is sufficient to learn about more complex classes.

6.4.5 How do Social Episodes affect Agents Exploration ?

In this section, we study the influence of social episodes on the exploration of agents. We investigate the effect of social episodes on 1) the coverage of the assignment goal space and the oracle goals space; 2) the distribution of goals that the agents encounter during training.

Table 6.3: Coverage of the Assignment and Oracle goal space.

Agents / Metrics	Social	HME-$\beta = 20$	HME-$\beta = 50$	HME-$\beta = 100$	HME-$\beta = 200$	Autotelic
Coverage $\mathcal{G}_{\mathcal{SP}}$	0.22 ± 0.04	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.97 \pm 0.01}$	0.66 ± 0.01	0.1 ± 0.001	0.10 ± 0.01
Coverage $\mathcal{G}^*$	$\mathbf{0.36 \pm 0.02}$	0.21 ± 0.01	0.20 ± 0.13	0.19 ± 0.01	0.18 ± 0.08	0.14 ± 0.02

Coverage of $\mathcal{G}_{\mathcal{SP}}$ and $\mathcal{G}^*$. Table 6.3 depicts the coverage of the assignment goal space $\mathcal{G}_{\mathcal{SP}}$ and the oracle goal space $\mathcal{G}^*$ for AB, SB and the different HME-GANGSTR agents. On the one hand, we note that HME-$\beta = 20$ and HME-$\beta = 50$ agents manage to maximize their coverage of $\mathcal{G}_{\mathcal{SP}}$, outperforming all the other agents. On the other hand, SB hold the best coverage of $\mathcal{G}^*$, outperforming AB and the HME-based agents. This is not surprising, as SB uses the whole oracle goal space as an assignment goal space when proposing goals by the SP. Interestingly, however, HME-$\beta = 20$ and HME-$\beta = 50$ agents show better coverage of $\mathcal{G}^*$ than AB.

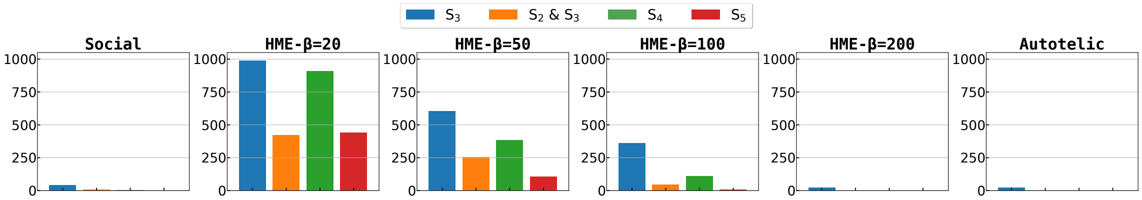


Figure 6.3: Distribution of goals achieved by the agents during training.

Distribution of Achieved Goals. Figure 6.3 presents the distribution of goals achieved by the agents during training. We focus on the more complex goals that are not trivial to be discovered in the first place. Namely, we consider the classes S_3, $S_2\&S_3$, S_4 and S_5. On the one hand, we note that AB, SB and HME-$\beta = 200$ fail to discover and achieve these goals. On the other hand, HME-$\beta = 20$, HME-$\beta = 50$, and HME-$\beta = 100$ agents are able to relatively achieve these goals more often. Note that HME-$\beta = 50$ actually achieves these goals many times with only 6.98% of social episodes (see Table 6.2). In the remainder of our study, we consider these agents and compare them to other baselines.

6.4.6 Can Automatic Curriculum Methods replace Social Interventions ?

In this section, we compare HME-$\beta = 50$ to two ACL methods. Namely, we are interested in assessing the global SR of HME-based agents and the ACL baselines we considered in our study.

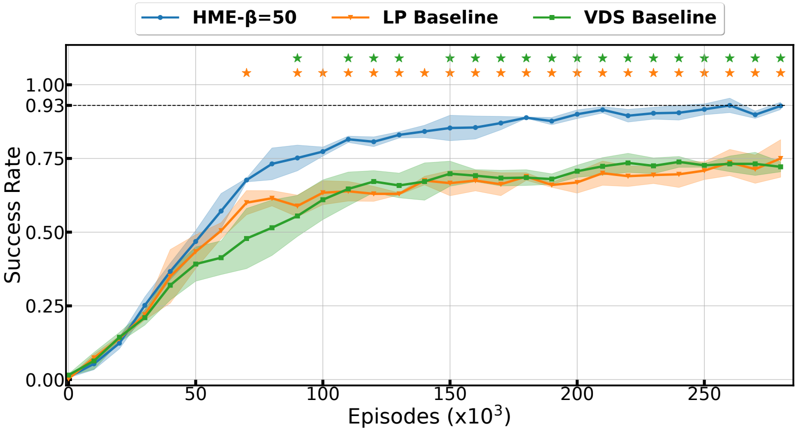


Figure 6.4: Global SR for HME-$\beta = 50$, the LP baseline and the VDS baseline.

Figure 6.4 presents the global SR for HME-$\beta = 50$, the LP baseline and the VDS baseline. The HME-$\beta = 50$ (blue curve) agent outperforms both ACL baselines. Actually, ACL methods get stuck at a local minima and struggle to further grow their repertoires of skills. This suggests that ACL methods are unlikely to replace social interventions. While ACL methods usually focus on the goals that are the most challenging for the learning agents' current policy — goals that are neither too hard nor too easy — these goals do not necessarily represent stepping stones that unlock new ones. By contrast, the social interventions leveraged by HME-$\beta = 50$ focus on goals that are certain to yield new ones, based on the SP expertise.

6.4.7 How effective is HME's Zone of Proximal Development Management ?

In this section, we investigate the role of the ZPD management scheme compared to other methods of exploration. Namely, we consider the UNIFOM&RANDOM, F2&RANDOM and F3&RANDOM baselines. We are interested in evaluating the global SR and the distribution of goals achieved by these agents compared to HME-based agents.

Global SR. Figure 6.5 presents the global SR for HME-$\beta = 50$ as well the three baselines considered in this study. Notably, HME-$\beta = 50$ outperforms all the three baselines. This suggests that the ZPD management induced by the SP within HME

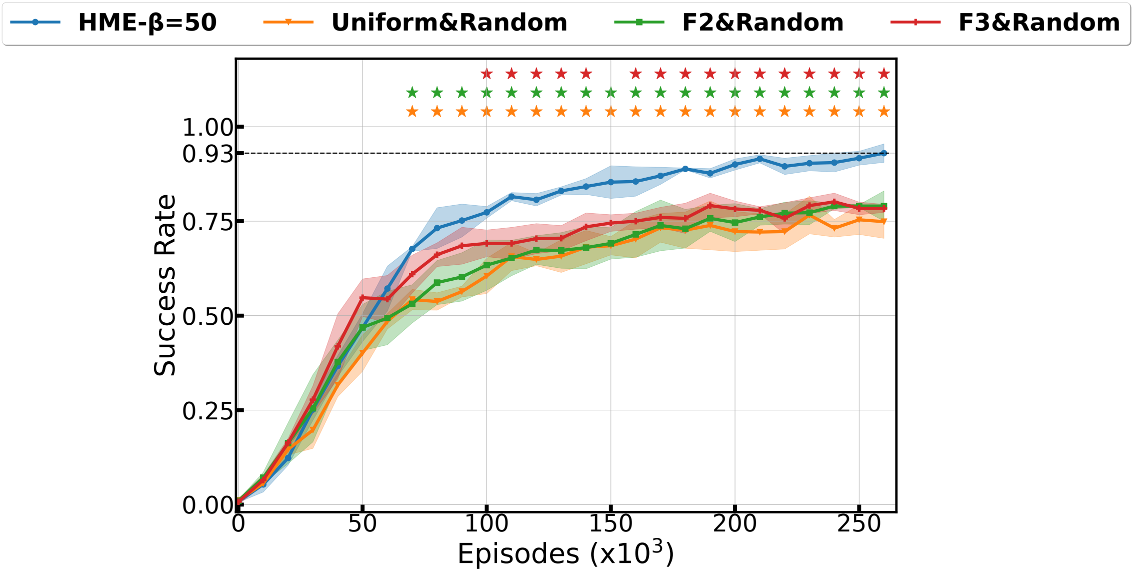


Figure 6.5: Global SR for different exploration techniques across training episodes.

is more efficient than randomly exploring from sparsely visited regions of the goal space (F2&Random) and from goals of intermediate difficulty (F3&Random).

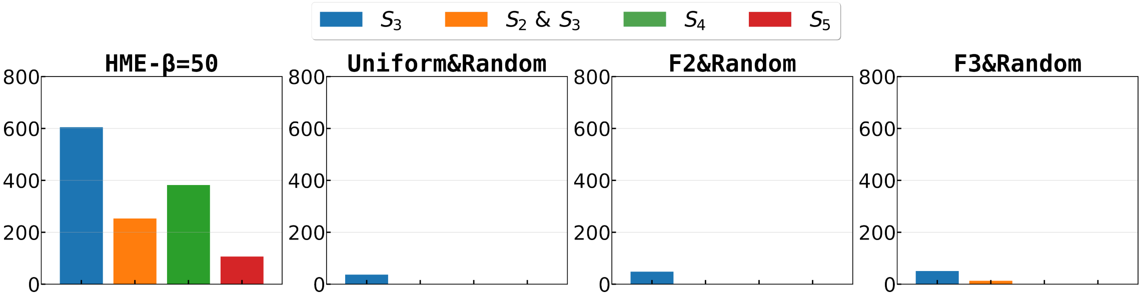


Figure 6.6: Distribution of goals achieved by the agents during training.

Distribution of Achieved Goals. Figure 6.6 presents the distribution of achieved goals. The Unifom&Random, F2&Random and F3&Random baselines manage to discover a few goals of stacks of three objects (blue bar). However, they fail at discovering the other more complex configurations. The fact that they also fail to achieve the stacks of three more often suggests that these configurations are not sampled often enough by these baselines. In fact, these agents perfectly master stacks of three, and focus their goal generation on other goals without realizing that exploring from S_3, even if it is perfectly mastered, would eventually lead to discovering new goals (stacks of 4 and 5).

6.4.8 How do Active Queries affect Social Interventions ?

In this section, we investigate the role of the active query mechanism based on the LP estimations used by HME-based agents to ask for the SP's assistance. To do so, we consider three baselines with fixed query probabilities: F-0.05, F-0.06 and F-0.07. These values were chosen so that the underlying amount of social interventions would be close to that of HME-$\beta = 50$.

Table 6.4: Evaluation metrics for the influence of the active query mechanism.

Agents / Metrics	HME-$\beta = 50$	F-0.05	F-0.06	F-0.07
% Social Episodes	6.98 ± 0.58	6.53 ± 0.27	7.75 ± 0.31	9.47 ± 0.09
Global SR	0.93 ± 0.02	0.91 ± 0.05	0.92 ± 0.03	0.91 ± 0.05
# Beyond / # Frontier	**0.25 ± 0.01**	0.23 ± 0.03	0.23 ± 0.03	0.19 ± 0.03
Proposed S_5	**1526 ± 317**	1718 ± 288	1884 ± 151	1706 ± 156
Achieved S_5	**106 ± 33**	71 ± 12	87 ± 15	138 ± 11

Table 5.3 highlights several metrics evaluated for HME-$\beta = 50$, F-0.05, F-0.06 and F-0.07. Namely, we report the amount of social episodes (first row), the global SR (second row), the ratio of beyond goals and frontier goals proposed by the SP (third row), the number of goals in S_5 proposed by the SP (fourth row) and the number of times agents achieved these goals (fifth row). Unsurprisingly, all the considered agents maximize their global SR (second row). However, the ratio of beyond goals divided by frontier goals is relatively higher in HME-$\beta = 50$ (third row). This suggests that the queries in HME-$\beta = 50$ are selected at a better time than the other baselines, allowing the agent to achieve the beyond goals more often from the first time. Besides, the SP proposed less goals from the S_5 class (fourth row), which is considered complex and the agent achieved these goals more often than the baseline with the same amount of social interventions (fifth row). We note that F-0.07 manages to achieve more of these goals, but this is not surprising since they receive more social interventions (first row). These results suggest that, for the same amount of social interventions, it's better to actively select queries based on LP estimation to enable agents to benefit more from these external signal.

6.4.9 Conclusion and Discussions

Recently, a lot of efforts have been made to endow artificial agents with the capacity to learn from humans. In this chapter, we considered the skill learning of autotelic agents immersed in a social interaction protocol as a *goal exploration process*: in order to grow their repertoires of skills, autotelic agents need to efficiently explore their goal space, using both their intrinsic motivations and their *social situatedness*. On the one hand, we argued that such agents should be endowed with some *internalization mechanisms* that enables them to remember external goals even though they have not encountered them physically yet. Besides, for such agents to still be autonomous, they need to deliberately select when to ask their social partner for help. To this end, we proposed to further equip them with an *internal goal source selector* which they use to select queries. On the other hand, we claimed that social interventions need to follow certain patterns such that they take the learning agent beyond its *zone of proximal development*, which represents the space between what the agent can fulfill alone and what it can fulfill with the help of expert caregivers. In this section, we discuss some limitations of our method.

Nature of Social Interventions

In HME-GANGSTR, the social partner (SP) intervenes during social episodes by *suggesting goals* whenever the agent seeks help. For simplicity, we supposed that goals expressed by the agent and SP have the same representation, and considered this representation to be based on predicate-based semantic configuration. In practice, if we want to model the SP by a real human, exhaustively defining a semantic configuration is not always trivial, especially if the number of objects to manipulate is high. Actually, the SP has to specify all the binary predicates between all pairs of objects. A more natural way to suggest goals would be to use partial goals: the SP does not have to specify all the predicates, but rather a small subset of constraints. For example, the SP does not have to specify that two objects would be close if they are stacked. This inter-dependency between predicates should alleviate the burden of specifying goals by the SP.

One of the easiest ways to suggest goals by the SP is to use natural language rather than providing sets of binary constraints. Natural language should in principle provide the SP with more flexibility. In fact, they do not have to specify a particular goal (i.e. a well defined configuration of objects), but can suggest more abstract goals by providing instructions that describe a particular arrangement of objects without specifying their identity (for example: *Build a pyramid*). We can imagine that the learning agents are endowed with a *language goal generator* (see Chapter 3). They use an instruction to generate a whole set of compatible goal configurations, from which they pick one to pursue. When achieved, the SP selects a beyond goal depending on the frontier goal that the agent selected itself, based on the natural language-based instruction.

Combining Social Interventions and ACL methods

ACL methods rely on intrinsic motivations to regulate the goal generation process in autotelic agents. These intrinsic motivations usually rely on the interaction data gathered in play. By contrast, social interventions rely on the SP expertise, including a wider knowledge of the goal space. In this chapter, we argued that ACL methods are unlikely to replace social interventions in order to explore unknown goal spaces, especially if the goal discovery requires specific sequences of actions. We claim that this is due to the fact that ACL methods focus on goals which do not necessarily represent stepping stones to new regions of the goal space. These stepping stones might be very easy goals — which would be ignored by ACL methods — but with good potential to unlock new goals. This is particularly true when goals are not continuous.

Social interventions and ACL methods could work in a complementary fashion. While ACL methods would help the learning agent self-organize its goals, social interventions would help it discover new ones. This can improve the sample efficiency in autotelic agents, and improve the coverage of their goal space.

Query Selection

HME-GANGSTR is endowed with an internal goal source selector which enables them to choose to either follow their own goals during autotelic episodes, or ask the SP for new goals to pursue during social episodes. The goal source selection is based on the estimation of a query probability: the probability to ask the SP for help. It is high if the agent is not showing enough progress on its already known goals. The progress on these goals is computed as the mean of progress on each goal separately. This aggregation can rapidly become inaccurate if the size of the discovered goals set increases. Actually, if most of the goals that the agent has discovered are mastered, the query probability would be low even though some other goals remain not mastered yet.

We may want to model the query probability differently to leverage small subsets of goals where the learning progress is not high enough. A possible solution is to compute the mean on the top N goals where the learning progress is high, where N would be a hyper-parameter. However, this might be inaccurate at the early stages of the training where the number of discovered goals is still low (close to or below N). Automatically tuning this hyper-parameter, and increasing it with the number of discovered goals might help. However, depending on the environment, this might become more or less difficult to tune. Another solution is to incorporate an estimation of the discovery rate to the modelling of the query probability. In this case, selecting queries would not only depend on the learning progress, but also on how many goals the agent is discovering. If the agent is discovering many goals, and even though it is performing well on these newly discovered goals, it would not ask the SP for help. Querying for social intervention would happen more often if the agent is not discovering any new goals.

Chapter Summary

This chapter makes a step towards *teachable autonomous agents*: the cognitive development of artificial agents must take its root in both social situatedness and the free exploration of the physical world [Sigaud et al., 2021]. To this end, this chapter introduces two contributions: 1) **G**o**A**l-conditioned **N**eural **G**raph with **S**emna**T**ic **R**epresentations (GANGSTR), an autotelic agent endowed with an internalization mechanism and an internal goal source selector, and 2) *Help Me Explore* (HME), a social interaction protocol where the SP efficiently catalyzes the development of the learner. GANGSTR embodies object-centered and relational inductive biases (GNN). It relies on both its intrinsic motivations and on its internalized goals to autonomously explore its goal space. When interacting within the HME interaction protocol, a simulated social partner (SP) simply suggests goals at the frontier of the agent's knowledge. When these goals are reached and are good stepping stones for further exploration, the SP goes further and suggests unknown goal configurations—adjacent nodes just beyond that frontier. These goals represent the agent's *zone of proximal development* (ZPD): the set of goals it cannot reach alone (unknown configurations), but can reach with social guidance (when the SP shows the stepping stone). Although low-level exploration remains powered by undirected noise, the high-level exploration is extended by the suggestions of the SP. This simple exploratory curriculum unlocks the potential of the agent. With only 0.5% of social episodes, it can discover pyramids and stacks of 3, stacks of 4 and stacks of 5.

Part Summary

The main scientific question we attempted to investigate in this part is the following: *How can we define social interaction protocols that allow efficient teaching of autotelic agents with light external interventions?* To this end, we first started by formalized the open-ended skill learning in autotelic agents under the guidance of an expert social partner (SP) as a *goal exploration process* augmented with a *goal source selector* that enable agents to switch between pursuing self-generated goals and goals proposed by an external program. In Chapter 5, we introduced *Hybrid Goal Exploration Processes* (HGEPs), a family of frameworks for exploring multiple goals from different sources. We showed that, when autotelic agents have no clue on what goals they can physically achieve and thus exclusively focus their intrinsic motivations to *physically encountered goals*, the external program suggests new opportunities for exploring novel regions of the goal space. More specifically, by targeting these external goals, autotelic agents are more likely to take new actions that they would not necessarily take when pursuing exclusively their own goals. We argued that synchronizing these different sources of goals could enable agents to explore more efficiently.

In Chapter 6, we proposed a way to perform this synchronization between self-generated and external goals. More specifically, we introduce *Help Me Explore* (HME), a social interaction protocol involving a social partner with expert knowledge on the underlying goal space and an autotelic agent, GANGSTR, endowed with an *internalization mechanism* and an *internal goal source selection module*. On the one hand, the SP intervenes by suggesting a *frontier goal* that the agent knows. This is made possible since the SP incrementally builds a model of the learning agent's current exploration limits. If GANGSTR succeeds at reaching this first goal, the SP continues with a new *beyond goal* that the agent does not yet know but that is accessible from the previous frontier goal. Following this scheme, the SP guides the agent beyond its *zone of proximal development* (ZPD). On the other hand, GANGSTR can remember the beyond goals proposed by the SP even though it has not physically encountered them yet. Then, it can rehearse these goals even in the absence of its caregiver. Besides, GANGSTR actively selects when to query the SP for help based on its estimation of its *learning progress*. Actually, if the agent estimates it is not currently progressing enough on its already discovered goals, it asks for *social episodes* where the SP performs the scheme described above. Otherwise, the agent performs *autotelic episodes*, where it autonomously generates goals to pursue based on its own sensorimotor experience and internalized goals.

We showed that when evolving within HME, GANGSTR efficiently explores and learns about its goal space, including the most complex goals. We also showed that the internal mechanisms of GANGSTR make the social interventions lighter. We believe this is a step towards teaching autotelic agents under social guidance with minimal involvement from the external caregiver.

Part III

Discussions

Chapter 7

Behavior Specification in RL

In Chapter 1, we introduced the reinforcement learning (RL) paradigm as a mathematical backbone to train artificial agents to accomplish a single task specified by a pre-defined reward function. We argued that the RL formulation provides the agent with information about *what* to do — the goal — but not necessarily about *how* to do it — the behavior — especially in sparse scenarios. We then presented a novel family of parametric criteria called *delayed geometric discounting criteria* as a generalization of the standard geometric discounts. Depending on the level of the delay, we showed that optimizing such criteria affects the agent behavior, as it becomes ready to discard short-term rewards for longer-term ones.

In this chapter, we discuss the problem of *behavior specification* in RL. We present three related methods: *constrained RL* (c-RL), *imitation learning* (IL) and *learning from human preferences* (LfHP).

Aspects of RL's Impracticality

The standard formulation in RL aims at maximizing the sum of returns obtained during the agent's interactions with its environment [Sutton and Barto, 2018]. This implies that a scalar reward function specific to the desirable task needs to be manually engineered beforehand. In addition, this reward function needs to be exhaustively defined if we want the learning agent to exhibit a particular behavior. A recent paper by Silver et al. [2021] argues that *reward is enough* for the emergence of sophisticated intelligence and the associated abilities. The authors suggest that even sparse rewards could enable an agent to learn a diversity of skills without manually specifying them. For example, if an agent receives a positive reward only if it places an object in a certain position, it would potentially be able to learn other skills such as pushing, sliding and grasping the object, even though the designer had not specified these underlying skills beforehand. Although this might be plausible in simple environments, it rapidly becomes challenging in more complex ones that incorporate several constraints and require specific reasoning and actions. In these scenarios, sparsely defined rewards would probably yield a sample-inefficient training. Consequently, the designer would need to define richer reinforcing signals, and even in this case, the underlying behavior is prone to negative behavioral side effects due to reward hacking [Amodei et al., 2016, Taylor et al., 2016, Everitt and Hutter, 2016, Krakovna et al., 2018]. A classic example of such behavior is the OpenAI's demo[1] of an RL agent in a boat racing game endlessly going in circles and collecting reward targets instead of actually performing the race.

Interestingly, a rebuttal paper entitled *scalar reward is not enough* by Vamplew et al. [2022] was presented as a response to Silver et al. [2021]. The authors suggest that scalar rewards could trigger some intelligent behavior in some cases, but they are insufficient for the development of more sophisticated artificial intelligence. They argue in favour of multi-objective models of reward maximization. We believe this is more plausible, especially knowing that RL lacks practical ways of specifying admissible and forbidden behaviors.

Constrained RL

The idea of C-RL first originated from the formulation of constrained MDPs, which is a framework that extends optimality within MDPs to the cases where additional constraints need to be satisfied in addition to the main objective [Altman, 1999]. Hence, C-RL belongs to the family of algorithms that leverage multi-objective models of reward maximization. A typical approach in C-RL is to use *Lagrangian methods* [Achiam and Sastry, 2017, Ray et al., 2019, Stooke et al., 2020, Zhang et al., 2020, Roy et al., 2021]. The basic idea is to augment the main objective with a weighted sum of the constraints. The weighting coefficients — the Lagrange multipliers — are determined through gradient steps. The obtained solution would maximize the main objective and satisfy the set of constraints at the same time.

Recently, C-RL methods have been investigated in several scenarios, especially in *safe* RL. Research used these methods to leverage safe RL problems such as learning via human intervention [Saunders et al., 2017], continuous control [Achiam et al., 2017b, Dalal et al., 2018], advertising systems Wang et al. [2016b] and video games [Roy et al., 2021]. These works train artificial agents to acquire specific behaviors by avoiding some unwanted ones and encouraging others. However, since all these methods need to mathematically specify constraints, it can become challenging in complex environments. In fact, some behaviors can be *recognizable* (watch a behavior and recognize it), but not necessarily *specified* (extract specific features of a particular behavior).

Imitation Learning

A natural way of specifying a particular behavior is by demonstrating it. An expert agent provides a set of trajectories typically through tele-operation. The learning agent then attempts to imitate the expert decisions by learning an optimal policy that follows the given trajectories. Consequently, IL is a straightforward way to

[1] https://openai.com/blog/faulty-reward-functions/

specify desired and undesired behaviors for the agent to reproduce. We consider three main methods for performing IL: *behavioral cloning*, *direct-policy learning* and *inverse RL*.

Behavioral Cloning represents the simplest form of imitation learning. It is based on supervised learning, and attempts to exactly replicate the demonstrated behavior [Pomerleau, 1998, Wu et al., 2019]. Agents typically learn a parameterized policy function that generates the same actions when it is in the same states as given in the demonstrating trajectories. Thanks to function approximators, such agents could eventually generalize to other states if they resemble the ones that were given in the demonstrations. Such a method can be effective in certain applications. However, they are not suitable for long-term planning due to the distributional shift between training and testing states [Codevilla et al., 2019, Kirk et al., 2021].

Direct-policy learning represents an interactive version of behavioral cloning [Ross et al., 2011]. It assumes that the learning agent has access to an interactive demonstrator at training time. Beyond leveraging expert trajectories, this demonstrator can evaluate the agent's rollout trajectories, which increases the amount of data at training time. Although these methods improve upon behavioral cloning, getting them to transfer and generalize to unseen scenarios in complex environments remains challenging.

Inverse RL is different from behavioral cloning and direct-policy learning in that it typically does not use supervised learning. These methods rely on the idea of inferring the reward function of the environment based on the experts' demonstrations. They later use RL methods to learn optimal policies that maximize the inferred rewarding signal. These methods are shown to be very efficient when training and to handle long-term planning [Abbeel and Ng, 2004, Syed et al., 2008, Syed and Schapire, 2010, Ho and Ermon, 2016, Finn et al., 2016]. However, they can be difficult to train since agents need to learn the reward function jointly with the policy, leading to potential non-stationarities.

IL enables experts to explicitly specify a desired behavior. Nevertheless, providing demonstrations is not always possible, depending on the task at hand and on the environment. Besides, getting the agents to generalize beyond the given demonstrations remains an open-challenge.

Learning from Human Preferences

An alternative way for an expert to specify desired behavior during training without leveraging demonstrations is to provide some preferences. Actually, LfHP has been investigated in the context of diverse skill acquisition in continuous control setups [Wilson et al., 2012, Christiano et al., 2017]. The basic idea is to allow an expert — typically a human expert — to evaluate the learning agent's trajectories by specifying binary signals indicating if a behavior is desirable or not. Using these signals, the agent needs to fit a reward function to the human preferences. It then trains its policy using RL techniques to maximize this inferred reward signal. During training, the agent can maintain an estimator of its uncertainty of the reward function [Christiano et al., 2017]. If it is not sure, it can potentially query the expert for new evaluations.

These methods have the advantage of not requiring expert trajectories. Besides, additional techniques could be used to make sure that the expert intervention is minimal [Christiano et al., 2017]. Yet, training models using these methods is challenging. In fact, perfectly fitting a reward function to the human preferences could be tricky, especially if there are many constraints that the expert needs to specify.

Summary

When training artificial agents to accomplish a specific task, it might be possible that specific behaviours are either desired or undesired to be performed meanwhile. Such constraints could be incorporated explicitly within the reward function used to train such agents with RL methods. Yet, exhaustively defining such a function could be challenging in complex environments and is prone to reward hacking. Several methods could be used for behavioural specification when training agents to perform a task. These methods could be decomposed in two categories: methods that do not require an expert in the training loop, and methods that require one. For the former, Constrained RL represents a framework that extends standard RL to the case where additional constraints need to be satisfied while optimizing the main objective — typically the sum of returns. For the latter, imitation learning and learning from human experts could be used to incorporate desired behaviours in the learning process of artificial agents.

Chapter 8

Towards More Teachable Agents

In Chapter 2, we introduced *teachable autotelic agents* and discussed the design of such agents. In Table 2.1, we enlisted properties that characterize teaching in humans, we considered several approaches in AI that leverage some of these properties and discussed the missing ones. In this chapter, we leverage Table 2.1 to establish a roadmap towards teachable autotelic agents. We first describe steps that can be performed in the short term by just combining the properties of existing systems. Then we outline remaining and more fundamental challenges, and turn towards research on the longer term.

Short Term Steps towards Teachable Autotelic Agents

The IMAGINE [Colas et al., 2020b], DECSTR [Chapter 3] and HME + GANGSTR [Chapter 6] agents all display some of the properties one may expect from teachable autotelic agents. They are even complementary with respect to several of these properties. For instance, IMAGINE and DECSTR are endowed with a basic capability to interpret language, which is not the case of HME + GANGSTR. But the latter implements reciprocal modelling of the tutor and the learner, by contrast with the former methods. Thus a good deal of the limitations of the above agents could be overcome by combining their capabilities.

Integrating all these features into a single teachable autonomous agent would significantly increase their flexibility, making it possible to leverage a more significant part of the vast repertoire of interaction protocols or "*pragmatic frames*" used in human tutoring [Vollmer et al., 2016] and, more generally, opening the possibility to more natural, non template-based interaction [Zhou and Small, 2020]. It would also open the way towards new research questions, such as modelling feedback about goal selection rather than just about action, or the need for an arbitration mechanism between intrinsic motivations and social feedback. Besides, such an integration may require some developmental dynamics, resulting in displaying *overlapping waves* of sensorimotor and linguistic development [Siegler, 1998].

In terms of comparison between potential methods, research on teachable autotelic agents is still not mature enough to benefit from well established dedicated benchmarks. But the existing frameworks can rely on any simulated environment whose dynamics of interaction with the agent is rich enough, such as the Fetch Manipulate environment we use in DECSTR and HME +GANGSTR, or BABYAI [Chevalier-Boisvert et al., 2018] and many others.

Longer Term Steps towards Teachable Autotelic Agents: Integrating Social Learning Capabilities

Table 2.1 shows that the combination described above would still lack most of the properties of inferential social agents. Thus an important direction for future research will consist in combining the properties of the existing teachable autotelic agents and endow them with further social inference capabilities.

However, this latter combination effort is less straightforward, as reinforcement learning and autotelic learning from one side and inferential social learning from the other side rely on widely different assumptions. Actually, in developmental psychology, there is a hot debate between two extreme views, the empiricist view according to which the infant brain is a *tabula rasa* which is progressively filled with knowledge out of experience only, and the nativist view for whom the infant brain comes with a vast repertoire of hard-wired skills that just get available as the infant develops. Most research in reinforcement learning and autotelic learning rather adopt the empiricist view, but though they cannot be coined as truly nativists, the proponents of the Inferential Social Learning framework [Gweon, 2021] assume at least that infants are predisposed to infer other's intentions. Building agents that account for such predispositions in the general case raises difficult issues, as the Bayesian inference processes proposed in [Vélez and Gweon, 2021] to account for inference capabilities must be configured with adequate priors that are generally given by the model designer. In other domains like intuitive physics or causality, researchers like Tenenbaum [Xu et al., 2021] start proposing frameworks to account for such prewired inference capabilities, but integrating these approaches with reinforcement and autotelic learning still remains a challenge.

Longer Term Steps towards Teachable Autotelic Agents: Integrating Language Proficiency

Looking more closely at Table 2.1, one can see that an agent combining the properties of IMAGINE, DECSTR, HME + GANGSTR and inferential social learning agents would still lack one property, which is language proficiency. In language-augmented agents such as IMAGINE and DECSTR, the template-based language learning mechanism does not mimic the language acquisition processes of infants. However, the developmental processes and constraints involved in language acquisition may play a crucial role in the more general acquisition of interaction capabilities. Thus, more realistic models of language acquisition and learning of semantic predicates in infants [Goldstein et al., 2003, Kuhl, 2004, Mandler, 2012] may be required to better account for these capabilities. Besides, communication itself could be extended to richer natural language interactions [Lynch and Sermanet, 2020] so as to increase the teachability of these agents. The very fast progress in language learning with large transformer models [Floridi and Chiriatti, 2020, Rae et al., 2021] creates an opportunity in this direction. This emerging topic was recently covered in another position paper, see Colas et al. [2022a].

Finally, in contrast with the above questions which should be answered soon given the current pace of learning agents research, there are a few questions which remain largely unaddressed. For instance, how can an autonomous agent learn to determine the positive or negative valence of sophisticated feedback signals such as attitudes or linguistic nuances? Or how can we endow agents with the capability to generalize immediately what they have learned from an intended demonstration to other contexts? More fundamentally, how can we extend the *language grounding* capabilities of current teachable autotelic agents to solve the harder *symbol grounding* problem [Harnad, 1990]? In short, language in DECSTR and IMAGINE is more indexical than symbolic because the language tokens do not form a system [Nieder, 2009]. The acquisition of symbolic behavior is an emerging topic [Santoro et al., 2021] which can be of fundamental importance for future agents as, from one side, considering a social partner is necessary to establish conventional meaning and, from the other side, such agents may need the flexibility of symbolic behavior to appropriately learn from natural tutors.

Summary

In this chapter, we discussed the steps that can be performed to endow autotelic agents with teachability properties. In the immediate future, the existing teachable autotelic agents — which exhibit some of these properties but not others — could be combined and integrated with inferential social learning capabilities and more natural language learning capabilities. Once this is done, given the fast progress currently observed in the design of autotelic learning agents, we expect to soon see good enough teachable autonomous agents to use them for quantitative analyses in developmental psychology studies and for a better design of education programs. We also believe that this starting point is a key move towards better insertion of AI agents in the society, with improved capabilities to communicate with and to adapt to their human users, which is one of the central concerns of AI research.

Chapter 9

Spatial Predicate-based Representations

In Chapter 3, we introduced the concept of *semantic configurations* as part of an abstract representation space based on binary spatial predicates. We argued that such concepts — the relational spatial concepts — are rapidly leveraged by human toddlers long before language acquisition. These insights were based on the *Mandlerian view* of prelinguistic concept acquisition, which is based on the *Perceptual Meaning Analysis* (PMA) mechanism that translates temporal information perceived by infants into an iconic static form. We then introduced *predicate-based autotelic agents*, which represent their goal space based on semantic configurations.

In this chapter, we discuss the process of learning such abstract representations rather than pre-defining them. The main constraint is that these representations need to align with semantic concepts shared between humans, including spatial relations such as proximity and stacking.

Goal Space Representations

In the context of goal-conditioned reinforcement learning (GC-RL), learning goal-conditioned policies requires the existence of a goal space from which goal representation would be drawn. There exist two different scenarios. On the one hand, if the goal space is already pre-defined, autotelic agents could rely on the associated goal achievement function and reward themselves over completion of the goals (see Section 1.3 of Chapter 1 for a formal definition of goals). The process of learning in such agents requires generating the goals and training on them. Some approaches use a *replay-based goal generation* process, which exclusively rehearses the goals that were encountered by the agents (See Chapters 3 and 4). Other approaches consider an *external goal generation* process, where the goal sampler is external and independent of the learning agents' experience [Schaul et al., 2015, Andrychowicz et al., 2017, Lanier et al., 2019, Li et al., 2019]. Finally, some approaches attempt to learn the support of valid goals based on the encountered ones, and perform *automatic goal generation* based on trained generative models [Florensa et al., 2018].

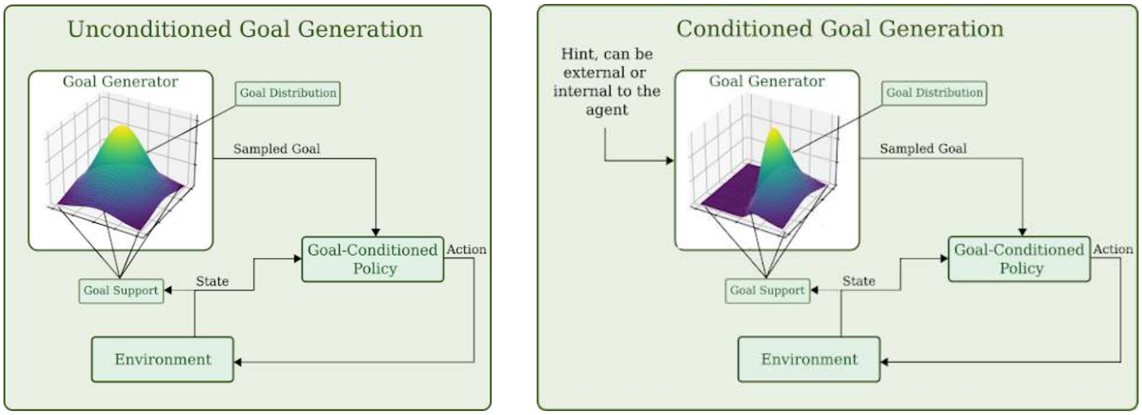
On the other hand, if the goal space is not already predefined, it should be learned during training together with its underlying reward function. Such approaches could make use of an external caregiver that provides descriptions of the achieved states enabling agents to learn embeddings of such descriptions [Colas et al., 2020b], or rely on information theoretic methods to maximize the entropy of the discovered states or to maximize the control [Eysenbach et al., 2018, Zhao et al., 2021]. However, the learned representations — which exclusively rely on the agent's sensors to categorize the skills — usually suffer from poor semantic distinguishement. For instance, the DIYAN [Eysenbach et al., 2018] and the MUSIC [Zhao et al., 2021] algorithms consider picking up an object and placing it at different positions as distinguishable skills.

Some concepts, such as proximity, cannot be defined objectively based only on sensorimotor information. Unless these concepts are used as building blocks for other ones (for example, two blocks are considered to be close if it is possible to stack a third block above them an construct a pyramid), it is challenging to learn them based only on the state space when the latter is continuous: what would be the distance threshold between two objects to consider them to be close to each other ? Since humans understand these concepts, it would be interesting to actually learn them by interacting with a human. This sort of interaction aims at aligning concepts as understood by humans with the ones learned by an embodied machine. In such scenarios, even though the machine has some arbitrary latent code for a particular goal, it should be able to associate this latent code to a particular semantic representation that is interpretable by a human.

Conditioned Goal Generation

A typical method for automatically generating valid goals consists in learning the support of the valid goal distribution using the distribution of goals encountered during training. When generating a goal using the learned generative model, this goal would be valid. Imagine if an agent encounters a diversity of goals, such as configurations with stacks, pyramids, and combinations of constructions. When automatically generating a goal, it is not straightforward how to focus on a particular feature — for example sampling pyramids. In fact, the goal generation process in this case does not focus on a specific region of the valid goal distributions. It would be interesting to leverage the goal generation process so that it could be *conditioned* on some hint. This hint could represent a specific feature of the desired goals — for example configuration with pyramids. The generative model would then use this hint to focus on a particular region of the learned goal space support. This is illustrated in Figure 9.1

Hints could take the form of descriptions based on natural language. This is the case of the *Language Goal Generator* that we introduced in Chapter 3. It takes as input a particular instruction, such as "*Construct a pyramid*", and outputs a set of valid goals where objects are arranged to form a pyramid. In general, hints are not



(a) (b)

Figure 9.1: Illustration of the goal generation module in the case where it is (a) not conditioned on anything and (b) conditioned on some external hints.

limited to language, but can be any sort of shared symbol such as binary predicates, gestures and gaze.

Guessing Games

Guessing games (GGs) were first introduced as part of *the Talking Heads Experiment* established by Steels [Steels, 2015]. It consists of a role play game involving two agents endowed with sensors enabling them to perceive the physical world. They do not necessarily need to be embodied — they do not need to perform actions — but GGs could easily be extended to the case of embodied agents. The roles involved in GGs are a *speaker* and a *hearer*. Both agents exchange these roles. Besides, a human experimenter can, at any time, pick these roles and replace an artificial agent.

Steels describes the rules of GGs as follows. Both the speaker and the hearer perceive a white board on which several shapes with different colors are drawn. The speaker focuses on one particular region of the board — the context — and draws the hearer's attention to that region. Then, the speaker chooses one specific object in that region — the topic. It provides a *verbal hint* to the hearer. This verbal hint consists of an expression that identifies the topic from all the other objects. It could be for example related to its colors, the number of its edges or its shape. Based on this verbal hint, the hearer tries to guess the topic chosen by the speaker and communicates its choice by pointing to that object. The game succeeds — both agents win — if the topic is identified by the hearer. Hence, this game is not adversarial but rather *cooperative*. If the hearer fails, the speaker chains with another hint, weakening its hypothesis that the hearer understood its first hint. This gradual communication enables both the hearer and the speaker to eventually align their understanding of concepts and the language used to describe them.

The idea of GGs could be extended to learning goal space representations that are semantically distinguishable and align with relational concepts such as proximity and stacking.

Learning Abstract Goal Representation through Guessing Games

To learn abstract goal representations, a human expert could play GGs with an artificial agent. Typically, the human expert would play the role of the speaker since we suppose that it is the human's concepts that we want to incorporate into the artificial agent. The topic in this case could correspond to either a particular state — in this case, it is static — or to a particular trajectory — in this case, it is dynamic. The human speaker gives a hint about this topic, and the artificial agent attempts to guess it. Progressively performing this scheme would eventually enable the artificial agent to guess the topics based on the hints. It could later take ownership of these hints and use them to train its conditioned-goal generator.

The fact that the artificial agent is endowed with a separate goal generator provides the possibility to decouple *skill learning* from *concept grounding* (See Chapter 3). In this case, the training of the agent could be decomposed into two phases:

- First, the agent autonomously interacts with the environment, discovers skills, learn to represent them and to achieve them in an unsupervised fashion. It jointly trains its goal generator to generate goals within the support of valid goals.

- Second, a human expert engages in a *double-sided* interaction with the agent. From the one side, GGs are triggered, where the expert plays the speaker and the agent plays the hearer. From the other side, the agent could query the expert about the concepts about which it is the most uncertain. This could be for example done by training an ensemble of classifiers, computing scores based on disagreement and selecting queries corresponding to concepts with the highest score.

The interactions with the human expert during the second phase, either through GGs or through selecting queries, enable the artificial agent to take ownership of the human's hints. Besides, it progressively updates its beliefs about the meanings of these concepts by conditioning its goal generator — the one trained in phase one — and updating it.

Summary

In this chapter, we discussed some insights about learning abstract goal representations that align with semantic concepts such as spatial relations. In these scenarios, learning representations in an unsupervised fashion is unlikely to provide a diversity of semantically distinguishable skills. This is mainly due to the continuity of the state space — the space which is usually used to learn goal representation in unsupervised RL. We argued that interactions with human experts in the form of *guessing games* could enable artificial agents to progressively update their autonomously learned representations to match the ones corresponding to the semantic concepts known by humans. We discussed the decoupling of skill learning and concept acquisition and we provided some ideas about training artificial agents to efficiently categorize their learned skills.

Chapter 10

Guiding Teachable Autotelic Agents

In Chapter 6, we endowed our autotelic agents with two new ingredients: *an internalization mechanism* that enables them to remember and rehearse goals that they did not physically discover yet; *an internal goal source selector* that enables them to autonomously select to query an expert social partner (SP) for help. Both these mechanisms translate two of the *teachability properties* outlines in Chapter 2: taking ownership of external social signals and turning them into *intra-personal psychological tools* [Vygotsky, 1994] through internalization; fulfilling *social communication-based transparency* by actively requesting the caregiver's guidance. We also introduced a social interaction protocol (HME) which enables an expert SP to progressively take the learning agent beyond their *zone of proximal development* (ZPD).

In this chapter, we discuss these mechanisms and attempt to dig a little deeper in the design of efficient social interaction protocols.

Role of Social Guidance

A social interaction protocol typically involves a learning agent and an expert caregiver. Ideally, a good interaction protocol needs to be efficient for both parties. On the one hand, the caregiver needs to be solicited as little as possible. The main reason behind this constraint is that the intervention of an external social partner usually comes at a cost. First, if the caregiver is a hard-coded program, modeling expert knowledge — on which the caregiver relies to provide adequate teaching signals — usually requires hand-engineered specific functions which can rapidly become complex depending on the environment. Second, since such interventions depend on the learning agent's current status, the caregiver needs to evaluate the agent's current progress, which could consume a significant amount of the training budget. Third, multiplying social interventions could affect the agent's autonomy if it becomes more reliable on the social partner. This could translate in some sort of heavy overfitting on the SP's signals, and the agent would generalize less to new situations when it is left alone. In Chapter 6, we managed to reduce the amount of social intervention down to 0.5% of the training budget. We have shown that such an amount is sufficient to discover and start learning the most complex goals in a five object manipulation benchmark. However, this amount corresponds to 1200 episodes, which is still significant .

On the other hand, the learning agent needs to continue using its own intrinsically motivated exploration rather than exclusively relying on the caregiver to discover new goals. We don't want to have a coupling of *individual episodes* and *social episodes* where the former specializes in consolidating previously known skills and the latter in discovering new ones. It would be interesting if the learning agents learned a model of their environment while interacting with the SP. In principle, this would enable them to imagine new goals, pursue them and learn about them autonomously. This is the case of the IMAGINE agent [Colas et al., 2020b], where agents imagine new goals based on the compositionality of language.

In Chapter 6, although it is the learning agent that selects to query the SP for helps, it has no control whatsoever on which goals are proposed. The latter are *dictated* by the SP, and the learning agent needs to pursue these particular goals during social episodes. This can be problematic if there is no synchronization between the proposed goals and the agent's current learning progress. It would be interesting if the learning agent could specify what region of the goal space it wants to query the SP about, or if it could have a choice to neglect the SP's propositions if it estimates that pursuing a goal proposed by the SP is irrelevant to its learning process. This is related to the *pragmatic aspect* of learning in children (see Chapter 2). Actually, we want our teachable agent to be able to efficiently infer the intent of the SP, compare it to their own purposes and choose or not to take it into consideration. Pragmatic learning was recently investigated in the context of multi-goal learning in AI, where pragmatic agents are able to infer the goal of the SP from a given demonstration [Caselles-Dupré et al., 2022].

Theory of Mind

In Chapter 6, social interventions correspond to goal proposals: the SP suggests goals that the learning agent pursues. This process was simplified by hypothesizing that the SP's goals are represented in the same space — the semantic configuration space — as the learning agent. More generally, during a social interaction, understanding the goal from the behavior or the instruction of a tutor is crucial. This clearly calls upon a mental model of the other party's expectation, which drives from a more general model known as a *theory of mind* [Dennett, 1987, Gopnik and Meltzoff, 1997]. The field of AI was interested in incorporating such models in the learning agents, resulting in the family of *inferential agents* [Gweon, 2021, Caselles-Dupré et al., 2022]. Within a social interaction protocol, both the caregiver and the SP need to construct a model for one another. On the one hand, the SP needs to be able to evaluate the agent's current exploration limits in order to efficiently guide it. In Chapter 6, this was simply done by considering the goals that the agent has reached at least once. However, it does not necessarily translate the agent's exploration limits: achieving a goal once could happen by pure luck, considering that the agent knows how to get back to that goal can be irrelevant. Consequently, the SP's model of the learning agent needs to take into account both its discovered goals and its current policy. On the other hand, the learning agent needs to construct a model of the SP in order to infer their intentions as discussed in the previous paragraph. Besides, such a model could also be beneficial when it enables to learning agent to simulate its SP's intervention even if the latter is absent. We further discuss these insights in the next paragraph.

Internalizing Social Signals

Internalizing social signals implies taking ownership of them. In other words, the learning agents construct an internal model using external signals that go beyond their embodied experience. We argued in Chapter 2 that this would regulate the learning agent's motivations, and showed in Chapter 6 that it enables autotelic agents to autonomously sample goals that they did not physically encounter. The internalization mechanism we proposed in Chapter 6 consists in storing the goals proposed by the SP in the same buffer as the physically discovered ones. This could be more or less efficient depending on the environment. Actually, as the number of goals increases, the internalized ones would rapidly get dissolved into the physically discovered ones. It would be interesting to build an attention mechanism that specifically considers the SP's goals.

An efficient internalization mechanism should go beyond the simple *remember* and *rehearse* scheme. It should encompass the whole social intervention process: internalizing the SP's intentions in order to be able to simulate the SP's interventions in novel situations. To do so, the learning agents need to build upon the theory of mind model of its SP. Based on the structure of the goal space, and on the frequency and nature of social goal proposals, the learning agent could predict what goal the SP would propose in certain situations.

Combining multiple forms of Social Guidance

In Chapter 5, we introduced *Hybrid Goal Exploration Processes* (HGEPs) as a family of algorithms for learning multiple goals from internal and external sources. We highlighted that an external program — such a social partner — proposes goals in any form and that these goals are fed to an *interpretation module* that translates them into the learning agent's representation space. The HME interaction protocol proposed in the study of Chapter 6 simplifies this idea by supposing that the SP's goals and the agent's goals are the same (the interpretation module corresponds to the identity function). However, in general, the learning agent needs to train a *multi-headed* interpretation module. Ideally, it would take as input any sort of social intervention — demonstrations, instructions, sets of constraints, preferences — and generate a set of compatible goals in the same representation space as the agent.

This idea is similar to the *Language Goal Generator* we introduced in Chapter 3, which is a generative model that takes as input an instruction in natural language and outputs a set of goals that respect the given instruction. An extension of such a generator to a multi-headed model that handles multiple forms of external goals could be interesting.

Summary

This chapter discusses the limits of the HME social interaction protocol proposed in Chapter 6 to guide teachable autotelic agents. We discussed the role of social guidance during the learning of autotelic agents, and argued that it should not be restricted to exploration. Actually, autotelic agents should be able to continue to explore alone even if the SP is absent. The social guidance should enable the learning agent to gain insights about the intent and purposes of their SP. This would allow them to build a model of their caregiver and efficiently internalize their social signals. By doing so, teachable autotelic agents should be able to imagine new goals that they did not encounter before by using the structure encapsulated within the social interventions. We also discussed the combination of multiple forms of social guidance, which provides more flexibility to the social partner when trying to guide the learning agent.

Conclusion

> *"Don't adventures ever have an end? I suppose not. Someone else always has to carry on the story." Bilbo Baggins, The Lord of the Rings.*

The central purpose of the present research is to design artificial agents capable of learning from external *social signals* while keeping *their own motivations and purposes*. The main questions around which we wrapped our investigations were the following:

1. What would be the most efficient fashion through which artificial agents represent their goals in order to communicate easily with social partners ?

2. What sort of biases could be induced within these agents' internal models to enable efficient transfer and generalization between different skills ?

3. Can we formalize learning multiple goals under the assistance of expert social partners ?

4. How can we design light social interaction protocols that would enable artificial agents to maximize their control over their goal space ?

Our work is built upon recent research on *autotelic agents*: artificial agents which are intrinsically motivated to represent, generate, pursue and learn about their own goals. We introduced a novel family of autotelic agents that can benefit from social signals, which we called *teachable autotelic agents*. Such agents need to leverage several *teachability properties* inspired from findings in developmental psychology and education sciences on the normal teaching of children.

Our agents are *predicate-based*: they represent their goals as sets of constraints based on relational spatial predicates. We showed that these semantic representations allow the decoupling of sensorimotor learning and language grounding, increase the behavioral diversity and facilitate the communication with social partners that use natural language.

Our agents are *graph-based*: they model both their policy and their state-action value function using *Graph Neural Networks* (GNNs). We showed that combining GNNs with predicate-based goal representations allow light computational schemes to be efficient in transferring to held-out goals including combinations of previously learned skills and skills of increasing complexity.

Our agents are *socially situated*: they are endowed with *internalization mechanisms* that enable them to remember social signals even in the absence of the social partner. We showed that this had a beneficial effect of regulating our agents' intrinsic motivations as they can autonomously rehearse social signals when they are left alone. Besides, our agents are endowed with an *internal goal source selector* that enables them to deliberately choose to either self-generate their own goals or ask their social partner for help. The query selection is based on the agents' own estimation of their competence: whenever they estimate that they are not progressing enough on the set of skills they already know, they query the social partner for unlocking new ones.

Beyond the design of teachable autotelic agents themselves, we introduced a novel *social interaction protocol*, HME, for teaching these agents under the supervision of an expert social partner. The latter continuously update their model of the learning agent's current exploration limits. They use this model to first propose a goal that the agent has already visited at least once and that is a stepping stone to new goals — *a frontier goal* — then continue with a adjacent new goal — *a beyond goal.* We showed that, by doing so, the social partner smoothly takes the learning agent beyond its *zone of proximal development* (ZPD), which represents the area between what the agent can achieve alone and what it can achieve with the help of an expert partner. Little by little, the learning agent becomes less dependent on its social partner, progressively growing its ZPD until it manages to learn about its entire goal space. We showed that a social intervention valued at 0.5% of the training budget was sufficient for the agent to unlock the most complex goals within a five object manipulation benchmark.

We believe that *teaching* embodied artificial agents with an expert human-in-the-loop is promising, especially if the learning agents can profit as much as possible from external social signals. They would eventually internalize them and use them to update their own internal models, which might include some long-term planning modules. We believe that combining these insights with the existing and inspiring research topics such as self-imagination or inferential social learning is an exciting idea. Resulting agents would be teachable, creative and pragmatic, which are useful properties in the quest of the autonomous open-ended acquisition of skills.

Part IV

Appendices

Appendix A

Appendix of Delayed Geometric Discounts

This appendix presents all additional methods, proofs, results, discussions as well as implemented details for the Delayed Geometric Discounts study of Chapter 1.

A.1 Non-Stationary Case

In the following, we consider yet a more diverse problem formulation by using a linear combination of the delayed losses. Let $\mathcal{L}_\eta$ be the following objective function defined as:

$$\mathcal{L}_\eta(\pi, r) := \mathbb{E}_{\pi,p_0}\Big[\sum_t \eta(t) r_t\Big] \quad \textit{such that} \quad \eta(t) = \sum_{d=0}^{D} w_d \Phi_d(t) \tag{A.1}$$

where the depth $D \in \mathbb{N}$ and the coefficients $w_d \in \mathbb{R}$ for any $d \in [0, D]$ are known.

In general, the optimal control in the sense of $\mathcal{L}_\eta$ is not stationary. Here, we propose to approximate the optimal control with a non-stationary policy over a finite horizon followed by a stationary one.

Approximate optimal control

In the general case, the optimal control is not necessarily stationary. Consider the problem of learning the optimal action profile $a^*_{0:\infty}$ which yields the following optimal value function:

$$V^*_\eta(s) := \max_{a_{0:\infty}} \mathbb{E}_{a_{0:\infty}}\Big[\sum_{t=0}^{\infty} \eta(t) r_t | s_0 = s\Big], \tag{A.2}$$

where $\mathbb{E}_{a_{0:\infty}}$ is the expectation over trajectories generated using the designated action profile. In this section we propose to solve this problem by approximating the value function from Equation A.2. This is achieved by applying the Bellman optimality principle in order to decompose V^*_η into a finite horizon control problem (optimising $a_{0:H}$ with $H \in \mathbb{N}$) and the optimal value function $V^*_{f^{H+1}(\eta)}$ where f is an operator transforming the weighting distribution as follows:

$$\Big[f(\eta)\Big](t) := \sum_{d=0}^{D} \big(\gamma_d \sum_{j=d}^{D} w_j\big)\Phi_d(t) = \langle \Gamma \cdot \mathbf{w}, \mathbf{\Phi}(t)\rangle$$

$$\textit{where} \quad \Gamma := \begin{bmatrix} \gamma_0 & \gamma_0 & \gamma_0 & \cdots & \gamma_0 \\ 0 & \gamma_1 & \gamma_1 & \cdots & \gamma_1 \\ 0 & 0 & \gamma_2 & \cdots & \gamma_2 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \gamma_D \end{bmatrix} \quad ; \quad \mathbf{w} := \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} \quad ; \quad \mathbf{\Phi}(t) := \begin{bmatrix} \Phi_0(t) \\ \Phi_1(t) \\ \Phi_2(t) \\ \vdots \\ \Phi_D(t) \end{bmatrix}$$

For the sake of simplicity, let $\langle \mathbf{1}, \eta\rangle := \sum_{d=0}^{D} w_d$ and $\big[f^n(\eta)\big](t) := \langle \Gamma^n \cdot \mathbf{w}, \mathbf{\Phi}(t)\rangle$ for any $n, t \in \mathbb{N}$.

Proposition 2 *For any state $s_0 \in \mathcal{S}$, the following identity holds:*

$$V^*_\eta(s_0) = \max_{a_{0:H}} \Big\{ \mathbb{E}_{a_{0:H}}\Big[\sum_{t=0}^{H} \langle \mathbf{1}, f^t(\eta)\rangle r_t\Big] + \mathbb{E}_{a_0,a_1,..,a_H}\big[V^*_{f^{H+1}(\eta)}(s_{H+1})\big]\Big\} \tag{A.3}$$

As a consequence, the optimal policy in the sense of $\mathcal{L}_\eta$ is to execute the minimising arguments $a^*_{0:H}$ of Equation A.3 and to execute the optimal policy in the sense $\mathcal{L}_{f^{H+1}(\eta)}$. Unfortunately, solving $\mathcal{L}_{f^{H+1}(\eta)}$ is not easier than the original problem.

However, under mild assumptions, for H large enough, this criterion can be approximated with a simpler one. In order to derive this approximation, recall that the power iteration algorithm described by the recurrence $\mathbf{v}_{k+1} = \frac{\Gamma \cdot \mathbf{v}_k}{\|\Gamma \cdot \mathbf{v}_k\|}$ converges to the unit eigenvector corresponding to the largest eigenvalue of the matrix Γ whenever that it is diagonalizable. In particular, if $0 < \gamma_D < ... < \gamma_0 < 1$ and $\mathbf{v}_0 = \mathbf{w}$, then Γ is diagonalizable, γ_0 is it's largest eigenvalue and the following holds:

$$\lim_{n\to\infty} \mathbf{v}_{n+1} = \lim_{n\to\infty} \frac{\Gamma^n \cdot \mathbf{w}}{\prod_{k\leq n} \|\Gamma \cdot \mathbf{v}_k\|} = [\mathbb{1}_{i=0}]_{i\in[0,D]} \quad \textit{and} \quad \lim_{n\to\infty} \frac{V^*_{f^n(\eta)}(s)}{\prod_{k\leq n} \|\Gamma \cdot \mathbf{v}_k\|} = V^*(s) \tag{A.4}$$

Under these premises, the right hand term of the minimisation problem in Equation A.3 can be approximated with the optimal value function in the sense of the classical RL criterion $\mathcal{L}$ (which optimal policy can be computed using any standard reinforcement learning algorithm in the literature).

Formally, we propose to approximate Equation A.3 with a proxy optimal value function $\tilde{V}^*_{\eta,H}(s_0)$:

$$\tilde{V}^*_{\eta,H}(s_0) = \max_{a_{0:H}} \Big\{ \mathbb{E}_{a_{0:H}}\Big[\sum_{t=0}^{H} \langle \mathbf{1}, f^t(\eta)\rangle r_t\Big] + \Big(\prod_{k\leq H} \|\Gamma \cdot \mathbf{v}_k\|\Big)\mathbb{E}_{a_0,a_1,..,a_H}\big[V^*(s_{H+1})\big]\Big\} \tag{A.5}$$

A direct consequence of Equation A.4 is that $\lim_{H\to\infty} \tilde{V}^*_{\eta,H}(s) = V^*_\eta(s)$ for any state $s \in \mathcal{S}$. In addition, for a given horizon H, the optimal decisions in the sense of the proxy problem formulation are to execute for the first H steps the minimising arguments $a^*_{0:H}$ of Equation A.5 (they can be computed using dynamic programming) and then execute the optimal policy in the sense of the γ_0-discounted RL (which can be computed using any standard RL algorithm).

Optimal Control Approximation

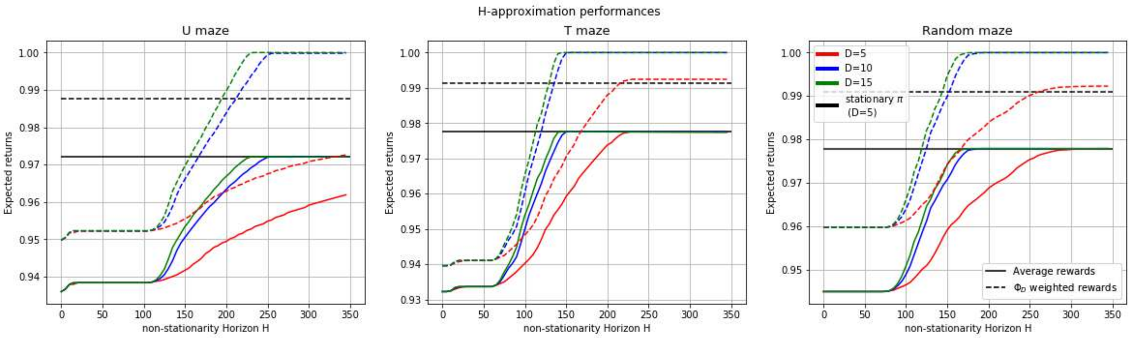


Figure A.1: Learned H-close optimal control

In this section we investigate the performances of the policies learned using Algorithm 6 as we vary the non stationarity horizon H for three depth parameters $D \in \{5, 10, 15\}$ (respectively the red, blue and green curves). We reported in Figure A.1 the expected returns as the average rewards using the continuous lines and as the Φ_D weighted rewards using dashed lines. As a baseline, we can observe in each figure the performances of the optimal policy in the sense of the geometrically discounted criterion when the non-stationarity horizon $H = 0$. We also reported the performances of the best stationary policy learned using GSAC for a depth parameter $D = 5$.

Algorithm 6 H-close optimal control

```
Compute $\pi^*, V^* \leftarrow$ POLICY ITERATION and initialize $\boldsymbol{v}_0 \leftarrow \boldsymbol{w}$
for $t \in [1, H]$ do
    Compute $\boldsymbol{v}_t \leftarrow \Gamma \cdot \boldsymbol{v}_{t-1}$
Initialise $\boldsymbol{V}_{H+1}(s) \leftarrow V^*(s) \cdot \prod_{k \leq H} \|\Gamma \cdot \mathbf{v}_k\|$
for $t \in [H, 0]$ do
    Solve $\pi_t(s) \leftarrow \arg\max_a \|\boldsymbol{v}_t\| \cdot c(s,a) + \mathbb{E}_{s' \sim \mathcal{P}(s,a)}\left[\boldsymbol{V}_{t+1}(s')\right]$
    Compute $\boldsymbol{V}_t(s) \leftarrow \|\boldsymbol{v}_t\| \cdot c(s, \pi_t(s)) + \mathbb{E}_{s' \sim \mathcal{P}(s,\pi_t(s))}\left[\boldsymbol{V}_{t+1}(s')\right]$
Return: $(\pi_t)_{t \in [0,H]}, \pi^*$
```

The first notable observation is that increasing the hyper-parameter H does indeed improve performances. In addition, we notice that the maximum possible improvement is reached using a finite horizon H (i.e. the maximising argument of both $\tilde{V}_{\eta,H}$ and V^*_η are the same). Intuitively, the H non stationary steps enable the agent to get to an intermediate state from which the optimal policy in the sense of the discounted RL formulation can lead to the state with the highest rewards. This explains the effectiveness of current hard exploration RL algorithmsEcoffet et al. [2019], Eysenbach et al. [2019]: by learning a policy that reaches interesting intermediate state, these methods are implicitly learning an approximate solution of $\tilde{V}_{\eta,H}$.

In the case of $D = 5$, the obtained performances using Algorithm 6 converged to the performances of the optimal stationary policy obtained using GSAC in both the random and T-maze. On the other hand, unlike GSAC, the H-close algorithm is not sensitive to initialisation: in fact, even for an arbitrarily high depth parameter ($D = 15$), the learned policy ends up saturating the average and the Φ_D weighted rewards. More interestingly, increasing the depth can be beneficial as we observe empirically that for higher depth parameters, the non-stationarity horizon required to achieve the best possible performances decreases.

A.2 Additional Results

A.2.1 Influence of the Depth Parameter in Discrete Mazes.

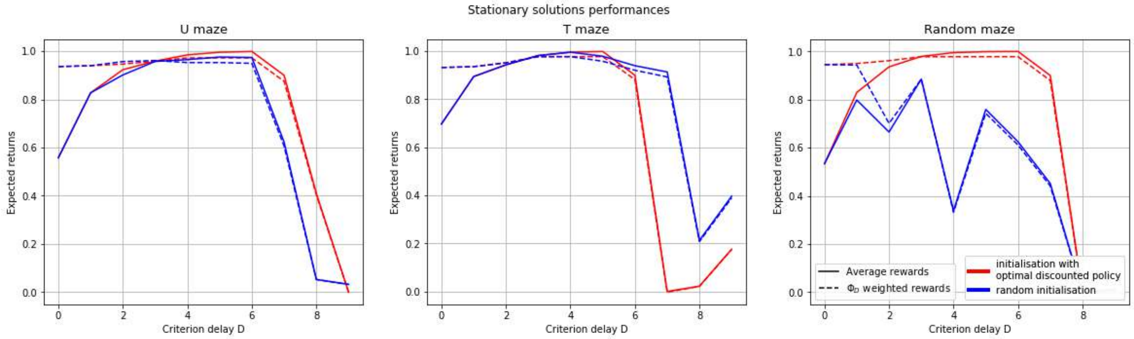


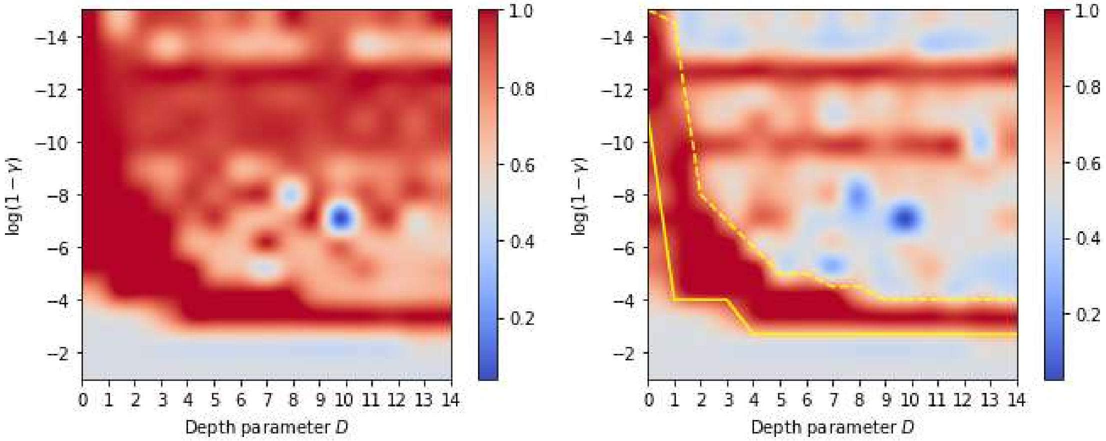
Figure A.2: Learned stationary policy (G-SAC) performances as the depth parameter varies

In Figure A.2, we reported the performances of GSAC as we varied the depth hyper-parameter D using two initialisation protocols. The blue curves are associated with randomly selected initial parameters while the red curves are associated with experiments where the policy is initialised with the solution of the geometrically discounted problem. The solid lines correspond to the average reward while the dashed lines correspond to the Φ_d weighted loss (as computed in $\mathcal{L}_d$).

In all experiments, the expectations were averaged across 25 runs of the algorithm using trajectories of length 4000 initialised in all possible states (uniform p_0) . A common observation is that for a depth D higher than 6 7 the algorithm was unstable and we couldn't learn a good stationary policy in a reliable way. However, the learned stationary policies with even a relatively shallow depth parameter yielded reliable policies that not only maximise the Φ_D weighted rewards but also improved the average returns. Notice how the baseline ($D = 0$, i.e. the geometrically discounted case) always under-performs when compared to the learned policies for a depth parameter around 3.

We also observe that the algorithm is sensitive to the used initialisation. Using the optimal policy in the sense of the geometrically discounted objective (red curves) helped stabilise the learning procedure in most cases: this is particularly true in the random maze environment where a random initialisation of the policy yielded bad performances even with a low depth parameter. This heuristic is not guaranteed to produce better performances in all cases, notice how for $D = 7$ in the T-maze, a random initialisation outperformed this heuristic consistently.

A.2.2 Ablation analysis (Discrete Corridor Environment)



(a) Max performance

(b) Average performance

Figure A.3: Success rate as a function of the Delay D and the discount values

In this section, we consider a simple 2000 states long corridor environment with a deceptive reward of 0.9 on one extremity, a desirable reward of 1 on the other and an adversarial reward of -1 in the middle (states 990 to 1010). For this environment, we consider that a policy is successful if it ends up reaching the best reward and that it fails in any other scenario (including the case where it reaches the deceptive reward). The success rate of a policy is consequently the proportion of states from which it reaches the best reward.

we investigate the performances of the obtained policies using GSAC[1] as we vary the delay parameter $D \in \{0, \ldots, 14\}$ and the discounts $\gamma_{i \leq D} = \gamma \in [1 - 10^{-1}, 1 -$

[1] A discrete version of the algorithm with down dynamics

10^{-14}]. For each couple of values, we evaluate the best (Figure A.3a) and the average (Figure A.3b) success rate of the learned policies in 10 randomly initialised runs of GSAC. The obtained performances are reported in Figure A.3 as heat-maps where higher success rates (close to 1) are associated with red and lower ones with blue.

Naturally, for low discount parameter γ, the success rate is around 0.5 as states on each side of the adversarial reward are encouraged to leave that area in the direction of the closest positive reward. Interestingly, there is a limiting curve (continuous yellow line in Figure A.3b) above which the best stationary policy in the sense of $\mathcal{L}_D$ has a success rate of 1, and a second line (dashed yellow line) above which numerical instabilities induce poor numerical performances.

Notice that if we consider the vertical line in Figure A.3 corresponding to $D = 0$, we recover the Blackwell criterion: there exists a critical value of the discount above of which agents are capable of reaching the desirable reward. Intuitively, these observations generalize this criterion to the delayed geometrically discounted framework: There exists a critical frontier that depends on both the delay D and the discounts $\gamma_{i\leq D}$, above of which optimal stationary policies in the sense of the delayed criterion $\mathcal{L}_D$ is also optimal in the sense of the average criterion

A.3 Implementation Details

This part includes details necessary to reproduce results in the continuous robotics environments. The code will be released with the camera-ready version of the paper.

Soft Actor Critic Details. Our algorithms are based on the *Soft Actor Critic* algorithm [Haarnoja et al., 2018]. Notably, we use the double Q-Networks trick to help tackle the overestimation bias [Fujimoto et al., 2018]. In our experiments, we do not automatically tune the entropy hyperparameter α. In fact, we found that fixing its value to $\alpha = 0.2$ is sufficient for the purpose of this paper.

Delayed Networks. As the results for the GSAC learned stationary policies show that the performance tends to decrease for high values of delay D, we opt for $D = 2$ in the continuous robotics experiments. Our main objective is to study the effect of shallow delayed geometric discounts on simulated robotics environments.

Networks Architecture. Each of the discount factors γ_1 and γ_2 is associated with a different critic and target critics networks. All these networks, as well as the policy, are 1-hidden layer networks of hidden size 256. They use *ReLU* activations and the Xavier initialization. We use Adam optimizers, with learning rates 3×10^{-4}. The list of hyperparameters is provided in Table A.1.

A.4 Proofs of the Technical Results

A.4.1 Useful intermediate results

We start by introducing some useful intermediate results that will be used later on to prove propositions 1 and 2.

Table A.1: Sensorimotor learning hyperparameters used in G-SAC.

Hyperparam.	Description	Values.
lr_actor	Actor learning rate	3×10^{-4}
lr_critic	Critic learning rate	3×10^{-4}
τ	Polyak coefficient for target critics smoothing	0.95
batch_size	Size of the batch during updates	256
hidden_size	Dimension of the networks' hidden layers	256
γ_0	Discount factor associated with the first delay	0.99
γ_1	Discount factor associated with the second delay	0.99
delayed_update_ratio	# of first critic updates before single second critic update	1
update_per_step	# of networks updates loop per a single environment step	1
target_update	Target networks soft updates per step	1
α	Entropy coefficient used in SAC	0.2
automatic_entropy	Automatically tune the entropy coefficient	*False*

Useful Results for Proposition 1

To derive the desired property of the value function, it is useful to derive a relationship between the coefficients $\Phi_D(t)$:

Lemma 1 *For any integer $D > 0$, the following equalities hold:*

$$\Phi_D(t) = \sum_{k=0}^{t} \gamma_D^k \Phi_{D-1}(t-k) = \Phi_{D-1}(t) + \gamma_D \Phi_D(t-1) = \sum_{d=0}^{D} \gamma_d \Phi_d(t-1)$$

Now, consider the state-value function $V_d^\pi(s)$ defined as:

$$V_D^\pi(s) := \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \Phi_D(t) r_t | s_0 = s\Big]$$

Lemma 1 can be used to derive a relationship between the value functions $(V_d^\pi)_{d=0}^D$ for any depth parameter $D \in \mathbb{N}$:

Proposition 3 *For any state $s \in \mathcal{S}$ and for any integer $D \in \mathbb{N}$, we have:*

$$V_D^\pi(s) = \mathbb{E}_{\substack{a\sim\pi(s)\\ s'\sim\mathcal{P}(s,a)}}\Big[c(s,a) + \sum_{d=0}^{D} \gamma_d V_d^\pi(s')\Big]$$

Useful results for Proposition 2

This proposition is proved using inductive reasoning. For this reason, we start by consider the simpler case of $H = 1$:

Proposition 4 *For any state $s_0 \in \mathcal{S}$, the following identity holds:*

$$V_\eta^*(s_0) = \max_{a_0}\Big\{\big[\sum_{d=0}^{D} w_d\big] c(s_0, a_0) + \mathbb{E}_{s_1}\big[V_{f(\eta)}^*(s_1)\big]\Big\} \qquad \text{(A.6)}$$

A.4.2 Proofs

In this section we provide the proofs of the technical results

Proof of Proposition 1

Recall that:

$$Q_D^\pi(s,a) = \mathbb{E}_{s' \sim \mathcal{P}(s,a)} \Big[V_D^\pi(s') \Big]$$

Then the statement from Proposition 3 can be reformulated as

$$Q_D^\pi(s,a) = \mathbb{E}_{\substack{s' \sim \mathcal{P}(s,a) \\ a' \sim \pi(s')}} \Big[c(s,a) + \sum_{d=0}^{D} \gamma_d Q_d^\pi(s',a') \Big] \tag{A.7}$$

Which means that Q_D^π is a fixed point of T_π^D. Given that this operator is a γ_D contraction with $\gamma_D \in (0,1)$, it follows that it is the unique fixed point.

Proof of Lemma 1

The proof relies on algebraic manipulations of the indices:

$$\begin{aligned}
\Phi_D(t) :=& \sum_{\substack{\{a_d \in \mathbb{N}\}_{i=0}^{D} \\ \textit{such that } \sum_d a_d = t}} \prod_{d=0}^{D} \gamma_d^{a_d} \\
&= \sum_{k=0}^{t} \gamma_D^k \Big[\sum_{\substack{\{a_d \in \mathbb{N}\}_{i=0}^{D-1} \\ \textit{such that } \sum_d a_d = t-k}} \prod_{d=0}^{D-1} \gamma_d^{a_d} \Big] = \sum_{k=0}^{t} \gamma_D^k \Phi_{D-1}(t-k)
\end{aligned}$$

This concludes the proof of the first equality. Similarly, the second equality is achieved through similar algebraic treatments:

$$\begin{aligned}
\Phi_D(t) &= \sum_{k=0}^{t} \gamma_D^k \Phi_{D-1}(t-k) \\
&= \Phi_{D-1}(t) + \sum_{k=1}^{t} \gamma_D^k \Phi_{D-1}(t-k) \\
&= \Phi_{D-1}(t) + \gamma_D \sum_{k=0}^{t-1} \gamma_D^k \Phi_{D-1}((t-1)-k) = \Phi_{D-1}(t) + \gamma_D \Phi_D(t-1)
\end{aligned}$$

This concludes the proof of the second equality. The last one can be deduced directly using induction. In fact, the induction is a direct consequence of the second equality, and the basis case is trivially verified as:

$$\Phi_0(t) = \gamma_0^t = \gamma_0 \Phi_0(t-1)$$

Proof of Proposition 3

The proof relies on some algebraic manipulation as well as the last equality from Lemma 1.

$$\begin{aligned}
V_D^\pi(s) &= \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty}\Phi_D(t)c(s_t,a_t)|s_0=s\Big]\\
&= \mathbb{E}_\pi\Big[c(s_0,a_0)+\sum_{t=1}^{\infty}\Phi_D(t)c(s_t,a_t)|s_0=s\Big]\\
&= \mathbb{E}_\pi\Big[c(s_0,a_0)+\sum_{d=0}^{D}\gamma_d\sum_{t=1}^{\infty}\Phi_d(t-1)c(s_t,a_t)|s_0=s\Big]\\
&= \mathbb{E}_\pi\Big[c(s_0,a_0)+\sum_{d=0}^{D}\gamma_d\sum_{t=0}^{\infty}\Phi_d(t)c(s_{t+1},a_{t+1})|s_0=s\Big]\\
&= \mathbb{E}_{\substack{a\sim\pi(s)\\ s'\sim\mathcal{P}(s,a)}}\Big[c(s,a)+\sum_{d=0}^{D}\gamma_d V_d^\pi(s')\Big]
\end{aligned}$$

where the last equality relies on the Markov property of MDPs.

Proof of Proposition 4

The proof relies on the linearity of the expectation as well as proposition 3. Let's denote in this proof with the policy π the sequence of action $a_0, a_1, \ldots, a_\infty$ and with the transposed policy $T\pi$ the sequence of actions $a_1, a_2, \ldots, a_\infty$. The following property then holds

$$\begin{aligned}
V_\eta^\pi(s) = \sum_{d=0}^{D} w_d V_d^\pi(s) &= \sum_{d=0}^{D} w_d \mathbb{E}_{\substack{a\sim\pi(s)\\ s'\sim\mathcal{P}(s,a)}}\Big[c(s,a)+\sum_{i=0}^{d}\gamma_i V_i^{T\pi}(s')\Big]\\
&= \mathbb{E}_{a\sim\pi(s)}\Big[\big(\sum_{d=0}^{D} w_d\big)c(s,a)\Big] + \mathbb{E}_{\substack{a\sim\pi(s)\\ s'\sim\mathcal{P}(s,a)}}\Big[\sum_{d=0}^{D} w_d\sum_{i=0}^{d}\gamma_i V_i^{T\pi}(s')\Big]\\
&= \mathbb{E}_{a\sim\pi(s)}\Big[\big(\sum_{d=0}^{D} w_d\big)c(s,a)\Big] + \mathbb{E}_{\substack{a\sim\pi(s)\\ s'\sim\mathcal{P}(s,a)}}\Big[\sum_{d=0}^{D}\gamma_d\big(\sum_{i=d}^{D} w_i\big)V_d^{T\pi}(s')\Big]\\
&= \mathbb{E}_{\substack{a\sim\pi(s)\\ s'\sim\mathcal{P}(s,a)}}\Big[\big(\sum_{d=0}^{D} w_d\big)c(s,a)+\sum_{d=0}^{D}\gamma_d\big(\sum_{i=d}^{D} w_i\big)V_d^{T\pi}(s')\Big]\\
&= \mathbb{E}_{\substack{a\sim\pi(s)\\ s'\sim\mathcal{P}(s,a)}}\Big[\big(\sum_{d=0}^{D} w_d\big)c(s,a)+V_{f(\eta)}^{T\pi}(s')\Big]
\end{aligned}$$

Using this equality and the Bellman property, it follows that the maximum value function V_η^* verifies the following:

$$\begin{aligned}
V_\eta^*(s_0) = \max_\pi V_\eta^\pi(s_0) &= \max_\pi \mathbb{E}_{s_1 \sim \mathcal{P}(s_0,a_0)} \Big[\big(\sum_{d=0}^{D} w_d \big) c(s_0, a_0) + V_{f(\eta)}^{T\pi}(s_1) \Big] \\
&= \max_{a_0, T\pi} \Big\{ \big(\sum_{d=0}^{D} w_d \big) c(s_0, a_0) + \mathbb{E}_{s_1 \sim \mathcal{P}(s_0,a_0)} \Big[V_{f(\eta)}^{T\pi}(s_1) \Big] \Big\} \\
&= \max_{a_0} \Big\{ \big(\sum_{d=0}^{D} w_d \big) c(s_0, a_0) + \max_{T\pi} \mathbb{E}_{s_1 \sim \mathcal{P}(s_0,a_0)} \Big[V_{f(\eta)}^{T\pi}(s_1) \Big] \Big\} \\
&= \max_{a_0} \Big\{ \big[\sum_{d=0}^{D} w_d \big] c(s_0, a_0) + \mathbb{E}_{s_1 \sim \mathcal{P}(s_0,a_0)} \big[V_{f(\eta)}^{*}(s_1) \big] \Big\}
\end{aligned}$$

Appendix B

Appendix of LGB

B.1 Pseudo-codes

Algorithms 7 and 8 present the high-level pseudo-codes of any algorithm following the LGB architecture for each of the three phases: the skill learning phase (G→B), the language grounding phase (L→G) and the instruction following phase (L→G→B).

Algorithm 7 LGB architecture G→B phase

```
▷ Goal → Behavior phase
Require Env $E$
Initialize policy $\Pi$, goal sampler $G_s$, buffer $B$
loop
    $g \leftarrow G_s$.sample()
    $(s, a, s', g, c_p, c'_p)_\tau \leftarrow E$.rollout($g$)
    $G_s$.update($c_p^T$)
    $B$.update($(s, a, s', g, c_p, c'_p)_\tau$)
    $\Pi$.update($B$)
return $\Pi, G_s$
```

Algorithm 8 LGB architecture L→G and L→G→B phases

```
▷ Language → Goal phase
Require $\Pi, E, G_s$, social partner $SP$
Initialize $LGG$
dataset $\leftarrow SP$.interact($E, \Pi, G_s$)
$LGG$.update(dataset)
return $LGG$
▷ Language → Behavior phase
Require $E, \Pi, LGG, SP$
loop
    instr. $\leftarrow SP$.listen()
    loop    ▷ Strategy switching loop
        $g \leftarrow LGG$.sample(instr., $c^0$)
        $c_p^T \leftarrow E$.rollout($g$)
        if $g == c_p^T$ then break
```

B.2 The DECSTR algorithm

B.2.1 Intrinsically Motivated Goal-Conditioned RL

Algorithm 9 presents the detailed pseudo-code of the sensorimotor learning phase (G→B) of DECSTR. It 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.

Algorithm 9 DECSTR: Sensorimotor Phase G→B.

```
Require: env $E$, # buckets $N_b$, # episodes before biased init. $n_{\text{unb}}$, self-evaluation probability $p_{\text{self\_eval}}$, noise function $\sigma()$
Initialize: policy $\Pi$, buffer $B$, goal sampler $G_s$, bucket sampling probabilities $p_b$, language module $LGG$.
loop
    self_eval ← random() $< p_{\text{self\_eval}}$        ▷ If $True$ then evaluate competence
    $g \leftarrow G_s$.sample(self_eval, $p_b$)
    biased_init ← $epoch < n_{\text{unb}}$        ▷ Bias initialization only after $n_{\text{unb}}$ epochs
    $s^0, c_p^0 \leftarrow E$.reset($biased\_init$)        ▷ $c_0$: Initial semantic configuration
    for $t = 1 : T$ do
        $a^t \leftarrow policy(s^t, c^t, g)$
        if not self_eval then
            $a^t \leftarrow a^t + \sigma()$
        $s^{t+1}, c_p^{t+1} \leftarrow E$.step($a^t$)
    episode ← $(s, c, a, s', c')$
    $G_s$.update($c^T$)
    $B$.update(episode)
    $g \leftarrow G_s$.sample($p_b$)
    batch ← $B$.sample($g$)
    $\Pi$.update(batch)
    if self_eval then
        $p_b \leftarrow G_s$.update_LP()
```

Policy updates with a Goal-Conditioned Soft Actor-Critic

Readers familiar with Markov Decision Process and the use of SAC and HER algorithms can skip this paragraph.

We want the DECSTR agent to explore a semantic configuration space and master reachable configurations in it. We frame this problem as a goal-conditioned MDP [Schaul et al., 2015]: $\mathcal{M} = (\mathcal{S}, \mathcal{G}_p, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where the state space $\mathcal{S}$ is the usual sensory space augmented with the configuration space $\mathcal{C}_p$, the goal space $\mathcal{G}_p$ is equal to the configuration space $\mathcal{G}_p = \mathcal{C}_p$, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the unknown transition probability, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \{0, 1\}$ is a sparse reward function and $\gamma \in [0, 1]$ is the discount factor.

Policy updates are performed with Soft Actor-Critic (SAC) [Haarnoja et al., 2018], 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 [Colas et al., 2019]. In our setting, we simply replace the originally targeted goal configuration by the currently achieved configuration in the transitions fed to SAC. 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. The HER mechanism substitutes the original goal with one of the configurations achieved later in the trajectory. This substitute 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 from the position of the objects in the inputs. The architecture used for the policy is depicted in Figure B.1.

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.

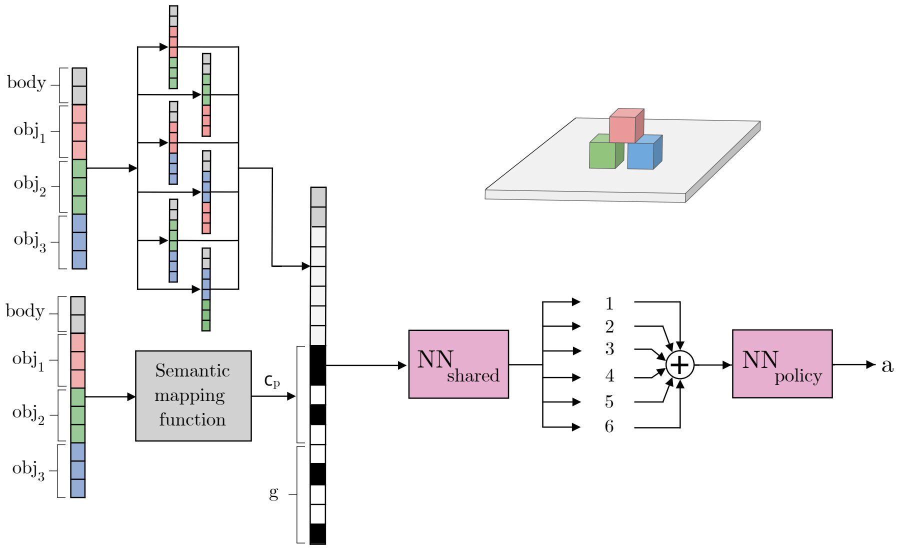


Figure B.1: Object-centered modular architecture for the policy.

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 3.3.3. 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., 2018]. Each of the 24 parallel worker 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 are 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 3.3.3 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 B.1.

Table B.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	$False$

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.

B.2.2 Language-Conditioned Goal Generator

In this section, we introduce useful details describing the *language goal generator* (LGG) used in Chapter 3.

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 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 3.3.3. 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 B.2. Note that *red*, *green* and *blue* refer to objects $o_1,\ o_2,\ o_3$ respectively.

Content of 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 where two pairs of blocks are close $c_i\ =\ [1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$ (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 is 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 B.2: List of instructions. Each of them specifies a shift of one predicate, either from false to true (0 → 1) or true to false (1 → 0). **block A** and **block B** represent two different blocks from {red, blue, green}.

Transition type	Sentences
Close $0 \to 1$ $(\times 3)$	*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 $1 \to 0$ $(\times 3)$	*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 $0 \to 1$ $(\times 6)$	*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 $1 \to 0$ $(\times 6)$	*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*.

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 → 1 (see Table B.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 NL sentence. The NL sentence is embedded with an 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 B.3). We found robust results for various layer sizes, various β below 1. and latent sizes above 9.

Table B.3: 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**]

B.3 Baselines and Oracle

The language-conditioned LB baseline is fully described in the main document.

B.3.1 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.

B.3.2 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. [2019]. The goal vector size is also 9 and contains the 3D target coordinates of the three blocks. This baselines 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 B.2. 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 are 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 by 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.

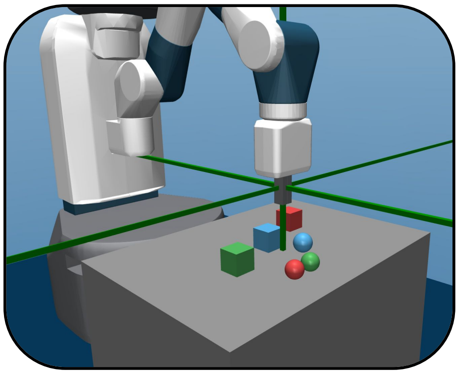


Figure B.2: The LGB-C baseline samples target positions for each block (example for a pyramid here).

B.4 Additional results

B.4.1 Comparison DECSTR - LGB-C in skill learning phase

Figure B.3 presents the average success rate over the 35 valid configurations during the skill learning phase for DECSTR and the LGB-C baseline. Because LGB-C cannot pursue semantic goals as such, we randomly sample a specific instance of this semantic goal: target block coordinates that satisfy the constraints expressed by it. 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, because of the error margins, does not achieve the original semantic goal.

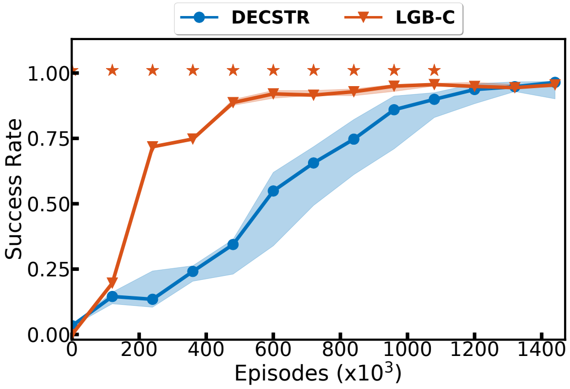


Figure B.3: Comparison DECSTR and LGB-C in the skill learning phase.

B.4.2 Automatic Bucket Generation

Figure B.4 depicts the evolution of the content of buckets along 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 respectively and grey are undiscovered configurations. At each moment, the discovered configurations are equally spread over the 5 buckets. A given configuration may thus 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, more configurations have been discovered but they are not always the same across runs. Finally, after 100 epochs, all configurations are found. Buckets are then steady and can be compared to expert-defined buckets. It seems that easier goals (top-most group) 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 to different buckets in different runs. This means that they are discovered in different orders depending on the runs. In summary, easier and harder goals from expert buckets 1 - 2 and 5 respectively seem to be well detected by our automatic bucket generations. Goals in medium-level expected difficulty as defined by expert buckets seem not to show any significant difference in difficulty for our agents.

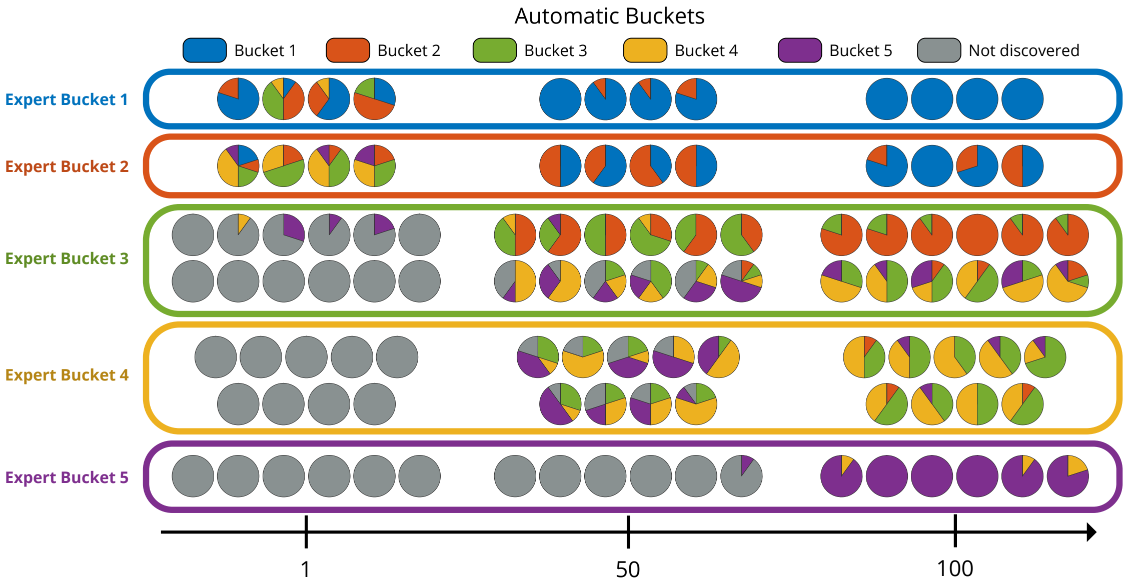


Figure B.4: Evolution of the content of buckets from automatic bucket generation: epoch 1 (2400 episodes, left), 50 (middle) and 100 (right). Each pie chart corresponds to one of the 35 valid configurations. It represents the distribution of the bucket attributions of that configuration across 10 runs. 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).

B.4.3 DECSTR Learning Trajectories

Figure B.5 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 3.5c shows, DECSTR performs much better than the *random goals* conditions. This proves that our automatic bucket algorithm generates useful goal clustering.

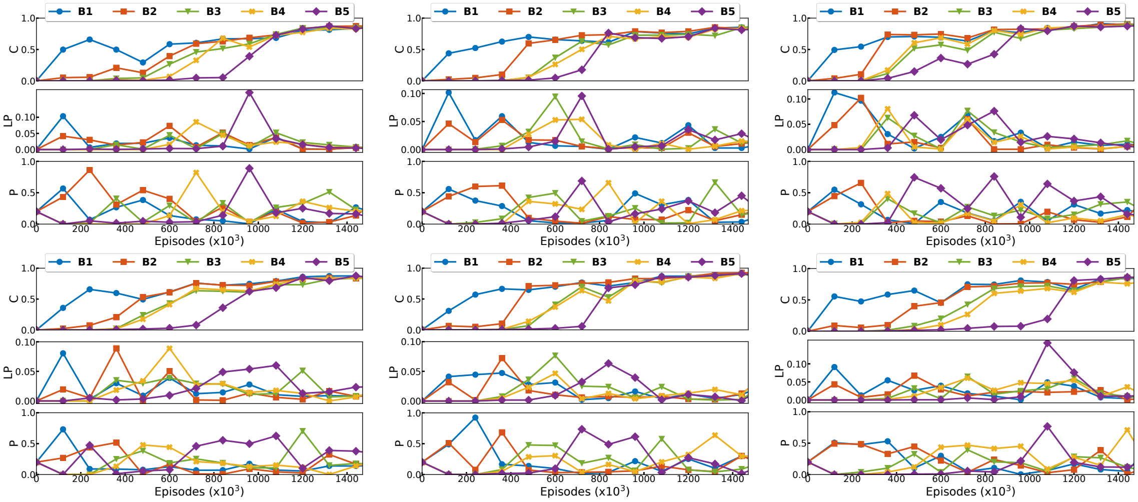


Figure B.5: Learning trajectories of 6 DECSTR agents.

Appendix C

Appendix of Autotelic Graph-based Agents

In this section, we present additional results and implementation details of the study conducted in Chapter 4 about transfer and generalization in autotelic agents.

C.1 Additional results

We introduce additional results which complement the ones presented in Chapter 4. More specifically, we study the relative importance of self-attention when using semantic goals.

C.1.1 Self-attention ablation

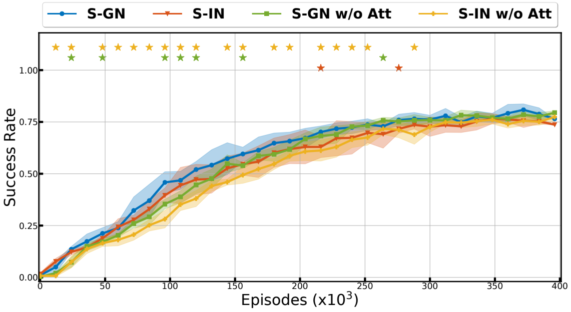


Figure C.1: Global SR across training episodes for S-GN, S-IN and their self-attention ablations counterparts. Mean ± standard deviations are computed over 5 seeds. Stars highlight statistical differences w.r.t S-GN agents (Welch's t-test with null hypothesis $\mathcal{H}_0$: no difference in the means, $\alpha = 0.05$.

We propose to remove the self-attention aggregation schemes from S-GN and S-IN,– the two best performing agents,– and introduce the corresponding ablations which use an unweighted sum when performing the pooling over edges or nodes.

Figure C.1 presents the global SR for these agents across training episodes. The differences only appear at the beginning of training. In fact, the global performance metrics in the ablations increase slower than their corresponding full-versions (blue vs green; red vs orange). However, all agents seem to behave similarly by the end of training. This suggests that self-attention improves sample efficiency, yielding GNN-based agents that can faster capture actionable relational features within their graphs.

C.2 Implementation details

In this part, we present details necessary to reproduce our results. We further open-source our code at https://github.com/akakzia/rlgraph_2.

GNN-based networks

Our four graph-based architectures use at most two shared networks, NN_{edge} and NN_{node}, respectively for computing updated edge features and node features. Both are 1-hidden-layer networks of hidden size 256. Taking the output dimension to be equal to 3× the input dimension for the shared networks showed better results. All networks use ReLU activation and the Xavier initialization. For edge-wise and node-wise aggregation, we use a one-headed self-attention module. Finally, to produce the output, all architectures use a readout network NN_{out}. The latter is also a 1-hidden-layer network of hidden size 256. For optimization, we use Adam with learning rates 10^{-3}. The list of hyperparameters is provided in Table C.1.

Parallel implementation of SAC-HER

All our experiments are based on a *Message Passing Interface* [Dalcin et al., 2011] to exploit multiple processors. Each of the 24 parallel workers maintains its own replay buffer of size 10^6 and performs its own updates. To synchronize experience between different workers, updates are summed over the 24 actors and the updated actor and critic networks are broadcast to all workers. Each worker alternates between 2 data collection episodes 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.

Table C.1: Hyperparameters used in this study.

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
rollouts_length	Number of episode steps per rollout	200
nb_updates	Number of updates per cycle	30
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	0.99
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
biased_init	Probability of following non-trivial scene reset scheme	0.2
self_eval_curriculum	Probability to perform self evaluations	0.1
curriculum_queue_length	Window over which LP estimations are made	1000

Appendix D

Appendix of HGEP

This appendix introduces additional material including implementation details, additional results and discussions for the HGEP study in Chapter 5. First, we present the hyper-parameters used in our experiments. Then, we present additional results. Finally, we provide a brief discussion around the additional results.

D.1 Implementation Details

We provide implementation details to reproduce our results in Table D.1. We use the Deep Deterministic Policy Gradient (DDPG) Algorithm ([Lillicrap et al., 2015]) with Hindsight Experience Replay (HER) strategy ([Andrychowicz et al., 2017]). For the fetch pick and place environment, we use an episode length of 50 and train the agents for 50 epochs. For the hand manipulation environments, we use an episode length of 100 and train agents for 200 epochs.

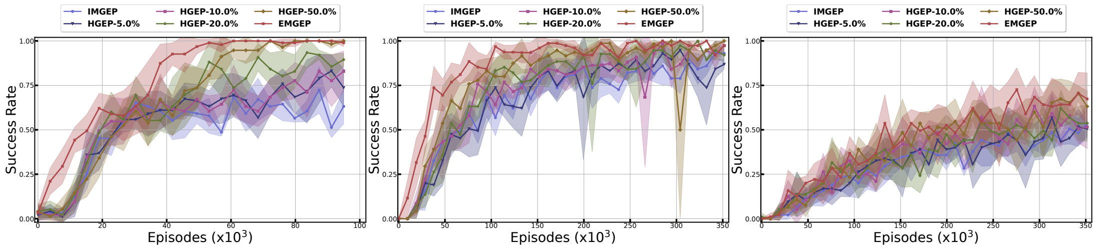
Table D.1: Hyperparameters used in the study on HGEPs.

Hyperparam.	Description	Values.
Workers	Number of workers	19
Cycles per epoch	Number of repeated cycles per epoch	50
Rollouts	Number of rollouts per worker	2
Updates	Number of updates per cycle	40
Replay Strategy	HER replay strategy	$future$
Replay k	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	0.99
τ	Polyak coefficient for target critics smoothing	0.95
lr_actor	Actor learning rate	10^{-3}
lr_critic	Critic learning rate	10^{-3}
ϵ_{random}	Probability of taking a random action during training	0.3
ϵ_{noise}	Exploration noise used to perturb actions taken by the policy	0.2
Action L2 Regularization	Coefficient of the L2 regularization	1

D.2 Additional Results

We first introduce additional details on the training success rate of the agents considered in the study of Chapter 5. Then, we introduce the special case of the hand reach environment.

D.2.1 Training Success Rate



(a) E1

(b) E2

(c) E3

Figure D.1: SR on externally generated goals in different environments.

Figure D.1 presents the success rate of the different agents considered in this study during training. The success rate is computed at each epoch by averaging 19 rollouts conducted on 19 different goals randomly generated by a hard-coded function within the environment. In all the considered environments, increasing the ratio of externally generated goals seems to increase the SR and sample efficiency. This is not surprising as by providing more external goals, the agent is more likely to train on goals that are closer to the goals on which it is evaluated. However, the E2 environment seems to require less externally generated goals to increase the agents SR. Recall that E2 corresponds to the hand manipulation of an egg-shaped object. The underlying goal space is easier to explore, since arbitrary finger movements are likely to move and rotate the object. As a result, such an environment does not necessarily require external goals to increase the SR, although external goals help increase the sample efficiency.

D.2.2 The Hand Reach Environment Case

Figure D.2 illustrates the *HandReach-v0* environment. The agent is an anthropomorphic robotic hand with 24 degrees of freedom, exactly as in environments E2 and E3 (See Chapter 5). The goal is a 15-dimensional vector defining a target position for each of the 5 fingers.

We present the evaluation metrics on the hand reach environment in Figure D.3.

- Figure D.3a presents estimations of the entropy of the distribution of discovered goals during training on the hand reach environment. As opposed to the other environments on which we evaluated our agents in the main document,

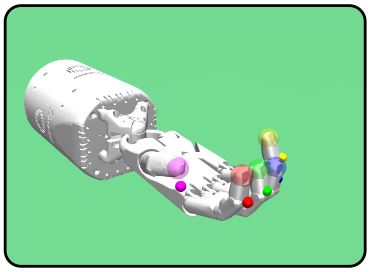
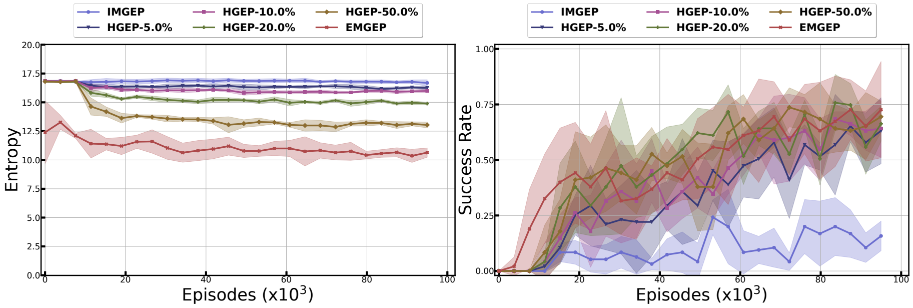


Figure D.2: Illustration of the HandReach-v0 (E4) environment



(a) SR on external goals.

(b) Entropy estimation during training.

Figure D.3: Evaluation Metrics on environment E4.

increasing the ratio of externally generated goals decreases the diversity of the discovered goals. The reason behind this is most likely related to the nature of the task. In fact, in the hand reach environment, moving the finger automatically changes the achieved goals. As a result, arbitrary actions are guaranteed to discover many goals. Increasing the externally provided goals will however constrain the distribution of discovered goals, especially since the exploration decreases with training.

- Figure D.3b shows the SR on the external goals during training on the hand reach environment. The IMGEP agent (blue curve), which only relies on physically discovered goals babbling, performs the worst. In fact, although this agent discovers a diverse set of goals (see entropy curve), these goals seem to be far in distribution term from those on which it is evaluated. By contrast, all other agents make use of the externally provided goals to adjust their distribution of discovered goals, and thus are able to increase their success rate

D.3 Discussion

The additional results we propose in this appendix show that combining different sources of goals is not trivial. In fact, although considering external goals seems to help increase the diversity of discovered goals in some environments, this is not true in other environments where arbitrary exploration is sufficient (for example the hand reach environment E4). Exclusively intrinsic agents that evolve in these environments are able by themselves to discover a sufficiently varied set of goals. However, when we evaluate them on the goals in which we are subjectively interested (the external goals which make sense for human designers), they are more likely to fail. Hence, there should be an adapted combination in the hybrid goal exploration process, where agents make use of their intrinsic motivation to discover a diverse set of goals, but still use the externally provided ones to learn more towards goals that make sense to human designers.

Appendix E

Appendix of HME

This section comes as a supplementary material for the study conducted in Chapter 6.

E.1 Proof of Concept Experiments on Mazes

In our preliminary experiments, we ran the HME agent in a set of continuous mazes that present hard exploration problems. These experiments act as a proof of concept for the more complete experiments presented in Chapter 6.

Environments

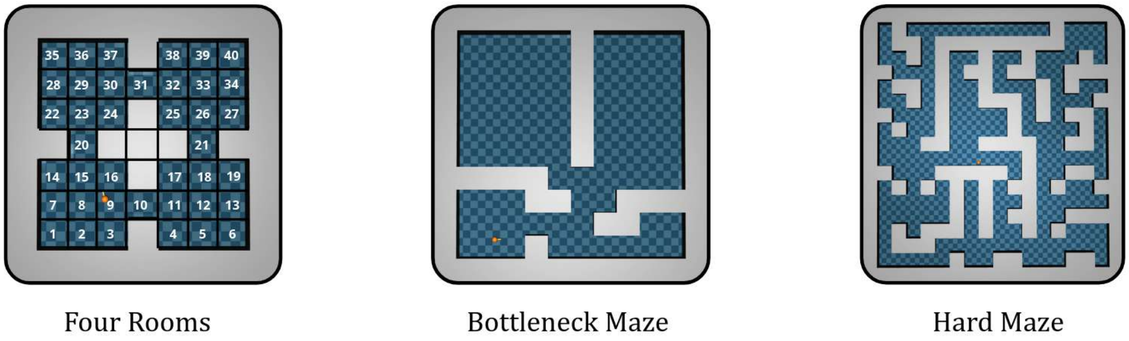
We consider a set of three maze navigation environments generated with an existing implementation[1] based on the MuJoCo physics engine (Todorov et al. [2012], see Figure E.1). The agent is represented as a moving point (orange point) that navigates specific areas (blue areas) delimited by thick walls (grey areas). Observations include the agent's position, velocity and orientation (6D). Actions control the velocity and orientation of the agent (3D). Goals include target positions for the agent to achieve (2D). Rewards are sparse such that the agent receives a reinforcing signal of +1 whenever they are within a threshold $\delta = 1cm$ from the target position.

The agent is given an extra representation function ϕ mapping the state to a discrete representation corresponding to the identity of an area or cell.

Experiments

These experiments demonstrate the benefit of using the HME interaction protocol. We use a simpler version of the HME agent that replaces the active query mechanism with a fixed probability to query the social partner and which does not internalize socially-suggested goals. On the one hand, we use simple flat MLP networks to model the agent's internal models, including its actor and critic. On the other hand, to

[1] https://github.com/kngwyu/mujoco-maze



Four Rooms Bottleneck Maze Hard Maze

Figure E.1: The 4 mazes considered in our study. They are of increasing difficulty in exploration.

model the SP's suggested goals, we construct and annotate cells from the bottom row upwards (see Figure E.1). The goals that the SP can propose correspond to the centers of these annotated cells. As the goal space is not known *a priori*, the agent has to 1) explore to discover goals and 2) exploit to learn to reach goals. At the beginning of each episode, the agent starts at the center of cell number 1.

We experiment with different levels of sociality: SP $p\%$ where p is the probability to query the social partner (SP) with $p \in \{0, 1, 10, 100\}\%$. SP 0% and SP 100% represent the individual and social baseline respectively. We use three metrics to evaluate the strategies: 1) the global success rate across all the goal space (known & unknown goals), 2 the local success rates for each goal (heat maps on the mazes) and 3) the goal space coverage (ratio of discovered goals).

Results

Figure E.2, presents the results for the first metric (global success rate across all goal space). The results unequivocally show that the presence of the SP helps the agent reach a better performance. With only 10% of social episodes, the agent is able to master all goals on *Corridor* and *Four Rooms*. Increasing the number of social episodes does not further improve performance, as the agent with 100% and 10% social episodes performs similarly. Notably, using only 1% of social episodes is enough to show satisfying performance, further highlighting the importance of the social partner to discover new goals.

In order to illustrate that the discovery of new goals drives performance, we analyzed the exploration metrics across trained agents in Figures E.3 and E.4. Note that adding as little as 1% of social episodes in all mazes drastically improves the local success rate on goals that go beyond the first room. Indeed, thanks to the SP, the agents are able to pass the bottleneck of the maze and discover the other rooms, while SP 0% stays in the first room because it is not able to explore efficiently. The ratio of discovered goals as a function of time plotted in Figures E.3 and E.4 shows that the agents with social episodes rapidly discover most of the goals while the

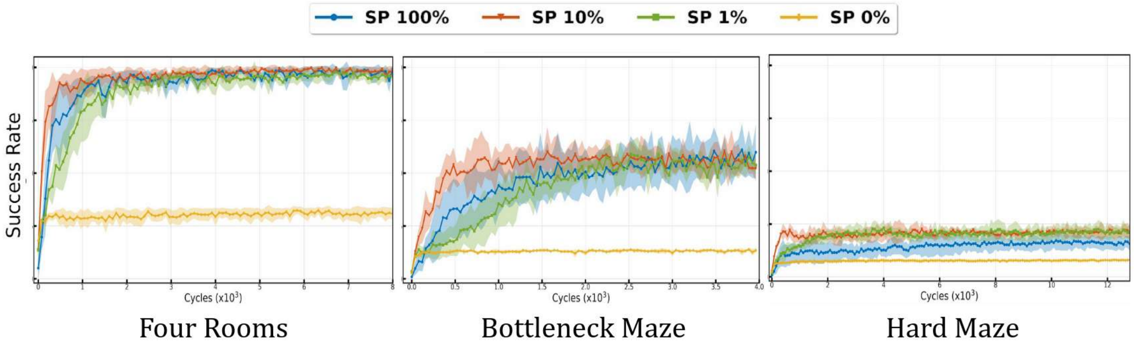


Figure E.2: Global success rate across all the goal space (known & unknown goals) on the four tested mazes.

agent without social episodes does not. For the Hard maze (Figure E.4, which is considered the most complex among the set of mazes we have chosen, 1% of social episodes enables the ratio of discovered goals to catch up with the ones with 10% and 100% of social episodes. However, the agent is still unable to maximize its SR on all the cells (middle part of Figure E.4).

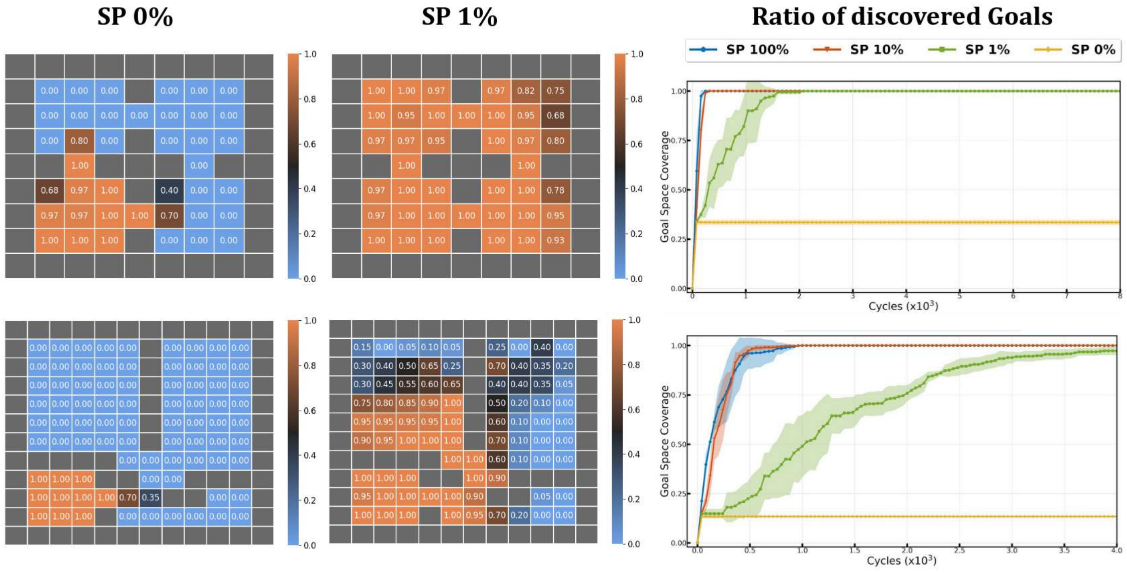


Figure E.3: Left: Local success rates for each goal on the Bottleneck maze. Right: ratio of discovered goals on Four Rooms and Bottleneck mazes.

E.2 Additional Results

In this section, we introduce additional evaluation metrics that accompany the study conducted in Chapter 6.

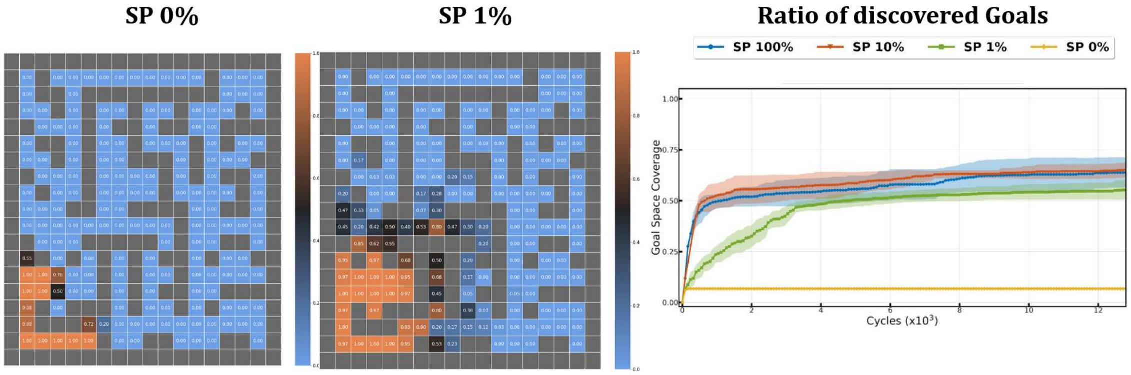


Figure E.4: Performance Metrics on the Hard maze. Left: SR per cell with 0% of social interventions; Middle: SR per cell with 1% of social interventions; Right: Coverage of the goal space computed as the ratio of discovered goals.

E.2.1 Internalization Study

To assess the relative importance of the internalization mechanism with which are endowed our autotelic agents, we introduce five variants of our algorithm:

- IN-1: **Rehearse and Generate.** When these agents fail at reaching the social beyond goals, they memorize them in a separate buffer (other than the one they use to store their own encountered goals). They do not keep track of the associated frontier goal. When performing autotelic episodes, they can choose to rehearse these beyond goals. When they do so, they generate an intermediate sub-goal that maximizes their reward estimator.

- IN-2: **Rehearse Pairs.** When failing the social beyond goals, these agents store both the failed goals and the associated frontier goals proposed by the SP. During autotelic episodes, they can rehearse the stored pair of goals, as if the SP was actually there to re-propose it.

- IN-3: **Rehearse Beyond.** Similar to IN-1. However, when performing rehearsal of social beyond goals during autotelic episodes, these agents directly target the selected goal (without stepping on an intermediate sub-goal).

- IN-4: **No attention.** When these agents fail at reaching the social beyond goals, they store them in the *same* buffer as their own physically encountered goals. When performing autotelic episodes, these agents uniformly select a goal from this shared buffer, without any particular attention to how they were discovered.

- w/o IN: **No internalization.** These agents simply forget their failed social beyond goals. When performing autotelic episodes, they exclusively focus on the goals that they have physically encountered on their own.

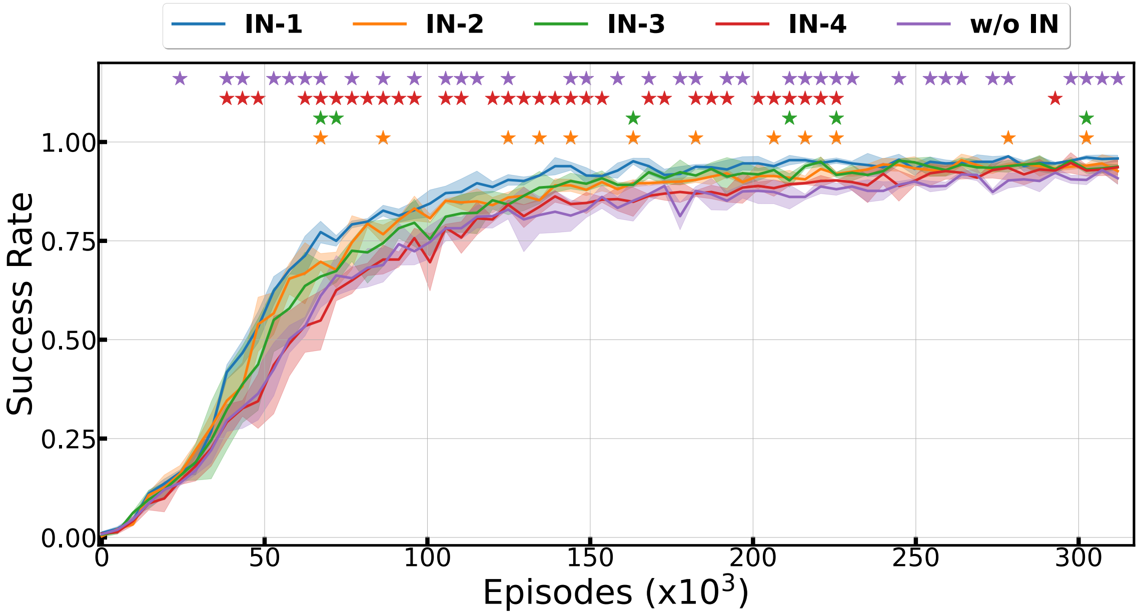


Figure E.5: Global SR across training episodes for different variants of the internalization mechanism.

Internalize vs No Internalize

Agents w/o IN, which immediately forget about the proposed social goals, show the worst performance (purple curve on Figure E.5). Besides, they present the highest statistical differences with reference the best performing agents IN-1 (purple stars). This suggests that storing the proposed social goals helps improve performance. This is not surprising since the internalizing agents learn not only about their physically encountered goals, but also about the socially proposed ones.

Attention vs No Attention to internalized pairs

Agents IN-4 do not use any type of attention to the proposed social goals. They only store them in the same buffer as the ones they have physically encountered. Although this improves the performance at the end of the training budget (IN-4 vs w/o IN), these agents are significantly slower than the ones that use attention by storing social goals in a different buffer (IN-1, IN-2 and IN-3).

Internalize Pairs vs Beyond

While the IN-2 agents store both the proposed frontier and beyond goals, the IN-3 agents only internalize the beyond goals. Comparing the global performance metrics of these agents (orange vs green curves) yields two interpretations: 1) storing the frontier pairs is helpful in the first phase of the training (by episode 120k), since the orange curve is above the green one; 2) once the agents acquire more skills, they are less dependent on the proposed frontier goal.

Rehearse Frontier vs Use own Frontier

The IN-1 agents, which generate their own intermediate known goal as a stepping stone to an internalized beyond goals, exhibit the best global performance (blue curve). Besides, Table E.1 shows that these agents manage to discover even more goals than the other agents. This suggests that relying on their own generated intermediate goals makes them less grounded in the SP's proposals, which further promotes their exploration of the goal space.

Table E.1: Amount of Social Interventions and Discovered Goals for the studied internalization variants.

Agents	IN-1	IN-2	IN-3	IN-4	w/o IN
% Social Episodes	7.42 ± 0.08	6.98 ± 0.95	7.05 ± 0.49	8.80 ± 1.60	7.54 ± 0.91
# Proposed Beyond Goals	1878 ± 221	1280 ± 114	1113 ± 173	951 ± 268	6478 ± 1623
# Discovered Goals	5833 ± 26	4306 ± 335	4298 ± 130	4403.0 ± 331	4177 ± 74

E.2.2 Induced Learned Trajectories

To zoom on the effect of social goal proposals on the agents' learning trajectories, we consider the best performing agents $\beta = 50$. Figure E.6 presents their learning trajectories for the classes S_3, S_4 and S_5. We only consider these classes because 1) they are part of the assignment goal space $\mathcal{G}_{SP}$ and 2) they are not trivial to be encountered by the agent alone through random exploration. Interestingly, the learning trajectories of the three considered agents shows a pattern: first, the SP suggests social goals (dashed lines), then, the agent immediately starts to physically encounter and reach the proposed goals (dotted lines) and finally, the SR of the considered class increases (plain lines). This suggests that the role of the SP is to help the agent encounter the least trivial goals that are hard to reach through random exploration. Once this is done, the agent can target these goals and efficiently master them.

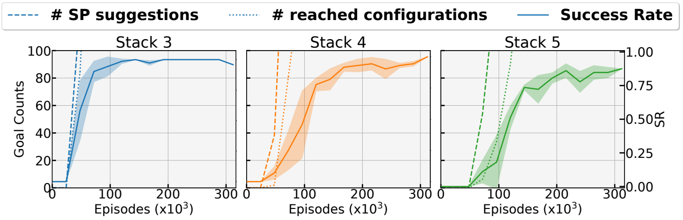


Figure E.6: Learning trajectories with $\beta = 50$. We report the # of goals proposed by SP (dashed), the # of times the agents encountered them (dotted) and their success rate (plain).

E.3 Representing the social partner's knowledge

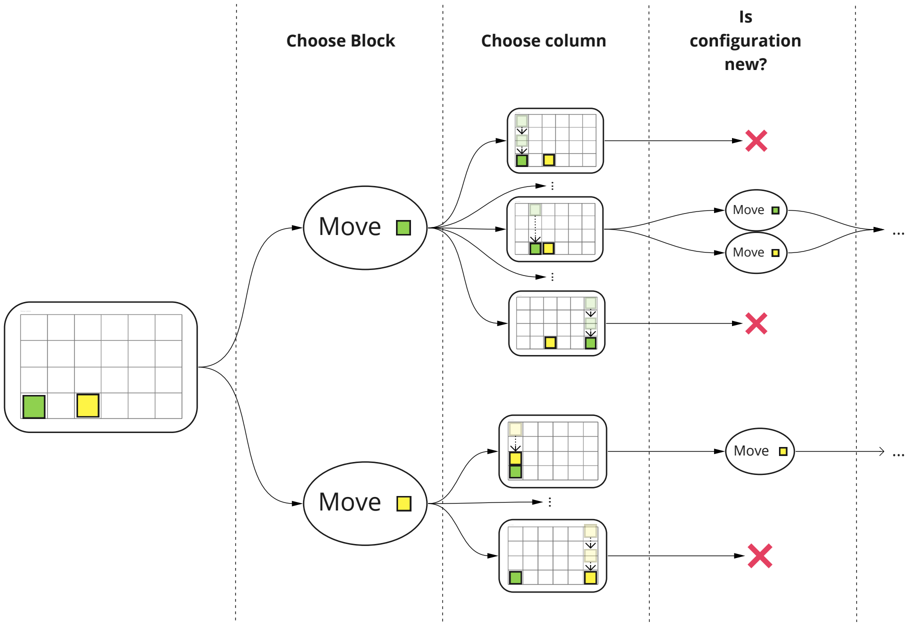


Figure E.7: Simplified example with 2 blocks in a 2D grid of the oracle tree construction procedure

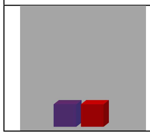
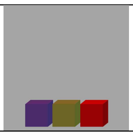
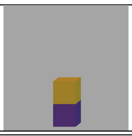
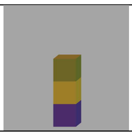
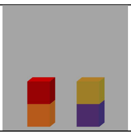
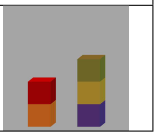
We model the social partner (SP) by a hard-coded program endowed with the set of goals that are potentially achievable by agent within the five-object manipulation benchmark. We choose to represent this domain knowledge as a directed semantic graph. This facilitates determining stepping stones in the agent's capabilities. However, it is impractical to manually enlist all the imaginable configurations and decide whether a pair should be linked or not in a graph. Semantic configurations can be infeasible for two reasons: 1) they are semantically impossible to obtain—e.g. two objects cannot both be above each other 2) they are physically impossible to achieve—e.g. in the case of inverted pyramids.

To avoid enlisting all semantic and physical constraints, we define two types of the SP's goal space:

- **Oracle Goal Space**: we design a 3D grid within which each cell can contain one object. Initially, all the blocks are initialized in different columns of the grid so that they are all far from each other (*root* node). From this state, we can select one object and move it to another column. If that column already contains another object, than the first one will be stacked above the second. By doing one action at each step, we can extract the current semantic configuration and link it to the previous one in the oracle graph. Iteratively repeating this process yields a tree starting from the *root* node. See Figure E.7 for a 2D illustration of the described process for two blocks. The number of nodes within the oracle goal space is valued to 12666.
- **Minimal Goal Space**: this represents a sub-graph of the oracle goal space defined above. It consists of a set of goal configurations exclusively including the path from no stacks at all to stacks of 5 objects for each permutation of objects (No stack at all → Stack of 2 → ... → Stack of 5). The total number of nodes within this minimal goal space is valued to 321.

In our main experiments, we use the *minimal goal space* to represent the SP's knowledge in all our agents except for the *social baseline.* In fact, our experiments have shown that the oracle goal space is needed when training includes *exclusively* social episodes. Note that this engineered process only serves to evaluate the capacities of the agent and is not used by the agent itself in any way.

E.4 Evaluation Classes

Close 1	Close 2	Stack 2	Stack 3	Stack 2&2	Stack 2&3
					

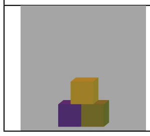
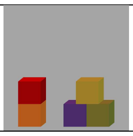
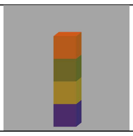
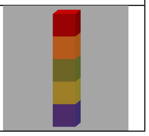
Pyramid	Pyr 3&Stack2	Stack 4	Stack 5
			

Table E.2: The different semantic classes used in evaluation. The class *Close i* regroups all semantic configurations where *i* pairs of blocks are close.

Table E.2 illustrates the 11 evaluation classes presented in the main paper. For the sake of simplicity here, we only represent the blocks that are concerned by the underlying predicates. All the predicates associated with the other blocks have values set to 0. We use a hard-coded function to generate a random configuration given the identifier of the considered class. We also use a dictionary where keys are configurations and values are identifiers of the classes to keep counts of either the SP proposed goals or the agent's encountered and achieved goals.

E.5 Object-centered architecture in Fetch Manipulate

In the *Fetch Manipulate* environment with 5 blocks we use the GANGSTR agent proposed in Akakzia and Sigaud [2022]. It perceives:

1. the low-level geometric states. Since the 5 objects share the same attributes dimensions (positions, velocities, orientations), the behavior with respect to an object should be independent from the object's attributes.

2. the high-level semantic configurations. Since the relations between all the pairs of objects share the same predicates (*close*, *above*), the behavior with respect to a binary semantic relation should be independent from the considered pair.

Thus, it is natural to encode both object-centered and relational inductive biases in our architecture. To do so, we model both the policies and critics of the agents as message passing graph neural networks (MPGNNs) [Gilmer et al., 2017]. We consider a graph of 5 nodes, each representing a single object. All the nodes are interconnected, yielding a compact graph of 20 directed edges. Furthermore, we consider the agent's body attributes as global features of the policy networks and both the agent's body attributes and the actions as global features of the critic networks. A single forward pass through this graph consists in three steps:

1. **Message computation** is performed for each edge. The features of the considered edge are concatenated with the features of the edge's source and target nodes before being fed to a first shared neural network NN_{edge}.

2. **Node-wise aggregation** is performed for each node. The features of the considered node are concatenated with an aggregation of the updated features of all the incoming edges. The resulting vectors are then concatenated with the global features of the graph before being fed to a second shared neural network NN_{node}.

3. **Graph-wise aggregation** is performed once for all the graph. The updated features of all the nodes of the graph are aggregated and fed to a third neural network NN_{readout}.

The aggregating function needs to be permutation-invariant. We use max pooling for the *node-wise aggregation* and summation for the *graph-wised aggregation*. The final output of NN_{readout} is either the action (in the case of the actor) or the Q-value (in the case of the critic).

E.6 Pseudo code

Algorithms 10 and 11 present the high-level pseudo-code for the individual and social learning episodes.

Algorithm 10 Individual Learning

```
**Require** Env $E$,
Initialize policy $\Pi$, semantic graph $\mathcal{G}_s$, path estimator $PE$, buffer $B$.
**loop**
    $g \leftarrow \mathcal{G}_s$.sample_node()
    $path \leftarrow PE$.sample_path($g, \mathcal{G}_s$)
    **loop** $g_i \in path$
        $trajectory \leftarrow E$.rollout($g_i$)
        $\mathcal{G}_s$.update($trajectory$)
        $PE$.update($trajectory$)
        $B$.update($trajectory$)
    $\Pi$.update($B$)
**return** $\Pi, PE, \mathcal{G}_s$
```

Algorithm 11 Social Learning

```
Require Env $E$, social partner $SP$
Initialize policy $\Pi$, semantic graph $\mathcal{G}_s$, path estimator $PE$, buffer $B$.
loop
    $g \leftarrow SP$.propose_goal($\mathcal{G}_s$)
    $path \leftarrow PE$.sample_path($g, \mathcal{G}_s$)
    loop  $g_i \in path$
        $trajectory \leftarrow E$.rollout($g_i$)
        $\mathcal{G}_s$.update($trajectory$)
        $PE$.update($trajectory$)
        $B$.update($trajectory$)
    if $g$ is a stepping stone and is reached then
        $g_b \leftarrow SP$.propose_unknown($\mathcal{G}_s$, $g$)
        $path \leftarrow PE$.sample_path($g_b, \mathcal{G}_s$)
        loop  $g_i \in path$
            $trajectory \leftarrow E$.rollout($g_i$)
            $\mathcal{G}_s$.update($trajectory$)
            $PE$.update($trajectory$)
            $B$.update($trajectory$)
    $\Pi$.update($B$)
return $\Pi, PE, \mathcal{G}_s$
```

E.7 Implementation Details

This part includes details necessary to reproduce the results. An anonymous version of our code will be made available at https://anonymous.4open.science/r/gangstr-2C4F.

GNN-*based networks.* Our object-centered architecture uses two shared networks, NN_{edge} and NN_{node}, respectively for the message computation and node-wise aggregation. Both are 1-hidden-layer networks of hidden size 256. Taking the output dimension to be equal to 3× the input dimension for the shared networks showed better results. 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 E.3.

Parallel implementation of SAC-HER*.* Our experiments rely on a *Message Passing Interface* [Dalcin et al., 2011] to exploit multiple processors. 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 actor and critic networks are broadcast to all workers. Each worker alternates between 10 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.

Table E.3: Hyperparameters used in GANGSTR.

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	10
rollouts_length	Number of episode steps per rollout	40
nb_updates	Number of updates per cycle	30
replay_strategy	HER replay strategy	$final$
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	0.99
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
α_{EMA}	EMA coefficient for SR edge estimation	0.01
edge_prior	Default value for edges' SR	0.5
$shortest_paths$	Number of shortest paths to sample from	5
shortest_safest_ratio	Ratio of alternation between shortest and safest paths	0.5

Appendix F

References of Figure 1.9

Table F.1: References of Figure 1.10

Label	Reference
UVFA	Schaul et al. [2015]
SR	Trott et al. [2019]
PlanGAN	Charlesworth and Montana [2020]
AIM	Durugkar et al. [2021]
I-HER	McCarthy and Redmond [2021]
GCSL	Ghosh et al. [2019]
L3P	Zhang et al. [2020]
GAP	Nair et al. [2019]
LEXA	Mendonca et al. [2021]
DDL	Hartikainen et al. [2019]
MAPGo	Zhu et al. [2021]
HER	Andrychowicz et al. [2017]
HER with Demos	Nair et al. [2018]
CHER	Fang et al. [2019]
G-HER	Bai et al. [2019]
GDP	Kuang et al. [2020]
IMAGINE	Colas et al. [2020b]
MC-HER	Lanier et al. [2019]
GNGSTR	Akakzia and Sigaud [2022]
MEGA	Pitis et al. [2020]
Skew-Fit	Pong et al. [2019]
SoRB	Eysenbach et al. [2019]
GoalGAN	Florensa et al. [2018]
Sub-goal Discovery	Paul et al. [2019]
HGG	Ren et al. [2019]
Hindsight Planner	Lai et al. [2020]
VDS	Zhang et al. [2020]
RIG	Nair et al. [2018]
DISCERN	Warde-Farley et al. [2018]
HVF	Nair et al. [2020]
AMIGo	Campero et al. [2020]
DECSTR	Akakzia et al. [2021]
HME	Akakzia et al. [2022]

Bibliography

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In Carla E. Brodley, editor, *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. doi: 10.1145/1015330.1015430.

Jinane Abounadi, Dimitrib Bertsekas, and Vivek S Borkar. Learning algorithms for markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40(3):681–698, 2001.

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

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational autoencoding learning of options by reinforcement. In *NIPS Deep Reinforcement Learning Symposium*, 2017a.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017b.

Ahmed Akakzia and Olivier Sigaud. Learning object-centered autotelic behaviors with graph neural networks. *arXiv preprint arXiv:2204.05141*, 2022.

Ahmed Akakzia, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. DECSTR: Learning goal-directed abstract behaviors using preverbal spatial predicates in intrinsically motivated agents. *ArXiv preprint*, abs/2006.07185, 2020a. URL https://arxiv.org/abs/2006.07185.

Ahmed Akakzia, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Grounding language to autonomously-acquired skills via goal generation. *arXiv preprint arXiv:2006.07185*, 2020b.

Ahmed Akakzia, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Grounding language to autonomously-acquired skills via goal generation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=chPj_I5KMHG.

Ahmed Akakzia, Olivier Serris, Olivier Sigaud, and Cédric Colas. Help me explore: Minimal social interventions for graph-based autotelic agents. *arXiv preprint arXiv:2202.05129*, 2022.

Muhannad Alomari, Paul Duckworth, David C Hogg, and Anthony G Cohn. Natural language acquisition and grounding for embodied robotic systems. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Lee Altenberg et al. The evolution of evolvability in genetic programming. *Advances in genetic programming*, 3:47–74, 1994.

Eitan Altman. *Constrained Markov decision processes: stochastic modeling.* Routledge, 1999.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*, 2016.

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5048–5058, 2017.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5): 469–483, 2009.

Rudolf Arnheim. *Visual thinking.* Univ of California Press, 1997.

David N Aspin and Judith D Chapman. Lifelong learning: concepts and conceptions. *International Journal of lifelong education*, 19(1):2–19, 2000.

Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Arian Hosseini, Pushmeet Kohli, and Edward Grefenstette. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:1806.01946*, 2018.

Dzmitry Bahdanau, Harm de Vries, Timothy J O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. Closure: Assessing systematic generalization of clevr models. *arXiv preprint arXiv:1912.05783*, 2019.

Chenjia Bai, Peng Liu, Wei Zhao, and Xianglong Tang. Guided goal generation for hindsight multi-goal reinforcement learning. *Neurocomputing*, 359:353–367, 2019.

Renee Baillargeon. Representing the existence and the location of hidden objects: Object permanence in 6-and 8-month-old infants. *Cognition*, 23(1):21–41, 1986.

Gianluca Baldassarre and Marco Mirolli. *Intrinsically motivated learning in natural and artificial systems*. Springer, 2013.

Gianluca Baldassarre, Tom Stafford, Marco Mirolli, Peter Redgrave, Richard M Ryan, and Andrew Barto. Intrinsic motivations and open-ended development in animals, humans, and robots: an overview. *Frontiers in psychology*, 5:985, 2014.

Albert Bandura and David C. McClelland. *Social learning theory*, volume 1. Englewood cliffs Prentice Hall, 1977.

Victor Bapst, Alvaro Sanchez-Gonzalez, Carl Doersch, Kimberly Stachenfeld, Pushmeet Kohli, Peter Battaglia, and Jessica B. Hamrick. Structured agents for physical construction. In *International Conference on Machine Learning*, pages 464–474. PMLR, 2019.

Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013a.

Adrien Baranes and Pierre-Yves Oudeyer. Active Learning of Inverse Models with Intrinsically Motivated Goal Exploration in Robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013b. Publisher: Elsevier.

Andrew G Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer, 2013.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Jonathan Baxter and Peter L Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29:1471–1479, 2016.

Beth Kemp Benson. Scaffolding. *English Journal*, 86(7):126, 1997.

Laura E Berk. Why children talk to themselves. *Scientific American*, 271(5):78–83, 1994a.

Laura E Berk. Why Children Talk to Themselves. *Scientific American*, 271(5): 78–83, 1994b. Publisher: JSTOR.

Daniel E Berlyne. Curiosity and exploration. *Science*, 153(3731):25–33, 1966.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

David Blackwell. Discrete dynamic programming. *The Annals of Mathematical Statistics*, pages 719–726, 1962.

Andreea Bobu, Marius Wiggert, Claire Tomlin, and Anca D Dragan. Feature expansive reward learning: Rethinking human input. *ArXiv preprint*, abs/2006.13208, 2020. URL https://arxiv.org/abs/2006.13208.

Richard S Bogartz, Jeanne L Shinskey, and Cindy J Speaker. Interpreting infant looking: The event set× event set design. *Developmental psychology*, 33(3):408, 1997.

Matthew Botvinick, Sam Ritter, Jane X Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23(5):408–422, 2019.

Jean-David Boucher, Ugo Pattacini, Amelie Lelong, Gerard Bailly, Frederic Elisei, Sascha Fagel, Peter F Dominey, and Jocelyne Ventre-Dominey. I reach faster when i see you look: gaze effects in human–human and human–robot face-to-face cooperation. *Frontiers in neurorobotics*, 6:3, 2012.

Cynthia Breazeal and Andrea L. Thomaz. Learning from human teachers with socially guided exploration. In *2008 IEEE International Conference on Robotics and Automation*, pages 3539–3544. IEEE, 2008.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Jerome Bruner. Child's Talk: Learning to Use Language. *Child Language Teaching and Therapy*, 1(1):111–114, 1985. Publisher: SAGE Publications Sage UK: London, England.

Jerome Bruner. The narrative construction of reality. *Critical inquiry*, 18(1):1–21, 1991.

Jerome S Bruner. Organization of early skilled action. *Child development*, pages 1–11, 1973.

Jerome S. Bruner. *The process of education*. Harvard University Press, 2009.

Jerome Seymour Bruner. *Acts of meaning*, volume 3. Harvard University Press, 1990.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*, 2020.

Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giro-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020.

Angelo Cangelosi and Matthew Schlesinger. From babies to robots: the contribution of developmental robotics to developmental psychology. *Child Development Perspectives*, 12(3):183–188, 2018.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Susan Carey. The origin of concepts. *Journal of Cognition and Development*, 1(1): 37–41, 2000.

Susan Carey. Where our number concepts come from. *The Journal of philosophy*, 106(4):220, 2009.

NR Carlson, W Buskist, ME Enzle, and CD Heth. Psychology: the science of behaviour (3rd canadian ed.), 2005.

Richard Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Hugo Caselles-Dupré, Olivier Sigaud, and Mohamed Chetouani. Pragmatically learning from pedagogical demonstrations in multi-goal environments. *arXiv preprint arXiv:2206.04546*, 2022.

Carlos Celemin and Javier Ruiz-del Solar. Coach: learning continuous actions from corrective advice communicated by humans. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 581–586. IEEE, 2015.

Tathagata Chakraborti, Subbarao Kambhampati, Matthias Scheutz, and Yu Zhang. Ai challenges in human-robot cognitive teaming. *ArXiv preprint*, abs/1707.04775, 2017. URL https://arxiv.org/abs/1707.04775.

Harris Chan, Yuhuai Wu, Jamie Kiros, Sanja Fidler, and Jimmy Ba. Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning. *ArXiv preprint*, abs/1902.04546, 2019. URL https://arxiv.org/abs/1902.04546.

Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

Henry Charlesworth and Giovanni Montana. Plangan: Model-based planning with sparse rewards and multiple goals. *Advances in Neural Information Processing Systems*, 33:8532–8542, 2020.

Alexandre Chenu, Nicolas Perrin-Gilbert, and Olivier Sigaud. Divide & conquer imitation learning. *arXiv preprint arXiv:2204.07404*, 2022.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Soonja Choi, Laraine McDonough, Melissa Bowerman, and Jean M Mandler. Early sensitivity to language-specific spatial categories in english and korean. *Cognitive Development*, 14(2):241–268, 1999.

Noam. Chomsky. *Syntactic Structures*. Mouton, 1957. ISBN 978-90-279-3385-0.

Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

Junyi Chu and Laura E. Schulz. Play, curiosity, and cognition. *Annual Review of Developmental Psychology*, 2:317–343, 2020.

Geoffrey Cideron, Mathieu Seurin, Florian Strub, and Olivier Pietquin. Self-educated language agent with hindsight experience replay for instruction following. *arXiv preprint arXiv:1910.09451*, 2019.

Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4693–4700. IEEE, 2018.

Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338, 2019.

Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340. PMLR, 2019.

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

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter F. Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity driven exploration. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b.

Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Intrinsically motivated goal-conditioned reinforcement learning: a short survey. *ArXiv preprint*, abs/2012.09830, 2020c. URL https://arxiv.org/abs/2012.09830.

Cédric Colas, Tristan Karch, Clément Moulin-Frier, and Pierre-Yves Oudeyer. Vygotskian autotelic artificial intelligence: Language and culture internalization for human-like ai. *arXiv preprint arXiv:2206.01134*, 2022a.

Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, 74:1159–1199, 2022b.

Céédric Colas. *Towards Vygotskian Autotelic Agents: Learning Skills with Goals, Language and Intrinsically Motivated Deep Reinforcement Learning.* PhD thesis, Université de Bordeaux, 2021.

Gergely Csibra and György Gergely. Natural pedagogy. *Trends in cognitive sciences*, 13(4):148–153, 2009.

Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2): 245–259, 2017.

Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In *International conference on machine learning*, pages 1106–1114. PMLR, 2018.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Lisandro D. Dalcin, Rodrigo R. Paz, Pablo A. Kler, and Alejandro Cosimo. Parallel distributed computing using python. *Advances in Water Resources*, 34(9):1124–1139, 2011.

Kerstin Dautenhahn. Getting to know each other—artificial social intelligence for autonomous robots. *Robotics and autonomous systems*, 16(2-4):333–356, 1995.

Nathaniel D Daw and Kenji Doya. The computational neurobiology of learning and reward. *Current opinion in neurobiology*, 16(2):199–204, 2006.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. *Robotics: Science and Systems VII*, pages 57–64, 2011.

Daniel Clement Dennett. *The intentional stance*. MIT press, 1987.

AH Dickenson, CM Brewer, and NA Hayes. Effects of topical baclofen on c fibre-evoked neuronal activity in the rat dorsal horn. *Neuroscience*, 14(2):557–562, 1985.

Anthony H Dickenson, Daniel Le Bars, and Jean Marie Besson. Diffuse noxious inhibitory controls (dnic). effects on trigeminal nucleus caudalis neurones in the rat. *Brain Research*, 200(2):293–305, 1980.

Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

Stephane Doncieux, David Filliat, Natalia Díaz-Rodríguez, Timothy Hospedales, Richard Duro, Alexandre Coninx, Diederik M. Roijers, Benoît Girard, Nicolas Perrin, and Olivier Sigaud. Open-ended learning: a conceptual framework based on representational redescription. *Frontiers in Robotics and AI*, 12, 2018. doi: 10.3389/fnbot.2018.00059.

Kenji Doya. Efficient nonlinear control with actor-tutor architecture. *Advances in neural information processing systems*, 9, 1996.

Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. Legibility and predictability of robot motion. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 301–308. IEEE, 2013.

Ishan Durugkar, Mauricio Tec, Scott Niekum, and Peter Stone. Adversarial intrinsic motivation for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:8622–8636, 2021.

Marc Ebner, Mark Shackleton, and Rob Shipman. How neutral networks influence evolvability. *Complexity*, 7(2):19–33, 2001.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

Manfred Eppe, Christian Gumbsch, Matthias Kerzel, Phuong DH Nguyen, Martin V Butz, and Stefan Wermter. Intelligent problem-solving as integrated hierarchical reinforcement learning. *Nature Machine Intelligence*, pages 1–10, 2022.

Mayalen Etcheverry, Clément Moulin-Frier, and Pierre-Yves Oudeyer. Hierarchically organized latent modules for exploratory search in morphogenetic systems. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Tom Everitt and Marcus Hutter. Avoiding wireheading with value reinforcement learning. In *International Conference on Artificial General Intelligence*, pages 12–22. Springer, 2016.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Ben Eysenbach, Xinyang Geng, Sergey Levine, and Russ R Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. *Advances in neural information processing systems*, 33:14783–14795, 2020.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. *Advances in neural information processing systems*, 32, 2019.

Teresa Farroni, Gergely Csibra, Francesca Simion, and Mark H. Johnson. Eye contact detection in humans from birth. *Proceedings of the National Academy of Sciences*, 99(14):9602–9605, 2002. ISSN 0027-8424. doi: 10.1073/pnas.152159999.

Tiffany M Field, Robert Woodson, Debra Cohen, Reena Greenberg, Robert Garcia, and Kerry Collins. Discrimination and imitation of facial expressions by term and preterm neonates. *Infant Behavior and Development*, 6(4):485–489, 1983.

Chelsea Finn. *Learning to Learn with Gradients*. PhD thesis, UC Berkeley, 2018.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In Jennifer G. Dy and Andreas Krause, editors, *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*, pages 1514–1523. PMLR, 2018.

Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694, 2020.

Sébastien Forestier and Pierre-Yves Oudeyer. Modular active curiosity-driven discovery of tool use. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3965–3972. IEEE, 2016a.

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

Sébastien Forestier and Pierre-Yves Oudeyer. Modular Active Curiosity-Driven Discovery of Tool Use. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3965–3972. IEEE, 2016b.

Pierre Fournier, Olivier Sigaud, and Mohamed Chetouani. Combining artificial curiosity and tutor guidance for environment exploration. In *Workshop on Behavior Adaptation, Interaction and Learning for Assistive Robotics at IEEE RO-MAN*, pages 1–8, Lisbon, Portugal, 2017.

Pierre Fournier, Cédric Colas, Mohamed Chetouani, and Olivier Sigaud. Clic: Curriculum learning and imitation for object control in non-rewarding environments. *IEEE Transactions on Cognitive and Developmental Systems*, 2019.

Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019.

Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Anne L Fulkerson and Sandra R Waxman. Words (but not tones) facilitate object categorization: Evidence from 6-and 12-month-olds. *Cognition*, 105(1):218–228, 2007.

Alexander Gajewski, Jeff Clune, Kenneth O Stanley, and Joel Lehman. Evolvability es: scalable and direct optimization of evolvability. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 107–115, 2019.

Claudio Gallicchio and Alessio Micheli. Graph echo state networks. In *The 2010 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2010.

Dirk Geeraerts. *Cognitive linguistics: Basic readings*, volume 34. Walter de Gruyter, 2006.

Dedre Gentner and Christian Hoyos. Analogy and Abstraction. *Topics in Cognitive Science*, 9(3):672–693, July 2017. ISSN 17568757. doi: 10.1111/tops.12278.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

Lila R Gleitman, Elissa L Newport, and Henry Gleitman. The current status of the motherese hypothesis. *Journal of child language*, 11(1):43–79, 1984.

Peter Godfrey-Smith. Theory and reality. In *Theory and Reality*. University of Chicago Press, 2021.

Michael H Goldstein, Andrew P King, and Meredith J West. Social interaction shapes babbling: Testing parallels between birdsong and speech. *Proceedings of the National Academy of Sciences*, 100(13):8030–8035, 2003.

Noah D. Goodman and Michael C. Frank. Pragmatic language interpretation as probabilistic inference. *Trends in Cognitive Sciences*, 20(11):818–829, 2016. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2016.08.005.

Noah D Goodman, Joshua B Tenenbaum, and Tobias Gerstenberg. Concepts in a probabilistic language of thought. Technical report, Center for Brains, Minds and Machines (CBMM), 2014.

Alison Gopnik and Andrew N Meltzoff. *Words, thoughts, and theories*. Mit Press, 1997.

Alison Gopnik, Andrew N Meltzoff, and Patricia K Kuhl. *The scientist in the crib: Minds, brains, and how children learn*. William Morrow & Co, 1999.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *ArXiv preprint*, abs/1611.07507, 2016. URL https://arxiv.org/abs/1611.07507.

H. P. Grice. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, New York, 1975.

Jonathan Grizou, Manuel Lopes, and Pierre-Yves Oudeyer. Robot learning simultaneously a task and how to interpret human instructions. In *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–8. IEEE, 2013.

Jonathan Grizou, Iñaki Iturrate, Luis Montesano, Pierre-Yves Oudeyer, and Manuel Lopes. Interactive learning from unlabeled instructions. In Nevin L. Zhang and Jin Tian, editors, *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI 2014, Quebec City, Quebec, Canada, July 23-27, 2014*, pages 290–299. AUAI Press, 2014.

Benjamin N Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proceedings of the 12th international conference on World Wide Web*, pages 48–57, 2003.

Romana Gruber, Martina Schiestl, Markus Boeckle, Anna Frohnwieser, Rachael Miller, Russell D Gray, Nicola S Clayton, and Alex H Taylor. New caledonian crows use mental representations to solve metatool problems. *Current Biology*, 29 (4):686–692, 2019.

Thomas R Gruber. Nature, nurture, and knowledge acquisition. *International journal of human-computer studies*, 71(2):191–194, 2013.

Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 922–929, 2019.

Yijie Guo, Jongwook Choi, Marcin Moczulski, Shengyu Feng, Samy Bengio, Mohammad Norouzi, and Honglak Lee. Memory based trajectory-conditioned policies for learning from sparse rewards. *Advances in Neural Information Processing Systems*, 33:4333–4345, 2020.

Zhaohan Daniel Guo and Emma Brunskill. Directed exploration for reinforcement learning. *arXiv preprint arXiv:1906.07805*, 2019.

Hyowon Gweon. Inferential social learning: Cognitive foundations of human social learning and teaching. *Trends in Cognitive Sciences*, 25(10):896–910, 2021.

Hyowon Gweon, Joshua B Tenenbaum, and Laura E Schulz. Infants consider both the sample and the sampling process in inductive generalization. *Proceedings of the National Academy of Sciences*, 107(20):9066–9071, 2010.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Marshall M Haith and Janette B Benson. Infant cognition. *None*, 1998.

John Burdon Sanderson Haldane. The interaction of nature and nurture. *Annals of eugenics*, 13(1):197–205, 1946.

William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

Jessica B Hamrick, Andrew J Ballard, Razvan Pascanu, Oriol Vinyals, Nicolas Heess, and Peter W Battaglia. Metacontrol for adaptive imagination-based optimization. *arXiv preprint arXiv:1705.02670*, 2017.

Jessica B Hamrick, Kelsey R Allen, Victor Bapst, Tina Zhu, Kevin R McKee, Joshua B Tenenbaum, and Peter W Battaglia. Relational inductive bias for physical construction in humans and machines. *ArXiv preprint*, abs/1806.01203, 2018. URL https://arxiv.org/abs/1806.01203.

Michael J Hannafin, Craig Hall, Susan Land, and Janette Hill. Learning in open-ended environments: Assumptions, methods, and implications. *Educational Technology*, 34(8):48–55, 1994.

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

Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. *arXiv preprint arXiv:1907.08225*, 2019.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

Susan J Hespos and Renée Baillargeon. Reasoning about containment events in very young infants. *Cognition*, 78(3):207–245, 2001.

Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John P. Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *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*, pages 3223–3230. AAAI Press, 2018.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29 (6):82–97, 2012.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Mark K. Ho, Michael L. Littman, James MacGlashan, Fiery Cushman, and Joseph L. Austerweil. Showing versus doing: Teaching by demonstration. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3027–3035, 2016.

Mark K Ho, James MacGlashan, Michael L Littman, and Fiery Cushman. Social is special: A normative framework for teaching with and learning from evaluative feedback. *Cognition*, 167:91–106, 2017.

Dr JPE Hodgson. First order logic. *Saint Joseph's University, Philadelphia*, 1995.

Thomas Hoffmann. Creativity and Construction Grammar: Cognitive and Psychological Issues. *Zeitschrift für Anglistik und Amerikanistik*, 66(3):259–276, 2018. ISSN 2196-4726, 0044-2305. doi: 10.1515/zaa-2018-0024.

Ronald A Howard. Dynamic programming and markov processes. *None*, 1960.

George Edward Hughes, Max J Cresswell, and Mary Meyerhoff Cresswell. *A new introduction to modal logic.* Psychology Press, 1996.

Clark L Hull. A behavior system; an introduction to behavior theory concerning the individual organism. *None*, 1952.

Clark Leonard Hull. Principles of behavior: An introduction to behavior theory. *None*, 1943.

Broekens? J. and Mohamed Chetouani. Towards transparent robot learning through tdrl-based emotional expressions. *IEEE Transactions on Affective Computing*, pages 1–1, 2019. doi: 10.1109/TAFFC.2019.2893348.

Julian Jara-Ettinger. Theory of mind as inverse reinforcement learning. *Current Opinion in Behavioral Sciences*, 29:105–110, 2019.

Firas Jarboui and Ahmed Akakzia. Delayed geometric discounts: An alternative criterion for reinforcement learning. 2021.

Hong Jun Jeon, Smitha Milli, and Anca D. Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 9414–9426, 2019.

Mark Johnson. *The body in the mind: The bodily basis of meaning, imagination, and reason.* University of Chicago press, 2013.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099, 1993.

Sham Kakade. Optimizing average reward using discounted rewards. In *International Conference on Computational Learning Theory*, pages 605–615. Springer, 2001.

Immanuel Kant. Critique of pure reason. 1781. *Modern Classical Philosophers, Cambridge, MA: Houghton Mifflin*, pages 370–456, 1908.

Frederic Kaplan and Verena V Hafner. The challenges of joint attention. *Interaction Studies*, 7(2):135–169, 2006.

Tristan Karch, Cédric Colas, Laetitia Teodorescu, Clément Moulin-Frier, and Pierre-Yves Oudeyer. Deep sets for generalization in RL. *ArXiv preprint*, abs/2003.09443, 2020. URL https://arxiv.org/abs/2003.09443.

Suzanne Kemmer and Michael Barlow. Introduction: A usage-based conception of language. *Usage-based models of language*, pages 7–28, 2000.

Charles Kemp and Joshua B Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692, 2008.

Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2021.

W. Bradley Knox and Peter Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Machine learning*, 84(1):171–203, 2011.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pages 2747–2755. PMLR, 2018.

Lyudmyla F Kozachenko and Nikolai N Leonenko. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16, 1987.

Victoria Krakovna, Laurent Orseau, Miljan Martic, and Shane Legg. Measuring and avoiding side effects using relative reachability. *arXiv preprint arXiv:1806.01186*, 2018.

Yingyi Kuang, Abraham Itzhak Weinberg, George Vogiatzis, and Diego R Faria. Goal density-based hindsight experience prioritization for multi-goal robot manipulation reinforcement learning. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 432–437. IEEE, 2020.

Patricia K Kuhl. Early language acquisition: cracking the speech code. *Nature reviews neuroscience*, 5(11):831–843, 2004.

Johannes Kulick, Marc Toussaint, Tobias Lang, and Manuel Lopes. Active learning for teaching a robot grounded relational symbols. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

Marjan Laal. Lifelong learning: What does it mean? *Procedia-Social and Behavioral Sciences*, 28:470–474, 2011.

Yaqing Lai, Wufan Wang, Yunjie Yang, Jihong Zhu, and Minchi Kuang. Hindsight planner. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 690–698, 2020.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266): 1332–1338, 2015.

Brenden M Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

Susan M Land. Cognitive requirements for learning with open-ended learning environments. *Educational Technology Research and Development*, 48(3):61–78, 2000.

John B. Lanier, Stephen McAleer, and Pierre Baldi. Curiosity-driven multi-criteria hindsight experience replay. *CoRR*, abs/1906.03710, 2019.

Adrien Laversanne-Finot, Alexandre Péré, and Pierre-Yves Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. *ArXiv preprint*, abs/1807.01521, 2018. URL https://arxiv.org/abs/1807.01521.

John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1674, 2018.

Joel Lehman. Evolution through the search for novelty. ., 2012.

Borja G León, Murray Shanahan, and Francesco Belardinelli. Systematic generalisation through task temporal logic and deep reinforcement learning. *arXiv preprint arXiv:2006.08767*, 2020.

Alan M Leslie. Tomm, toby, and agency: Core architecture and domain specificity. *Mapping the mind: Domain specificity in cognition and culture*, 29:119–48, 1994.

Alan M Leslie. Developmental parallels in understanding minds and bodies. *Trends in cognitive sciences*, 9(10):459–462, 2005.

Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. *ArXiv preprint*, abs/1912.11032, 2019. URL `https://arxiv.org/abs/1912.11032`.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

Christina Lichtenthäler and Alexandra Kirsch. Legibility of robot behavior: A literature review. *https://hal.archives-ouvertes.fr/hal-01306977*, 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Angelica Lim and Hiroshi G Okuno. The mei robot: towards using motherese to develop multimodal emotional intelligence. *IEEE Transactions on Autonomous Mental Development*, 6(2):126–138, 2014.

Jinying Lin, Zhen Ma, Randy Gomez, Keisuke Nakamura, Bo He, and Guangliang Li. A review on interactive reinforcement learning from human social feedback. *IEEE Access*, 8:120757–120765, 2020.

Jessica Lindblom and Tom Ziemke. Social situatedness: Vygotsky and beyond. In *Proceedings of the Second International Workshop on Epigenetic Robotics. Modelling cognitive development in robotic systems*, pages 71–78. Lund University Cognitive Studies, 2002.

Jessica Lindblom and Tom Ziemke. Social situatedness of natural and artificial intelligence: Vygotsky and beyond. *Adaptive Behavior*, 11(2):79–96, 2003.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

Manuel Lopes and Pierre-Yves Oudeyer. The strategic student approach for life-long exploration and learning. In *IEEE International Conference on Development and Learning and Epigenetic Robotics*, pages 1–8. IEEE, 2012.

Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. *Advances in neural information processing systems*, 25, 2012.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*, 2019.

Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection Science*, 15(4):151–190, 2003.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Gary Lupyan. Linguistically modulated perception and cognition: The label-feedback hypothesis. *Frontiers in psychology*, 3:54, 2012.

Corey Lynch and Pierre Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

James MacGlashan and Michael L. Littman. Between imitation and intention learning. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3692–3698. AAAI Press, 2015.

Jean M. Mandler. Preverbal representation and language. *Language and space*, page 365, 1999.

Jean M Mandler. On the spatial foundations of the conceptual system and its enrichment. *Cognitive science*, 36(3):421–451, 2012.

Jean M Mandler and Cristóbal Pagán Cánovas. On defining image schemas. *Language and cognition*, 6(4):510–532, 2014.

Daniel J Mankowitz, Augustin Žídek, André Barreto, Dan Horgan, Matteo Hessel, John Quan, Junhyuk Oh, Hado van Hasselt, David Silver, and Tom Schaul. Unicorn: Continual learning with a universal, off-policy agent. *arXiv preprint arXiv:1802.08294*, 2018.

Ellen M Markman. *Categorization and naming in children: Problems of induction.* mit Press, 1989.

Robert McCarthy and Stephen J Redmond. Imaginary hindsight experience replay: Curious model-based learning for sparse reward tasks. *arXiv preprint arXiv:2110.02414*, 2021.

Laraine McDonough, Soonja Choi, and Jean M Mandler. Understanding spatial relations: Flexible infants, lexical adults. *Cognitive psychology*, 46(3):229–259, 2003.

Andrew N Meltzoff. Infant imitation after a 1-week delay: long-term memory for novel acts and multiple stimuli. *Developmental psychology*, 24(4):470, 1988.

Andrew N Meltzoff. Born to learn: What infants learn from watching us. *The role of early experience in infant development*, pages 1–10, 1999.

Andrew N Meltzoff. 'like me': a foundation for social cognition. *Developmental science*, 10(1):126–134, 2007.

Andrew N Meltzoff and M Keith Moore. Newborn infants imitate adult facial gestures. *Child development*, pages 702–709, 1983.

Elliott Mendelson. *Introduction to mathematical logic.* Chapman and Hall/CRC, 2009.

Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021.

Henok Mengistu, Joel Lehman, and Jeff Clune. Evolvability search: directly selecting for evolvability in order to study and produce it. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 141–148, 2016.

Csikszentmihalyi Mihaly. *Beyond boredom and anxiety: experiencing flow in work and play.* Jossey-Bass Publishers, 2000.

Pasquale Minervini, Matko Bosnjak, Tim Rocktäschel, and Sebastian Riedel. Towards neural theorem proving at scale. *arXiv preprint arXiv:1807.08204*, 2018.

P Read Montague, Peter Dayan, and Terrence J Sejnowski. A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of neuroscience*, 16(5):1936–1947, 1996.

David S Moore. *The Dependent Gene: The Fallacy of" nature Vs. Nurture"*. Macmillan, 2003.

David S Moore. *The developing genome: An introduction to behavioral epigenetics*. Oxford University Press, 2015.

Yugi Nagai and Katharina J. Rohlfing. Computational analysis of motionese toward scaffolding robot action learning. *IEEE Transactions on Autonomous Mental Development*, 1(1):44–54, 2009. doi: 10.1109/TAMD.2009.2021090.

Yukie Nagai and Katharina J. Rohlfing. Can motionese tell infants and robots" what to imitate". In *Proceedings of the 4th International Symposium on Imitation in Animals and Artifacts*, pages 299–306. Citeseer, 2007.

Ashvin Nair, Shikhar Bahl, Alexander Khazatsky, Vitchyr Pong, Glen Berseth, and Sergey Levine. Contextual imagined goals for self-supervised robotic learning. *ArXiv preprint*, abs/1910.11670, 2019. URL https://arxiv.org/abs/1910.11670.

Ashvin Nair, Shikhar Bahl, Alexander Khazatsky, Vitchyr Pong, Glen Berseth, and Sergey Levine. Contextual imagined goals for self-supervised robotic learning. In *Conference on Robot Learning*, pages 530–539. PMLR, 2020.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

Anis Najar, Olivier Sigaud, and Mohamed Chetouani. Training a robot with evaluative feedback and unlabeled guidance signals. In *25th IEEE International Symposium on Robot and Human Interactive Communication*, pages 261–266. IEEE, 2016.

Anis Najar, Olivier Sigaud, and Mohamed Chetouani. Interactively shaping robot behaviour with unlabeled human instructions. *Autonomous Agents and Multi-Agent Systems*, 34:1–35, 2020.

David Navon. Forest before trees: The precedence of global features in visual perception. *Cognitive psychology*, 9(3):353–383, 1977.

Chrystopher L. Nehaniv and Kerstin Dautenhahn. The correspondence problem. *Imitation in animals and artifacts*, 41, 2002.

Allen Newel and Herbert A Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

Allen Newell. *Unified theories of cognition*. Harvard University Press, 1994.

Allen Newell and Herbert Simon. The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3):61–79, 1956.

Allen Newell and Herbert A Simon. Computer science as empirical inquiry: Symbols and search. In *ACM Turing award lectures*, page 1975. 2007.

Allen Newell, John C Shaw, and Herbert A Simon. Report on a general problem solving program. In *IFIP congress*, volume 256, page 64. Pittsburgh, PA, 1959.

Dung Nguyen, Svetha Venkatesh, Phuoc Nguyen, and Truyen Tran. Theory of mind with guilt aversion facilitates cooperative reinforcement learning. In *Asian Conference on Machine Learning*, pages 33–48. PMLR, 2020.

Khanh Nguyen, Dipendra Misra, Robert E. Schapire, Miroslav Dudík, and Patrick Shafto. Interactive learning from activity description. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8096–8108. PMLR, 2021. URL http://proceedings.mlr.press/v139/nguyen21e.html.

Sao Mai Nguyen and Pierre-Yves Oudeyer. Active Choice of Teachers, Learning Strategies and Goals for a Socially Guided Intrinsic Motivation Learner. *Paladyn*, 3(3):136–146, 2012. Publisher: Springer.

Sao Mai Nguyen and Pierre-Yves Oudeyer. Socially guided intrinsic motivation for robot learning of motor skills. *Autonomous Robots*, 36(3):273–294, 2014a.

Sao Mai Nguyen and Pierre-Yves Oudeyer. Socially Guided Intrinsic Motivation for Robot Learning of Motor Skills. *Autonomous Robots*, 36(3):273–294, 2014b. Publisher: Springer.

Andreas Nieder. Prefrontal cortex and the evolution of symbolic reference. *Current opinion in neurobiology*, 19(1):99–108, 2009.

Dennis Normile. Nature from nurture, 2016.

Todd Oakley. Image schemas. *The Oxford handbook of cognitive linguistics*, pages 214–235, 2007.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pages 2661–2670. PMLR, 2017.

James Olds and Peter Milner. Positive reinforcement produced by electrical stimulation of septal area and other regions of rat brain. *Journal of comparative and physiological psychology*, 47(6):419, 1954.

P-Y Oudeyer, Jacqueline Gottlieb, and Manuel Lopes. Intrinsic motivation, curiosity, and learning: Theory and applications in educational technologies. *Progress in brain research*, 229:257–284, 2016.

Pierre-Yves Oudeyer and Frederic Kaplan. What Is Intrinsic Motivation? A Typology of Computational Approaches. *Frontiers in neurorobotics*, 1:6, 2007. Publisher: Frontiers.

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

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

Stephen E Palmer. Visual perception and world knowledge: Notes on a model of sensory-cognitive interaction. *Explorations in cognition*, pages 279–307, 1975.

Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pages 5062–5071. PMLR, 2019.

Sujoy Paul, Jeroen Vanbaar, and Amit Roy-Chowdhury. Learning from trajectories via subgoal discovery. *Advances in Neural Information Processing Systems*, 32, 2019.

Ivan Pavlov. Physiology of digestion. In *Nobel lectures: Physiology or medicine*, pages 141–155. Elsevier, 1904.

Ivan Petrovitch Pavlov and William Gantt. Lectures on conditioned reflexes: Twenty-five years of objective study of the higher nervous activity (behaviour) of animals. *None*, 1928.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

Maxime Petit, Stéphane Lallée, Jean-David Boucher, Grégoire Pointeau, Pierrick Cheminade, Dimitri Ognibene, Eris Chinellato, Ugo Pattacini, Ilaria Gori, Uriel Martinez-Hernandez, et al. The coordinating role of language in real-time multimodal learning of cooperative tasks. *IEEE Transactions on Autonomous Mental Development*, 5(1):3–17, 2012.

Jean Piaget. *Le langage et la pensée chez l'enfant*, volume 1. Delachaux and Niestlé, 1923.

Jean Piaget. The construction of reality in the child. *Journal of Consulting Psychology*, 19(1):77, 1955.

Jean Piaget. *The development of thought: Equilibration of cognitive structures.* Viking, 1977. (Trans A. Rosin).

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7750–7761. PMLR, 2020.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

D Pomerleau. An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems*, 1, 1998.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep RL: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

Danil V Prokhorov and Donald C Wunsch. Adaptive critic designs. *IEEE transactions on Neural Networks*, 8(5):997–1007, 1997.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

Sebastien Racaniere, Andrew K Lampinen, Adam Santoro, David P Reichert, Vlad Firoiu, and Timothy P Lillicrap. Automated curricula through setter-solver interactions. *ArXiv preprint*, abs/1909.12892, 2019. URL `https://arxiv.org/abs/1909.12892`.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv preprint*, abs/2112.11446, 2021. URL `https://arxiv.org/abs/2112.11446`.

Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7:1, 2019.

Siddharth Reddy, Anca D. Dragan, Sergey Levine, Shane Legg, and Jan Leike. Learning human objectives by evaluating hypothetical behavior. 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*, pages 8020–8029. PMLR, 2020.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

Joseph Reisinger, Kenneth O Stanley, and Risto Miikkulainen. Towards an empirical measure of evolvability. In *Proceedings of the 7th annual workshop on Genetic and evolutionary computation*, pages 257–264, 2005.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. *Advances in Neural Information Processing Systems*, 32, 2019.

Frank Röder, Manfred Eppe, and Stefan Wermter. Grounding hindsight instructions in multi-goal reinforcement learning for robotics. *arXiv preprint arXiv:2204.04308*, 2022.

Matthias Rolf, Jochen J Steil, and Michael Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, 2(3):216–229, 2010.

Joshua Romoff, Peter Henderson, Ahmed Touati, Emma Brunskill, Joelle Pineau, and Yann Ollivier. Separating value functions across time-scales. In *International Conference on Machine Learning*, pages 5468–5477. PMLR, 2019.

Barak Rosenshine, Carla Meister, et al. The use of scaffolds for teaching higher-level cognitive strategies. *Educational leadership*, 49(7):26–33, 1992.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Julien Roy, Roger Girgis, Joshua Romoff, Pierre-Luc Bacon, and Christopher Pal. Direct behavior specification via constrained reinforcement learning. *arXiv preprint arXiv:2112.12228*, 2021.

Joshua S Rule, Joshua B Tenenbaum, and Steven T Piantadosi. The child as hacker. *Trends in cognitive sciences*, 24(11):900–915, 2020.

Catherine Saint-Georges, Mohamed Chetouani, Raquel Cassel, Fabio Apicella, Ammar Mahdhaoui, Filippo Muratori, Marie-Christine Laznik, and David Cohen. Motherese in interaction: At the cross-road of emotion and cognition? (a systematic review). *PLOS ONE*, 8(10):null, 10 2013. doi: 10.1371/journal.pone.0078103.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR, 2018.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

Adam Santoro, Andrew Lampinen, Kory Mathewson, Timothy Lillicrap, and David Raposo. Symbolic behaviour in artificial intelligence. *arXiv preprint arXiv:2102.03406*, 2021.

William Saunders, Girish Sastry, Andreas Stuhlmueller, and Owain Evans. Trial without error: Towards safe reinforcement learning via human intervention. *arXiv preprint arXiv:1707.05173*, 2017.

Franco Scarselli, Sweah Liang Yong, Marco Gori, Markus Hagenbuchner, Ah Chung Tsoi, and Marco Maggini. Graph neural networks for ranking web pages. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pages 666–672. IEEE, 2005.

Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):537–547, 2003.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE transactions on autonomous mental development*, 2(3):230–247, 2010.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR, abs/1502.05477*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

W. Schultz, P. Dayan, and P. R. Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

Anton Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the tenth international conference on machine learning*, volume 298, pages 298–305, 1993.

Daniel Seita, David Chan, Roshan Rao, Chen Tang, Mandi Zhao, and John Canny. Zpd teaching strategies for deep reinforcement learning from demonstrations. *ArXiv preprint*, abs/1910.12154, 2019. URL https://arxiv.org/abs/1910.12154.

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing*, pages 362–373. Springer, 2018.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Robert S Siegler. *Emerging minds: The process of change in children's thinking*. Oxford University Press, 1998.

Olivier Sigaud, Hugo Caselles-Dupré, Cédric Colas, Ahmed Akakzia, Pierre-Yves Oudeyer, and Mohamed Chetouani. Towards teachable autonomous agents. *arXiv preprint arXiv:2105.11977*, 2021.

David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.

BF Skinner. The behavior of organisms. *New York, Appleton-Century-Crofts*, 1938.

Vladimir M Sloutsky. From perceptual categories to concepts: What develops? *Cognitive science*, 34(7):1244–1286, 2010.

Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005.

Raymond M Smullyan. *First-order logic*. Courier Corporation, 1995.

David Sobel and Jessica Sommerville. The importance of discovery in children's causal learning from interventions. *Frontiers in Psychology*, 1:176, 2010.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

Aleksandr Sokolov. *Inner Speech and Thought.* New York: Plenum Press, 1972.

Elizabeth Spelke. Initial knowledge: Six suggestions. *Cognition*, 50(1-3):431–445, 1994.

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

Luc Steels. The autotelic principle. In *Embodied artificial intelligence*, pages 231–242. Springer, 2004.

Luc Steels. *The Talking Heads experiment: Origins of words and meanings*, volume 1. Language Science Press, 2015.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.

Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. Open-ended learning leads to generally capable agents. *ArXiv preprint*, abs/2107.12808, 2021. URL `https://arxiv.org/abs/2107.12808`.

Freek Stulp and Stefan Schaal. Hierarchical reinforcement learning with movement primitives. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 231–238. IEEE, 2011.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Gerald J. Sussman. A computational model of skill acquisition. Technical report, HIT Technical Report AI TR-297, 1973.

Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13:12, 2019.

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

Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999a.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999b.

Richard S Sutton et al. *Introduction to reinforcement learning.* MIT Press, 1998.

Umar Syed and Robert E Schapire. A reduction from apprenticeship learning to classification. *Advances in neural information processing systems*, 23, 2010.

Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pages 1032–1039, 2008.

Austin Tate et al. Interacting goals and their use. In *IJCAI*, volume 10, pages 215–218, 1975.

Jessica Taylor, Eliezer Yudkowsky, Patrick LaVictoire, and Andrew Critch. Alignment for advanced machine learning systems. *Ethics of Artificial Intelligence*, pages 342–382, 2016.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Approaching the symbol grounding problem with probabilistic graphical models. *AI magazine*, 32(4):64–76, 2011.

Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022): 1279–1285, 2011.

Chen Tessler and Shie Mannor. Reward tweaking: Maximizing the total reward while planning for short horizons. *arXiv preprint arXiv:2002.03327*, 2020.

Esther Thelen and Linda B Smith. *A dynamic systems approach to the development of cognition and action.* MIT press, 1996.

Andrea L Thomaz and Cynthia Breazeal. Experiments in socially guided exploration: Lessons learned in building robots that learn with and without human teachers. *Connection Science*, 20(2-3):91–110, 2008a.

Andrea L Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6-7):716–737, 2008b.

Andrea L Thomaz, Guy Hoffman, and Cynthia Breazeal. Reinforcement learning with human teachers: Understanding how people want to teach robots. In *ROMAN 2006-The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 352–357. IEEE, 2006.

E. L. Thorndike. *Animal Intelligence.* MacMillan Company, New York, 1911.

Edward L Thorndike. Animal intelligence: An experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements*, 2(4):i, 1898.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Michael Tomasello. The social bases of language acquisition. *Social development*, 1 (1):67–87, 1992.

Michael Tomasello. Emulation learning and cultural learning. *Behavioral and Brain Sciences*, 21(5):703–704, 1998.

Michael Tomasello. The human adaptation for culture. *Annual review of anthropology*, 28, 1999.

Michael Tomasello. *Constructing a language: A usage-based theory of language acquisition*. Harvard university press, 2005.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4950–4957. ijcai.org, 2018. doi: 10.24963/ijcai.2018/687.

Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Alan Turing. Intelligent machinery. *B. Jack Copeland*, page 395, 1948.

Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

Emre Ugur, Erhan Oztop, and Erol Sahin. Goal emulation and planning in perceptual space using learned affordances. *Robotics and Autonomous Systems*, 59(7-8): 580–595, 2011.

Peter Vamplew, Benjamin J Smith, Johan Källström, Gabriel Ramos, Roxana Rădulescu, Diederik M Roijers, Conor F Hayes, Fredrik Heintz, Patrick Mannion, Pieter JK Libin, et al. Scalar reward is not enough: A response to silver, singh, precup and sutton (2021). *Autonomous Agents and Multi-Agent Systems*, 36(2):1–19, 2022.

Sjoerd Van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

James W Varni, O Ivar Lovaas, Robert L Koegel, and Nancy L Everett. An analysis of observational learning in autistic and normal children. *Journal of Abnormal Child Psychology*, 7(1):31–43, 1979.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *ArXiv preprint*, abs/1707.08817, 2017. URL https://arxiv.org/abs/1707.08817.

Arthur F Veinott. On finding optimal policies in discrete dynamic programming with no discounting. *The Annals of Mathematical Statistics*, 37(5):1284–1294, 1966.

Natalia Vélez and Hyowon Gweon. Learning from other minds: An optimistic critique of reinforcement learning models of social learning. *Current Opinion in Behavioral Sciences*, 38:110–115, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

Anna-Lisa Vollmer, Britta Wrede, Katharina J. Rohlfing, and Pierre-Yves Oudeyer. Pragmatic frames for teaching and learning in human–robot interaction: Review and challenges. *Frontiers in neurorobotics*, 10:1–10, 2016.

L. S. Vygotsky. Tool and Symbol in Child Development. In *Mind in Society*, chapter Tool and Symbol in Child Development, pages 19–30. Harvard University Press, 1978a. ISBN 0674576292. doi: 10.2307/j.ctvjf9vz4.6.

L. S. Vygotsky. Tool and Symbol in Child Development. In *Mind in Society*, chapter Tool and Symbol in Child Development, pages 19–30. Harvard University Press, 1978b. ISBN 0674576292. doi: 10.2307/j.ctvjf9vz4.6.

Lev S. Vygotsky. Tool and symbol in child development. *The Vygotsky reader*, 1994.

Andrey Vyshedskiy. 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.

Günter P Wagner and Lee Altenberg. Perspective: complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976, 1996.

Sebastian Wallkotter, Silvia Tulli, Ginevra Castellano, Ana Paiva, and Mohamed Chetouani. Explainable agents through social cues: A review, 2021.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016a.

Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *ArXiv preprint*, abs/1611.05763, 2016b. URL https://arxiv.org/abs/1611.05763.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1sqHMZCb.

David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. *Advances in neural information processing systems*, 30, 2017.

Chen-Yu Wei, Mehdi Jafarnia Jahromi, Haipeng Luo, Hiteshi Sharma, and Rahul Jain. Model-free reinforcement learning in infinite-horizon average-reward markov decision processes. In *International conference on machine learning*, pages 10170–10180. PMLR, 2020.

Deena Skolnick Weisberg, Kathy Hirsh-Pasek, and Roberta Michnick Golinkoff. Guided play: Where curricular goals meet a playful pedagogy. *Mind, Brain, and Education*, 7(2):104–112, 2013.

Deena Skolnick Weisberg, Kathy Hirsh-Pasek, Roberta Michnick Golinkoff, Audrey K Kittredge, and David Klahr. Guided play: Principles and practices. *Current Directions in Psychological Science*, 25(3):177–182, 2016.

J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504): 599–600, 2001.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. *Advances in neural information processing systems*, 25, 2012.

Judith A Winn. Promises and challenges of scaffolded instruction. *Learning Disability Quarterly*, 17(1):89–104, 1994.

Terry Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.

Patrick H Winston. Learning structural descriptions from examples. *None*, 1970.

David Wood, Jerome S Bruner, and Gail Ross. The role of tutoring in problem solving. *Child Psychology & Psychiatry & Allied Disciplines*, 1976.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–707, 2017.

Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pages 6818–6827. PMLR, 2019.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

Kai Xu, Akash Srivastava, Dan Gutfreund, Felix Sosa, Tomer Ullman, Josh Tenenbaum, and Charles Sutton. A bayesian-symbolic approach to reasoning and learning in intuitive physics. *Advances in Neural Information Processing Systems*, 34: 2478–2490, 2021.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

Yue Yu, Patrick Shafto, Elizabeth Bonawitz, Scott C-H Yang, Roberta M Golinkoff, Kathleen H Corriveau, Kathy Hirsh-Pasek, and Fei Xu. The theoretical and methodological opportunities afforded by guided play with young children. *Frontiers in psychology*, 9:1152, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

Weishan Zhang, Liang Xu, Zhongwei Li, Qinghua Lu, and Yan Liu. A deep-intelligence framework for online video processing. *IEEE Software*, 33(2):44–51, 2016.

Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic Curriculum Learning through Value Disagreement. *arXiv:2006.09641 [cs, stat]*, June 2020. URL http://arxiv.org/abs/2006.09641. arXiv: 2006.09641.

Rui Zhao, Yang Gao, Pieter Abbeel, Volker Tresp, and Wei Xu. Mutual information state intrinsic control. *arXiv preprint arXiv:2103.08107*, 2021.

Li Zhou and Kevin Small. Inverse reinforcement learning with natural language goals. *ArXiv preprint*, abs/2008.06924, 2020. URL https://arxiv.org/abs/2008.06924.

Menghui Zhu, Minghuan Liu, Jian Shen, Zhicheng Zhang, Sheng Chen, Weinan Zhang, Deheng Ye, Yong Yu, Qiang Fu, and Wei Yang. Mapgo: Model-assisted policy optimization for goal-oriented tasks. *arXiv preprint arXiv:2105.06350*, 2021.

Xi Zhu, Zhendong Mao, Chunxiao Liu, Peng Zhang, Bin Wang, and Yongdong Zhang. Overcoming language priors with self-supervised learning for visual question answering. *arXiv preprint arXiv:2012.11528*, 2020.

Jordan Zlatev. The epigenesis of meaning in human beings, and possibly in robots. *Minds and Machines*, 11(2):155–195, 2001.